NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

VHDL MODELING AND SIMULATION FOR A DIGITAL TARGET IMAGING ARCHITECTURE FOR MULTIPLE LARGE TARGETS GENERATION

by

Håkan Bergön

September 2002

Thesis Advisor:	Douglas J. Fouts
Co-Advisors:	Man-Tak Shing
	Phillip E. Pace

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: VHDL Modeling and Simulation for a Digital Target Imaging Architecture for Multiple Large Targets Generation			**5. FUNDING NUMBERS**
6. AUTHOR(S) Håkan Bergön			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Research Laboratory			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

The subject of this thesis is to model and verify the correctness of the architecture of the Digital Image Synthesizer (DIS). The DIS, a system-on-a-chip, is especially useful as a counter-targeting repeater. It synthesizes the characteristic echo signature of a pre-selected target. The VHDL description of the DIS architecture was exported from Tanner S-Edit, modified, and simulated. Different software oriented verification approaches were researched and a White-box approach to functional verification was adopted. An algorithm based on the hardware functionality was developed to compare expected and simulated results. Initially, the architecture of one Range Bin Modulator was exported. Modifications to the VHDL source code included modeling of the behavior of the N-FET and P-FET transistors as well as Ground and Vdd (the voltages connected to the drains of the FETs). It also included renaming of entities to comply with VHDL naming conventions. Simulation results were compared to manual calculations and Matlab programs to verify the architecture. The procedure was repeated for the architecture of an Eight-Range Bin Modulator with equally successful results. VHDL was then used to create a super class of a 32-Range Bin Modulator. Test vectors developed in Matlab were used to yet again verify correct functionality.

14. SUBJECT TERMS Digital Image Synthesizer, Counter-Targeting Repeater, Range Bin Modulator, VHDL, White-box, Matlab			**15. NUMBER OF PAGES** 194
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

VHDL MODELING AND SIMULATION FOR A DIGITAL TARGET IMAGING ARCHITECTURE FOR MULTIPLE LARGE TARGETS GENERATION

Håkan Bergön
Major, Swedish Army
BSSE, Swedish National Defense College, 2000

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

AND

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author: Håkan P.I. Bergön

Approved by: Douglas J. Fouts, Thesis Advisor

Man-Tak Shing, Co-Advisor

Phillip E. Pace, Co-Advisor

Dan C. Boger, Chairman
Information Sciences Department

Luqi, Chairman
Software Engineering Program

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The subject of this thesis is to model and verify the correctness of the architecture of the Digital Image Synthesizer (DIS). The DIS, a system-on-a-chip, is especially useful as a counter-targeting repeater. It synthesizes the characteristic echo signature of a pre-selected target. The VHDL description of the DIS architecture was exported from Tanner S-Edit, modified, and simulated. Different software oriented verification approaches were researched and a White-box approach to functional verification was adopted. An algorithm based on the hardware functionality was developed to compare expected and simulated results. Initially, the architecture of one Range Bin Modulator was exported. Modifications to the VHDL source code included modeling of the behavior of the N-FET and P-FET transistors as well as Ground and Vdd (the voltages connected to the drains of the FETs). It also included renaming of entities to comply with VHDL naming conventions. Simulation results were compared to manual calculations and Matlab programs to verify the architecture. The procedure was repeated for the architecture of an Eight-Range Bin Modulator with equally successful results. VHDL was then used to create a super class of a 32-Range Bin Modulator. Test vectors developed in Matlab were used to yet again verify correct functionality.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to direct a special thanks to my family; Casey, Emma and Jack, for putting up with me during long periods of intensive studies at the Naval Postgraduate School.

I would also like to thank the Swedish National Defence College for believing in my ability to successfully complete this education.

Furthermore, I would like to thank Professor Phillip E. Pace for his support and friendship. Professor, you gave me constant encouragement and a challenging research project, you gave me the tools to succeed, –Thank You. I would also like to single out Professor Douglas J. Fouts who guided me through the first stumbling steps in implementing VHDL and patiently explained the hardware functionality of our project; - You are a big part of my success. Finally I would like to thank Professor Man-Tak Shing who agreed to take me under his wings and embraced me in the Software community, - Professor I learned a lot.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The subject of this thesis is to model and verify the correctness of the architecture of the Digital Image Synthesizer (DIS). The DIS, a system-on-a-chip, is especially useful as a counter-targeting repeater. It synthesizes the characteristic echo signature of a pre-selected target, i.e., the user has the opportunity to generate copies the echo signature and displace them. The V-H-D-L-description of the DIS architecture was exported from Tanner S-Edit, modified, and simulated. The advantages of using the VHDL text-based programming environment was explored in both creation of models and superior simulation speed. Different software oriented verification approaches were researched and a White-box approach to functional verification was adopted. An algorithm based on the hardware functionality was developed to compare expected and simulated results. Initially, the architecture of one Range Bin Modulator was exported. Modifications to the VHDL source code included modeling of the behavior of the N-FET and P-FET transistors as well as Ground and Vdd (the voltages connected to the drains of the FETs). It also included renaming of entities to comply with VHDL naming conventions. Simulation results were compared to manual calculations and Matlab programs to verify the architecture. The procedure was repeated for the architecture of an Eight-Range Bin Modulator with equally successful results. VHDL was then used to create a super class of a 32-Range Bin Modulator, again with its functionality verified by Matlab test vectors. Finally, two additional super classes of a 128- and a 512-Range Bin Modulator were programmed.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. DIGITAL IMAGE SYNTHESIZERS

1. Background

The threat of modern, wideband imaging synthetic aperture radar (SAR) and inverse synthetic aperture radar (ISAR) create a difficult ship defense problem. With image capability, one cannot simply transmit a false signal to counteract the missile radar, but must instead create an image resembling the image in an adversary's threat library.

The concept of image synthesizers is not new. Analog Image Synthesizers (AIS), using lengths of cable to delay interrogating signals, have been used as counter-targeting repeater decoys. AISs had serial taps along the length of the cable thereby creating different range-bins. Each tap modulated the signal in amplitude and/or frequency to synthesize reflections from surfaces within the specified range-bin. After summing the signals from the respective range-bins, the synthesized signal is retransmitted and returns a false echo.

The drawbacks with the analog systems, however, were that they were unreliable and hard to use. The AISs were noisy and could not store a signal over a long period of time and thereby reduced bandwidth and limited the size of the synthesized object. The cable length made the system bulky and unmanageable, and at the same time, prevented effective programming of the operating parameters.

The Digital Image Synthesizer (DIS) eliminates most of the drawbacks of the AIS. First, it is by no means bulky and future applications may, apart from ships, include aircrafts and Unmanned Aerial Vehicles (UAV). Second, the tapped delay line processors are capable of storing the signal for as long as necessary. Thus, the bandwidth is increased and synthesizing larger objects is possible. Third, the programmable nature of the respective range-bins facilitates movement of the DIS from one target type to another (Ref.[1]) and (Ref.[2]).

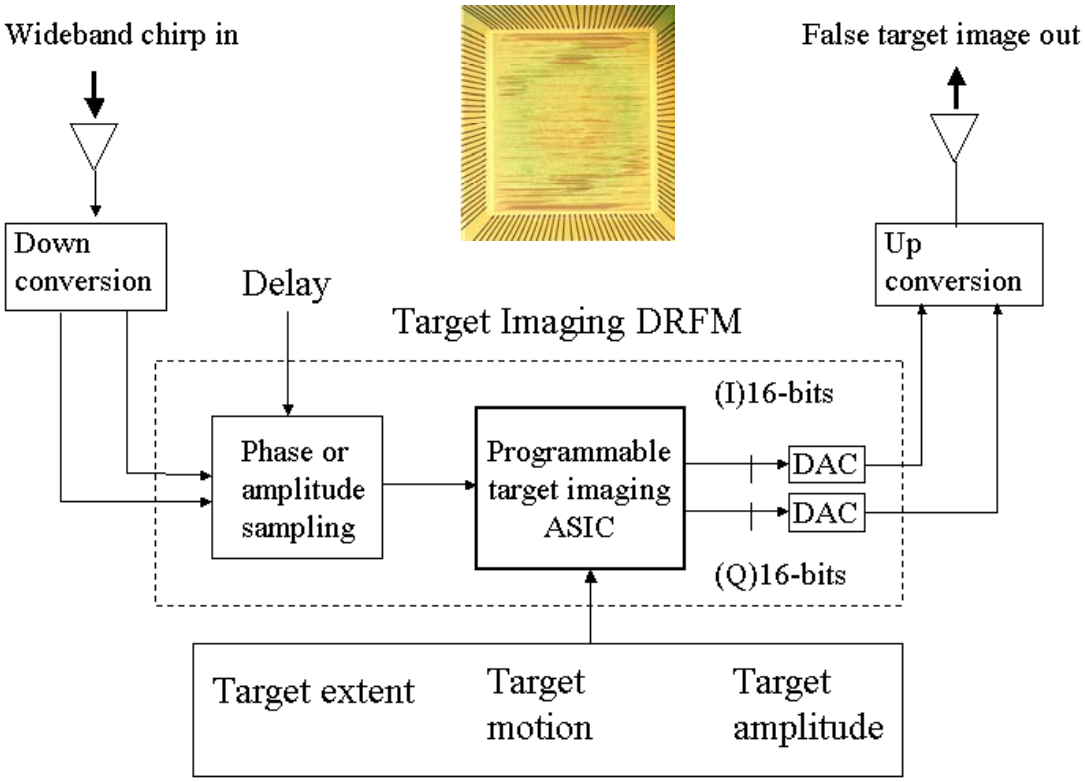


Figure 1. The DIS Concept. (After: a Presentation by Dr. Phillip Pace at Office of Naval Research (ONR) May 2001).

2. Functionality of the Digital Image Synthesizer

Figure 1 represents an overview of the technical approach to the DIS, and a functional block diagram representing is presented in Figure 2. The antenna receives a wideband chirp signal from interrogating search radar(s). The system receiver down-converts the signal and breaks it into In-Phase and Quadrature components (I and Q) where I is the real part and Q is the imaginary part of the signal. The signal information is then digitized and stored in a Digital Radio Frequency Memory (DRFM). The phase samples are then read serially from the DRFM into the DIS through the tapped delay line(s), or the range-bin processor(s).

The DIS ASIC is controlled by an off-chip microprocessor. A look-up table is used to generate the appropriate I and Q values after the implementation of a phase shift on the digitized phase samples. The amplitude required of the resulting data is controlled

by the microprocessor and is implemented by left shifts in the gain multiplier. Each range-bin processor performs summations in series. The last range-bin in the application produces a total sum representing a digital false target image sample. After digital to analog conversion of the I and Q, up-conversion and transmission of the false target occurs.

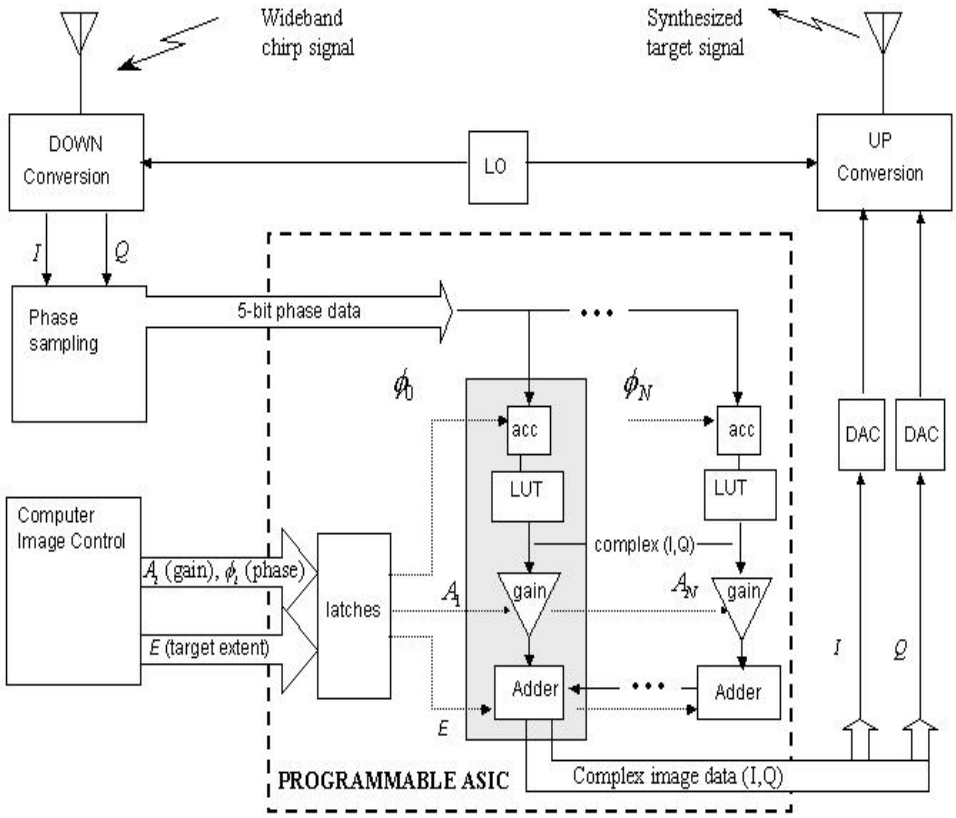


Figure 2. Block Diagram of the Technical Approach for the Digital Image Synthesizer. (From Ref.[2])

The range resolution of the DIS synthesized false target is determined by the resolution possible by each range-bin and the number of range-bins in series. The resolution can be calculated by:

$$R_R = \frac{C}{2 f_{cl}}$$

$$M_{SZ} = R_R * N_{RB}$$

where R_R is the range resolution of an individual range-bin, f_{cl} is the clock frequency of the chip and C is the speed of light. With a DIS operating at 600 MHz, the range resolution is 0.25 m. The maximum size of the synthesized false target M_{SZ} is then dependent of the number of range-bins N_{RB} (Ref.[12]).

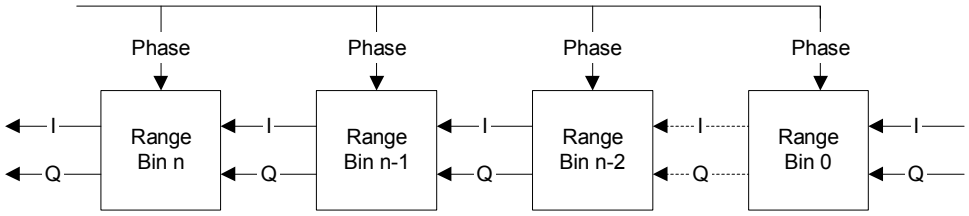


Figure 3. Architecture of DIS Implementation.

The plan is to eventually create a DIS with 512 range-bins operating at 600-800MHz. As seen in Figure 3, the phase of the range-bins are fed in parallel while I and Q are fed in series from one range-bin to the next.

The following real ISAR image, visual image and Matlab-synthesized image are an example and proof of concept of what this technique can provide (Ref. [12]).

Figure 4. USS Crocket and AN/APS-137 ISAR Image of the USS Crockett.

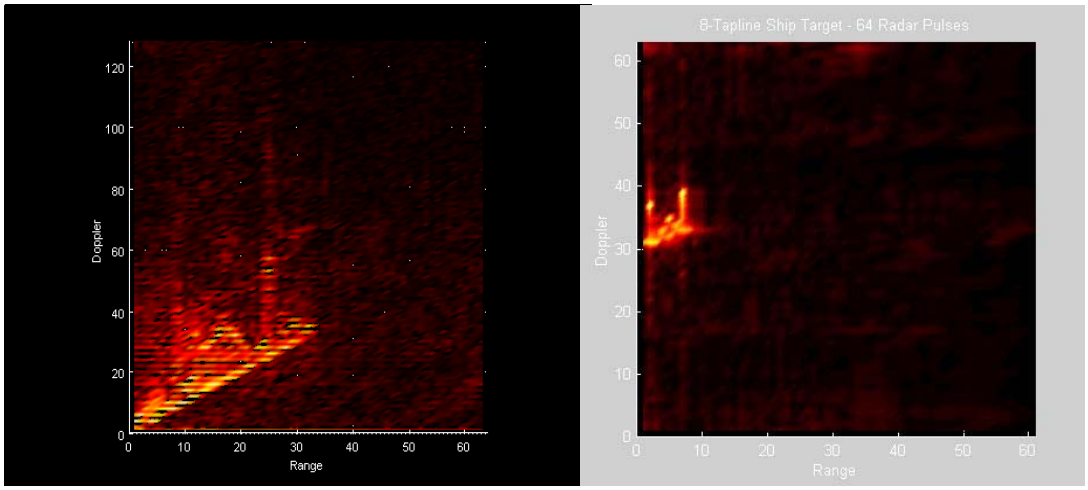


Figure 5. False Target Images Generated by a 32 Range-Bin, 256 Pulse Matlab Simulation (Left) and 8 Range-Bin Proof-of-Concept DIS Integrated Circuit (Right).

The CMOS proof of concept, 8 Range-bin, Integrated Circuit (IC) was developed using the Tanner Tools Pro IC design software package. This IC has been fabricated and tested and found fully functional at a 70 MHz clock speed (Ref.[12]). As is seen in Figure 4 and Figure 5 the synthesized images have a strong resemblance to the one generated by the ISAR.

B. PRINCIPAL CONTRIBUTIONS

The objective of the research in this thesis was to verify the design and functionality of the single Range-bin Modulator circuit as well as the 8-Range-bin Modulator circuit designed with Tanner Tools Pro. The verification was to occur using VHSIC Hardware Description Language (VHDL), where VHSIC in turn stands for Very High-Speed Integrated Circuits. The VHDL tool used was Active-HDL 5.1, by Aldec. Another goal of the research was to produce larger multiples of range-bins using VHDL. A 32-Range-bin processor was created through VHDL.

The first step was to export simple logic gates from S-Edit into a VHDL format. Simulations in which the result was known and obvious were then generated in order to understand the process.

Second, larger and more complex adders and registers were exported. The behaviors of the field effect transistors (FETs) were implemented in VHDL and the correct operations verified.

Third, the correctness of first one single and then eight combined range-bins was tested.

Finally, the 8-Range-bin processor was used in order to create a software super-class of 32 Range-bins.

In all instances, software was generated automatically, modified by hand, and tested in order to verify correctness of the designed component.

C. THESIS OUTLINE

The purpose of this thesis is to verify the circuit design and schematic of serially connected Range Bin Modulators operating at clock speeds of 600 MHz. The remainder of this thesis is organized as follows.

Chapter II presents the capabilities of VHDL as the means to design and/or verify digital circuit design.

Chapter III ventures into different methodologies of verification of a hardware design using software methods.

Chapter IV outlines the methodology and process of code extraction, as well as presents the modifications necessary in order to simulate the design in VHDL.

Chapter V presents the verification methodology used in this thesis. It displays obtained simulation results from different levels of the overall design.

Chapter VI summarizes the results of this thesis and makes recommendations on further verifications and the use of VHDL.

Appendix A contains a tutorial describing the process to follow to create a VHDL design using an externally created source file.

Appendix B contains a tutorial describing the process to follow to create a Test Bench using a saved wave form.

Appendix C contains VHDL code for a 1-bit adder.

Appendix D contains VHDL code and Test Bench for the single Range-bin modulator.

Appendix E contains VHDL code and Test Bench for the 8 Range-bin modulator.

Appendix F contains VHDL code and Test Bench for the 32 Range-bin modulator.

THIS PAGE INTENTIONALLY LEFT BLANK

II. CAPABILITIES OF VHDL

A. INTRODUCTION

1. History of VHDL

The acronym VHDL is a two-layer acronym that stands for **V**HSIC **H**ardware **D**escription **L**anguage, where VHSIC in turn stands for **V**ery **H**igh-**S**peed **I**ntegrated **C**ircuits. DoD initiated the VHDL program in 1980 to address the hardware life-cycle crises by improving documentation and reducing maintenance costs. By 1985, a team of DoD contractors, including TI and IBM, delivered the first version of the language. By 1987, VHDL had become an IEEE standard and by 1988, an ANSI standard. After the addition of some new features, the current standard of the language is IEEE 1076-1993. Drafts for a revised standard are currently in progress.

2. Digital Design Using HDL

The design of a digital system starts, as with other designs, with requirements specifications. Eventually a physical implementation of a chip is created through a stepwise, refined functional design. A typical activity flow in a top-down design environment can be seen in Figure 6.

As with software specifications, one major problem is capturing the client's requirements. This is perhaps a first indication of the use of formalism in hardware specifications, validation and verification.

After the top-level specification, decomposition from behavior to structure leads to the eventual physical design.

Historically different HDLs were appropriate for different levels of abstraction. Graphical editors were the design environment of choice, providing the hardware engineer with a "sense and feel" of the progress of the design. One such example is the Tanner Tools Pro S-Edit program used to design the schematics tested in this thesis. S-Edit consists of parts to design pictorial schematics. It does not, however, include a logic-level simulator such as Verilog or VHDL, but S-Edit, which is the pictorial schematic capable of generating and exporting VHDL code. This code is used throughout this thesis.

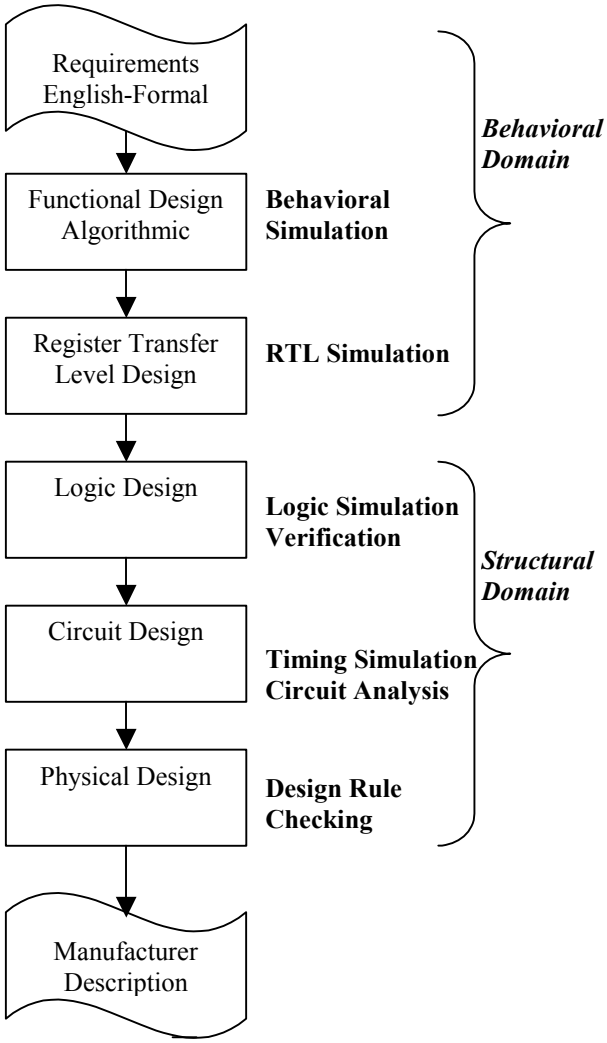


Figure 6. Typical Activity Flow.

This thesis spans the last activities of the behavioral domain as well as the logic simulation and verification.

In contrast to Tanner SPICE, VHDL and Verilog provide a series of constructs that can be applied at different levels of abstractions to provide multiple views of the system as exemplified in Figure 7. The languages are mainly text based, but graphical interfaces allow "old-fashioned" design. These HDLs are used throughout the development cycle by transforming from one level of abstraction, or so called synthesis, to another.

VHDL and Verilog are technology independent and not tied to a specific methodology. They can be used as a design tool for a custom or an ASIC chip as well as an FPGA.

The languages strongly resemble regular programming languages but are specially oriented to describe hardware structures and behaviors. One of the main differences is the ease at which parallel operations are implemented versus sequential ones.

The concept of Virtual Prototyping relies heavily on the capability of the HDL. Previously, the software that processed the data streams on a board design could not be tested until the hardware was available. However, with a HDL capable of describing the exact behavior of the components, it is possible to simulate the completed hardware for software development purposes.

3. Logic Synthesis

The real driver for the modern HDLs is the ability to move from one level of abstraction to another. Logic Synthesis can, for instance, transform a Register Transfer Level (RTL) description of a circuit into combinatorial logic. On this level, it is also possible to apply software verification techniques such as model checking and theorem proving. Logic synthesis is performed in two steps. First, the translations of the HDL description into an intermediate form are completed. Second, an optimization process of more vendor-specific technology mapping is conducted.

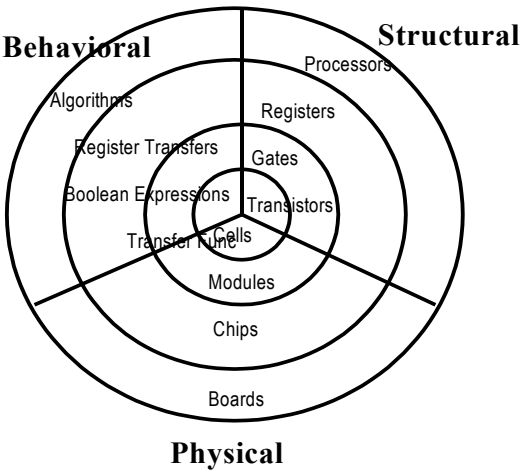


Figure 7. Different Design Views and their Level of Abstractions. (After Ref[14])

B. OVERVIEW OF VHDL CAPABILITIES AND ACTIVE VHDL

1. VHDL as a Programming Language

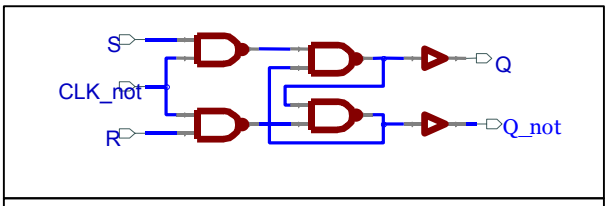
The primary hardware abstraction in VHDL is the design *entity*[1]. It represents a part of the design with well-defined inputs and outputs and performs a well-defined function. Each *entity* consists of two parts: its declaration and its *architecture*. The *entity* declaration defines the interfaces much like a software class declaration while the *architecture* body describes input-output transformations and/or the internal composition or behavior of the *entity* more like a software object. Interactions between concurrent statements are modeled through *signals.*

A *component* describes a substructure of the design entity that is interconnected through *signals.* Sequential statements such as loop and *case* statement are grouped together under the concurrent *process* statement. During execution all concurrent statements are executed during one simulation cycle and the values of all modeled signals are being computed. No VHDL model should depend on the order of execution of its concurrent statements.

When a signal takes on a new value, the sensitivity list of the concurrent statement decides if the statement is sensitive to that particular signal and acts accordingly. When all concurrent statements are suspended, simulation time advances.

The design and matching code in Figure 8 implements the behavior of the signals with logical statements on its signals. The same functionality could have been implemented in several different ways.

[1] Words in italic are protected VHDL constructs.




```
 library IEEE;
use IEEE.std_logic_1164.all;
entity My_flip_flop is
  port(
      CLK_not : in STD_LOGIC;
      R : in STD_LOGIC;
      S : in STD_LOGIC;
      Q : out STD_LOGIC;
      Q_not : out STD_LOGIC);
end My_flip_flop;
architecture My_flip_flop of My_flip_flop is
signal NET107 : STD_LOGIC;
signal NET124 : STD_LOGIC;
signal NET37 : STD_LOGIC;
signal NET41 : STD_LOGIC;
begin
NET41 <= not(CLK_not and S);
NET37 <= not(R and CLK_not);
NET124 <= not(NET107 and NET41);
NET107 <= not(NET37 and NET124);
Q_not <= NET107;
Q <= NET124;
end My_flip_flop;
```

Figure 8. Gate Level Design and Equivalent Code of RS Flip-Flop.

2. Active HDL

The tool chosen to perform the VHDL simulations was Active-HDL 5.1 developed by Aldec, Inc. of Henderson, NV. Active-HDL provides a number of features useful in the development as well as testing of hardware components. Its simulation technology features include compliance with IEEE VHDL1076-87/93 and IEEE Verilog1364-95. Furthermore, it supports EDIF 2.0.0 and Single or Mixed Language Configurations. The design flow manager of the language can be viewed in Figure 9.

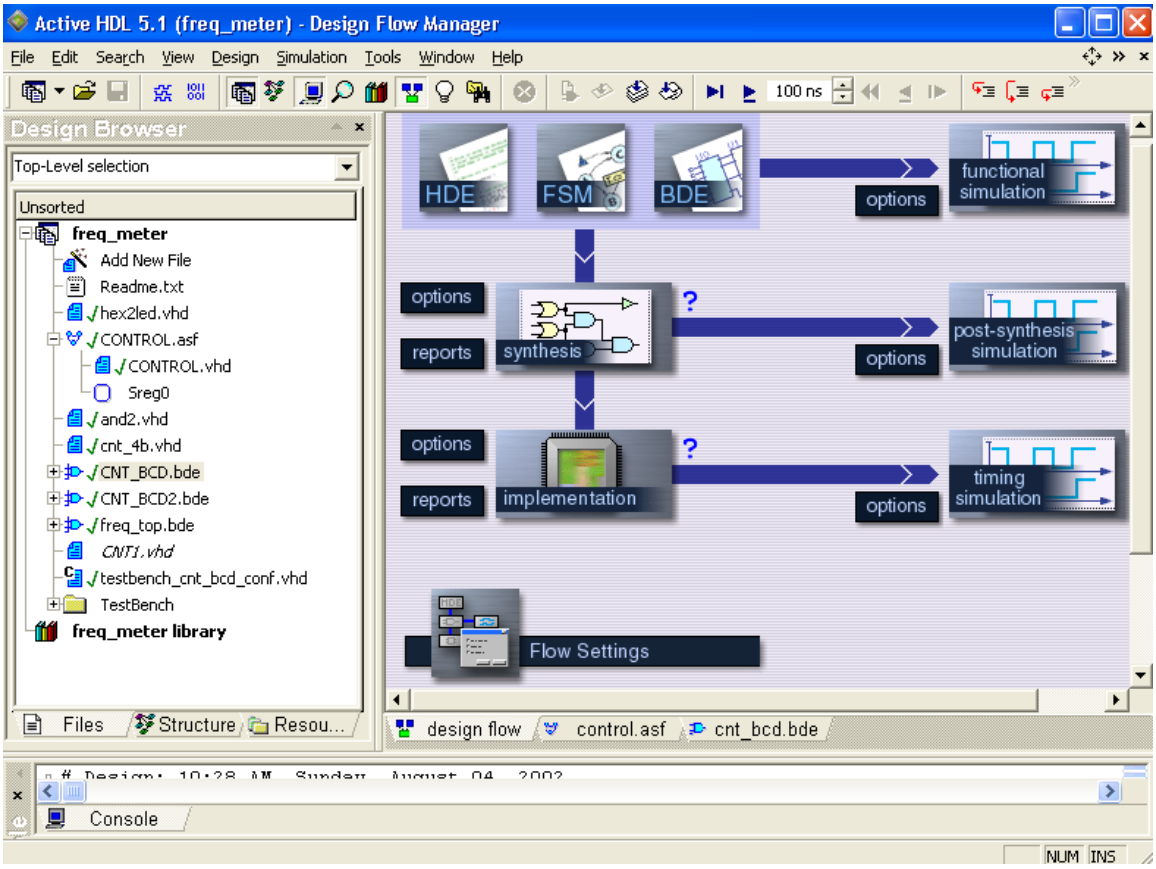


Figure 9. Design Flow Overview in Active-HDL 5.1.

Active-HDL provides the user the opportunity to create a design in three different ways:

- Through the HDL text editor , Figure 10, the user can build its model as with any other software language
- Through a Block Diagram Editor, Figure 11, graphical symbols of gates and combinatorial logic elements can be combined into larger entities
- Through the Finite State Machine Editor, Figure 12, the user can graphically enter a design based on state diagrams

The Active-HDL text editor resembles programming in, for instance, C or C++. This environment is tightly integrated with the compiler and simulator in order to provide debugging capabilities. Furthermore, the text editor provides, among other things, built-in language assistance, the capability of automatically generating design structures, setting and clearing of code breakpoints and cross probing of error messages. Active-

HDL has the ability to create block diagrams or finite state machines from the source code.

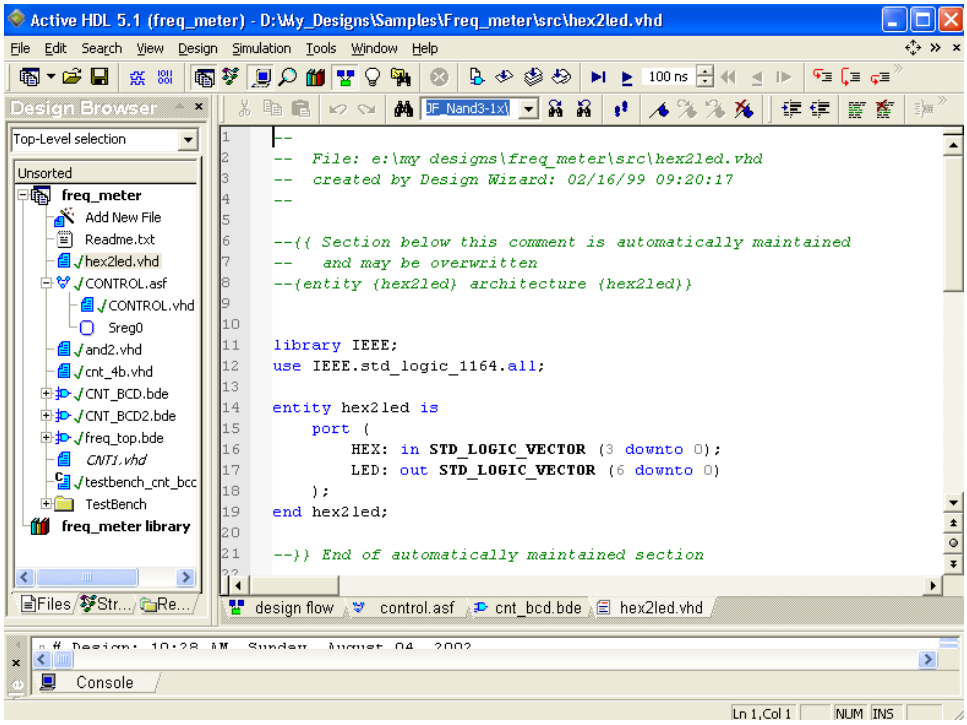


Figure 10. Text Editor in Active-HDL.

The Block Diagram Editor is a form of graphical description of a design entity in which each diagram has a counterpart in the VHDL source code. Active-HDL has a built in, vendor independent, symbol library with basic gates and combinatorial logic elements. Furthermore, Active HDL provides the designer with the ability to create their own combinatorial logic to save for reuse in subsequent applications. Other features of the Block Diagram Editor are the capabilities to import and export EDIF schematics as well as the feature of fast Design Rule Checking (DRC). The block diagram, when compiled, automatically generates source code that can be executed.

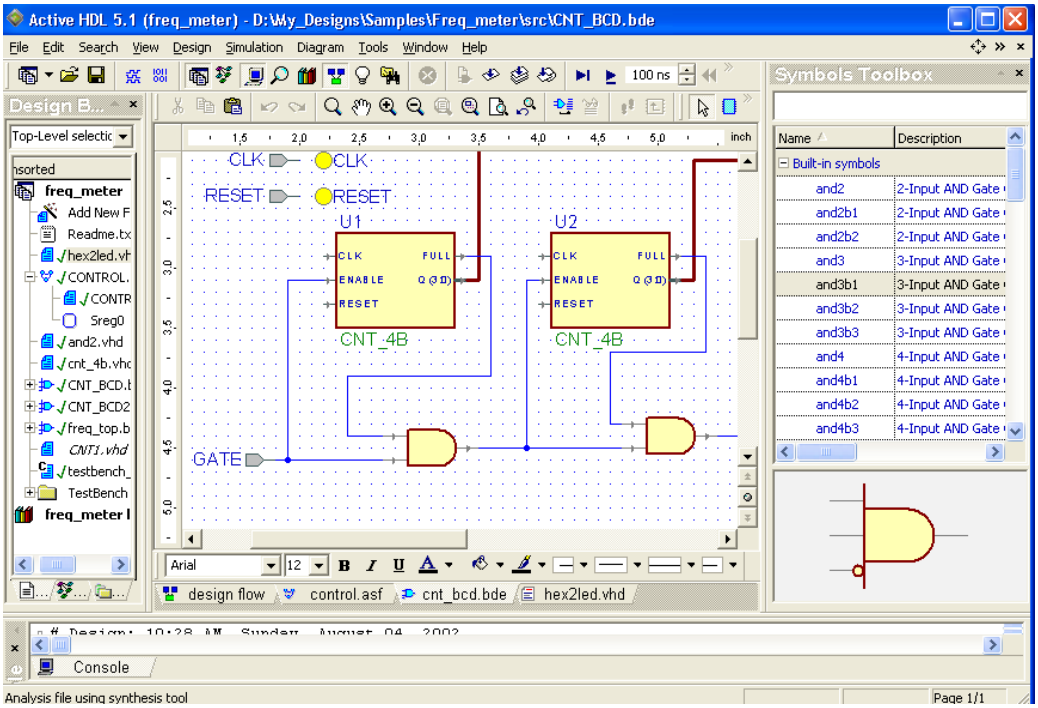


Figure 11. Block Diagram Editor in Active-HDL.

The Hierarchical Finite State Machine Editor allows the user to graphically enter a state diagram based design. State machines can then automatically be converted into HDL code for viewing and debugging.

In order to provide the capability to manufacture System-on-Chip (SoC) designs, Active-HDL offers a number of vendor specific libraries. It provides a seamless integration from design, through testing, to production when combined with the appropriate synthesizer.

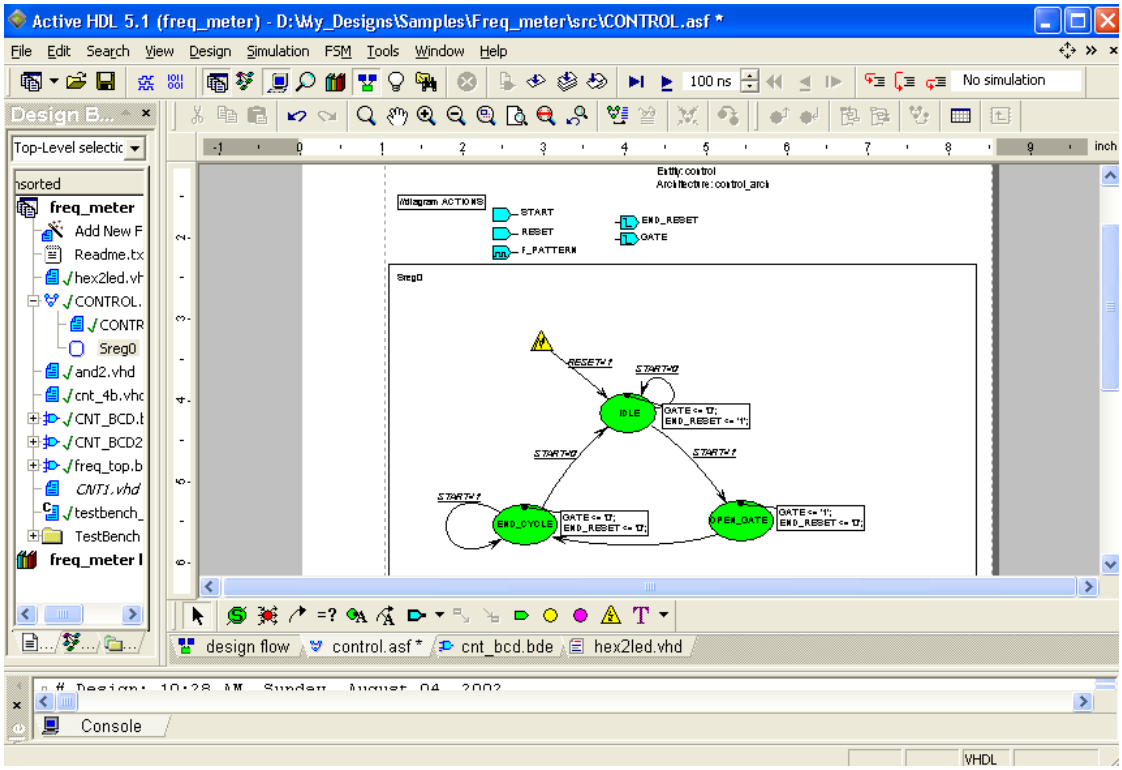


Figure 12. Hierarchical State Machine Editor in Active-HDL.

This thesis uses the capability of Active HDL to support automated software engineering. The ability to go from code to block diagram proved valuable as well as the capability of the application to support testing and verification. Active HDL handles test benches, coded and generated manually, as well as automatically generated test benches where saved wave forms are used.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SOFTWARE VERIFICATION METHODS

Different software methodologies can be used in order to remove the tedious verification of hardware designs. Previously, hardware was verified after a prototype was built. This was an expensive practice as changes were hard to implement and a new prototype might have had to be built instead. Similarly, a graphical hardware design languages normally is time consuming, compared to VHDL, when designs are simulated. VHDL only simulates "1", "0" , "Undefined" and "High Impedanse" while T-Spice is a circuit simulator and must keep track of all voltages, currents and charges on all wires. This section explores some software methods suitable to test and verify hardware design.

A. TESTING AND VERIFICATION

A test can be defined as an activity in which a system or component is executed under specified conditions and the results are observed and evaluated with respect to correctness. Verification is the process to ensure whether the component was built according to specifications. Testing is part of the verification process. In today's design efforts, testing and verification (TaV) needs to be planned early in the process. TaV is clearly a critical part of a project. Nowadays, huge efforts are undertaken to produce tools and methodologies in order to reduce overall verification time. (Ref.[5])

In its strictest interpretation, testing cannot take place until a prototype or a finished product is built. In this thesis, however, testing also refers to verification of hardware design using test vectors and a software test bench.

1. Reconvergence

Since the purpose of verification is to ensure that transformation generates the expected results, a second, reconvergent path with a common source is needed, see Figure 13.

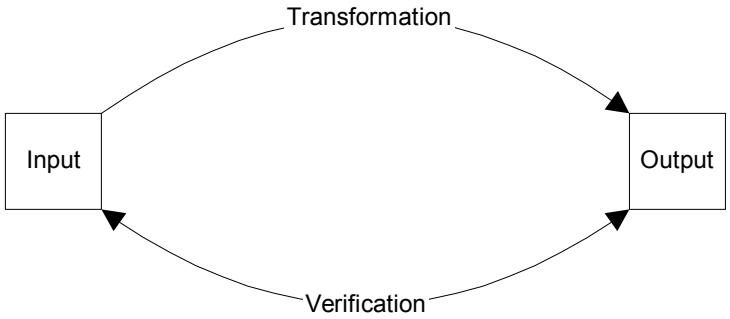


Figure 13. Transformation and Verification Flow. (From.[5])

Transformation can be any process that takes an input and produces an output. The verification process links the result with the starting point, making it possible for the verification effort to compare the actual output with the expected output.

One problem that arises in verification is the human factor. Figure 14 introduces specifications; misinterpreted they may introduce errors in the verification process.

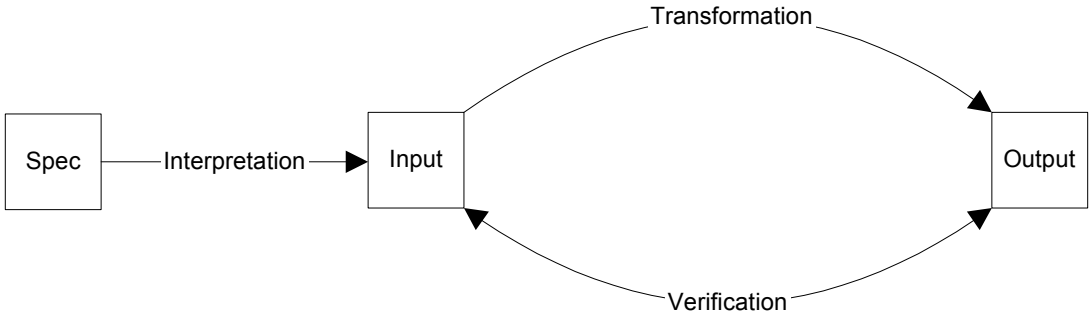


Figure 14. The Human Factor. (From.[5])

If the same team, or individual, who designed the entity is also involved in performing the verification process, obvious risks can arise causing the verification to be flawed. In that case, verification is that of the *interpretation* and not the *specifications.* If the interpretation is wrong in any way, so is the verification, and therefore, the error may never be caught with these verification efforts.

In order to prevent human interpretation errors, increased automation and redundancy can be used. Automation removes human intervention, but it is not always possible and it is seldom feasible in processes that are not well designed. Furthermore, there is no guarantee that the automation tool is flawless.

Redundancy is another way to reduce risk. It requires duplication of all transformation resources. Interpretations are performed independently and results are compared at a common output. Figure 15 shows how redundancy can be implemented and guarded against the misinterpretation of ambiguous specifications.

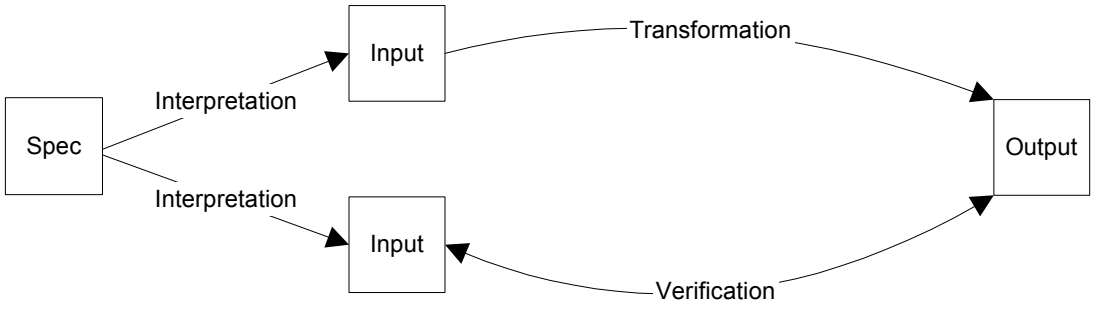


Figure 15. Redundancy (From.[5]).

B. FORMAL VERIFICATION

Different systems lend themselves to different types of verification. The following section will introduce some methods that might be used in the verification of an Integrated Circuit design represented in VHDL.

1. The Use of Logic

In order to achieve error-free Very Large Scale Integrated Circuit (VLSI) designs, different, complementing approaches to simulation and synthesis have been developed. One such attempt is to apply formal verification of the design's correctness. Formal verification, in this sense, is to verify the functionality correctness of the circuit.

There are a couple of inherent problems when deriving the formal verification however. First, conventional HDL languages lack the power for formal behavior descriptions. Second, a large gap exists between circuit descriptions and the

mathematical domain (Ref.[4]). In order to bridge these problems, most development environments use a HDL, such as VHDL or Verilog, or a subset of them, and implement some form of "formalized behavior" descriptions of the language.

The logic domain is the part of the mathematical domain most suitable to model the characteristics and properties of the applicable object. Logic, including first-order predicates, higher order predicates and temporal logic, is the overwhelming choice in performing formal verifications (Ref.[3]).

To exemplify formal verification, a simple adder is constructed.

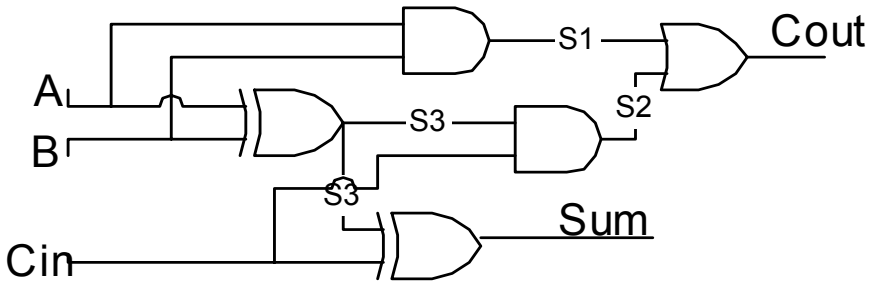


Figure 16. Logic Gate Representation of an Adder.

If the adder in Figure 16 is modeled in first order logic it might look like this:

For All t >= 0 =>
(S1(t)=A(t) AND B(t),
S3(t)=A(t) XOR B(t),
S2(t) = Cin(t) ANDS3(t),
Sum(t) = Cin(t) XOR S3(t),
Cout(t) = S1(t) OR S2(t))

Where A XOR B
Is equivalent to (A AND $\overline{B}$) OR ($\overline{A}$ AND B)
And
Cin XOR S3
Is equivalent to (Cin AND $\overline{S3}$) OR ($\overline{Cin}$ AND S3)

After removing S1 to S3 the expression will read:

For All t>= 0 =>
(Sum(t) = Cin(t) XOR (A(t) XOR B(t)),
Cout(t) = A(t) AND B(t) OR
(Cin(t) AND (A(t) XOR B(t)))

All that remains is to verify that the specification in the VHDL model corresponds to the logical model for all values of A, B and Cin.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity My_Full_Adder is
port(
A : in STD_LOGIC;
B : in STD_LOGIC;
Cin : in STD_LOGIC;
Cout : out STD_LOGIC;
Sum : out STD_LOGIC
);
end My_Full_Adder;
architecture My_Full_Adder of My_Full_Adder is
signal N3 : STD_LOGIC;
signal N2 : STD_LOGIC;
signal N1 : STD_LOGIC;
begin
N1 <= B and A;
N2 <= Cin and N3;
N3 <= B xor A;
Sum <= Cin xor N3;
Cout <= N2 or N1;
end My_Full_Adder;
```

Concentrating on the last section of the code, after the *begin* statement, the Sum was verified next. Sum = Cin XOR N3, and N3 in turn equals B XOR A, leading to Sum = Cin XOR(A XOR B)). This is the same expression in the logical description.

2. Binary Decision Diagrams and Computational Tree Logic

Binary Decision Diagrams (BDD) and Computational Tree Logic (CTL) are two other basic parts of formal verification.

a. BDD

BDD is a rooted directed acyclic graph with two terminal nodes: the 0-terminal and the 1-terminal. An ordered Binary Decision Diagram (OBDD) is a BDD in which the input variables appear in a fixed order on all the paths of the graph and no variable appears more than once in the path. A Reduced Order BDD (ROBDD) is an OBDD that results from the repeated application of the rules described in Figure 17 to Figure 19:

1. Remove duplicate terminals:

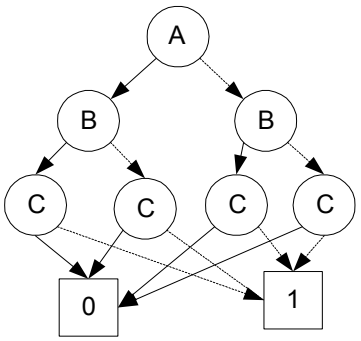


Figure 17. ROBDD Creation Step 1.

2. Condense duplicate nodes with identical parents and children:

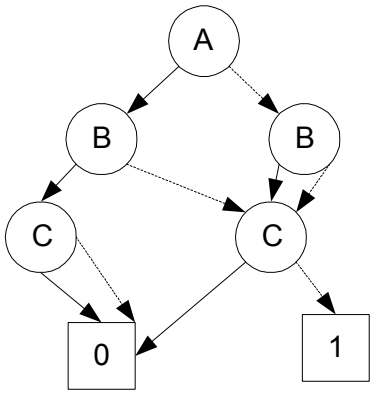


Figure 18. ROBDD Creation Step 2.

3. Remove redundant nodes:

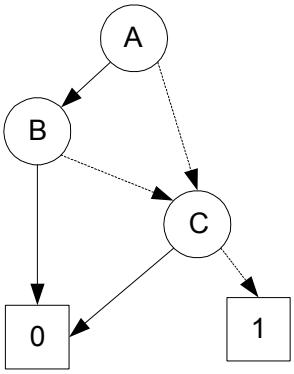


Figure 19. ROBDD Creation Step 3.

b. CTL

CTL adds path quantifier (**A, E**) and temporal operators (**X, G, F, U, W**) to first order logic. Temporal logic is used to express properties of possible simulations of a design. The path quantifier **A(E)** selects all (some) simulations, and the temporal operator **X (G, F, U, W)** selects the next simulation cycle (all cycles, some cycle, until some cycle, unless some cycle).

As an example, the expression:

AG(p=>A[p U q])

corresponds in plain English to: "From all cycles in which p holds, p always continues to hold until q holds".

3. Equivalence Checking

The simplest form of formal verification is proving the equivalence of two circuits. FORTE (Ref.[6]) allows the user to verify both combinatorial as well as sequential equivalence. Its purpose is to prove that two circuits produce the same output regardless of input.

An exhaustive application of all possible inputs and a comparison of all the outputs are involved for the combinatorial circuit. If the same output is produced, the circuits are functionally equivalent.

When the circuits possess different state properties, sequential equivalence must be checked instead. To be equivalent, the circuits must start in some initial state and have identical outputs and transitions for all possible sequences of inputs. Equivalence checking is only interested in comparing boolean and sequential logic functions. The functions to a specific technology are not mapped.

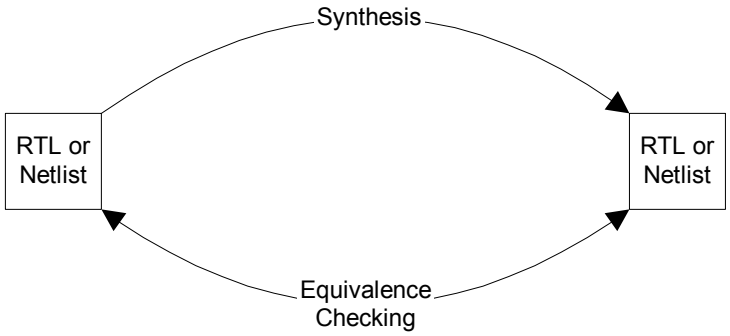


Figure 20. Equivalence Checking Paths. (From.[5])

The most common use of equivalence checking is the comparison of netlists, shown in Figure 20. This ensures correctness of the synthesis tool and that manual modifications implemented during netlist post-processing did not change any functionality.

4. Model Checking

Model checking is a relatively recent application of formal verification. As seen in Figure 21, it attempts to prove or disprove certain design assertions or characteristics.

Model checking seems to be the most investigated approach used for the verification of a hardware chip. Siemens Circuit Verification Environment (CVE) is one approach described by Borman et al. (Ref.[7]). CVE is a BDD based model checker that supports VHDL and Electronic Design Interchange Format (EDIF). It generates VHDL test benches for proposed counterexamples if it detects a design error. CVE is operated from a menu driven graphical user interface (GUI).

The designer has to specify the properties to be model checked and must then consider entire sets of behavior. Two features help the designer. One feature is CVE's Interval Language (CIL) which is an extension of Boolean VHDL expressions. The second is an algorithm that automatically generates a special finite state machine (FSM) representation for synchronous circuits.

Another model checker supporting VHDL is CV (Ref.[9]). CV essentially uses logic CTL as its specification language. The VHDL description is compiled into a state-

transition graph represented internally by BDDs. Then model checking techniques are used to determine if the VHDL specification holds. CV performs computation of reachable states and eliminates unreachable states from the simulation. Further it implements a boolean functional vector to limit the explosion of the size of the transitions in larger systems.

A third example of a model checker implementing VHDL is RuleBase. (Ref.[9]) RuleBase is a formal verification tool developed by IBM Haifa Research Laboratory. It is an enhanced version of SMV developed by Ken McMillan at Carnegie-Mellon University. As in the previous example, it uses the CTL model checking verification method through its own language called Sugar. Sugar is built on top of CTL and provides a method for hardware designers, who are not CTL experts, to read and write specifications.

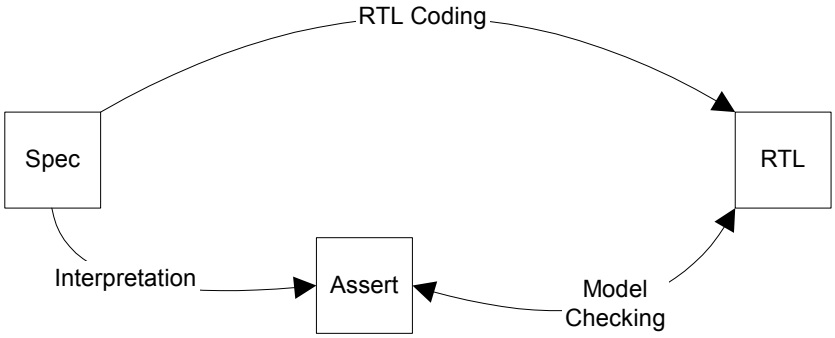


Figure 21. Model Checking Paths. (From.[5])

The greatest difficulty in applying model checking is to identify which assertions to prove. Of the identified assertions, only a subset can feasibly be proven.

5. Theorem Proving

PREVAIL (Ref. [10]) is a menu driven, automatic proof environment that verifies certain categories of synchronous sequential circuits. A VHDL subset is taken as input as well as a description style associated with formal semantics. The tool operates in two steps:

- After compilation of the VHDL code, PREVAIL inputs an intermediate form of the entity and architecture description. It then builds a corresponding, proof-oriented, functional circuit representation.

- The second part uses a query-answer dialogue with the designer in order to determine circuit type and selects between a tautology checker (checking for redundant repetitions) and the Boyer-Moore theorem-prover.

The Boyer-Moore theorem-prover allows the inductive definition of abstract data types called shells. A Boolean recognizer recognizes whether an object belongs to the shell. Furthermore, the definition of recursive functions are allowed and a robust verification by the system of the correctness of the recursive form is performed. Next, the inductive theorems expressing properties of these recursive functions are proven. To prove a property by induction, the prover automatically generates an induction scheme according to the definition of the recursive function involved in the property.

6. Functional Verification

The purpose of a functional verification is to ensure that a design implements intended functionality. Functional verification compares the design specification to a measured result. It must, however, be noted that unless the specification is written in a formal language, it is impossible to prove that the intended specifications are met. Functional verification can show that the intent of the specification is met but it can hardly prove that the functionality is faultless.

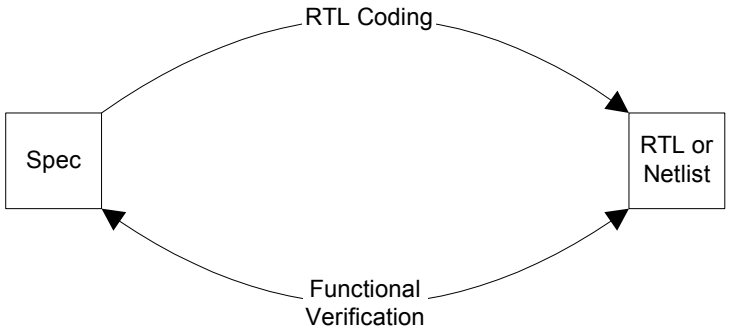


Figure 22. Functional Verification Paths. (From.[5])

Functional verification as depicted in Figure 22 can be implemented through different approaches and methodologies. The approaches, black-box, white-box, and

grey-box combined with bottom-up or top-down methodologies, constitute the cornerstones of functional verification.

a. Black-Box Verification

The black box verification, Figure 23, implies no knowledge of the internal implementation of a particular design. Verification takes place through the interpretation of the output from a specific input.

Figure 23. Outline of Black-Box Verification.

The difficulties with black-box verification is its lack of controllability. It is a challenge to design a certain state combination or to isolate a specific function. This leads to difficulties in determining the source and location of potential problems as well as its occurrence in time inside the black-box.

The main advantage of black-box verification is its independence of a specific implementation. black-box verification can be used on hardware in the form of ASIC chips or FPGAs as well as a design represented in software. Another advantage is the ability to construct functional verification in parallel with the development of the implementation itself.

b. White-Box Verification

White-box verification, Figure 24, sometimes named clear-box or glass-box, provides full visibility of the internal mechanisms of the implementation.

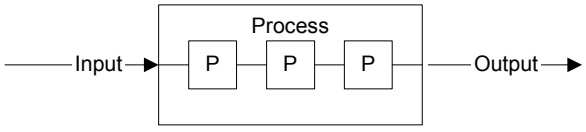


Figure 24. Outline of White-Box Verification.

The advantages of having control over the internal structure of the implementation are obvious. Interesting combinations of inputs can be designed to trigger particular functions. The result of the input can be followed through the design and errors can be captured where they occur. White-box verification is especially useful in order to check the functionality of counters or overflow guards.

The drawbacks of the approach are the symbioses between the test and its host implementation. It cannot be used in the same format on other implementations. It also requires detailed knowledge of the Unit Under Test (UUT) in order to know which conditions to create and which results to expect.

c. Grey-Box Verification

The compromise between the two is the grey-box, or opaque, verification seen in Figure 25. The verification efforts benefit from the knowledge of the internal structure of the implementation while treating it as a back-box.

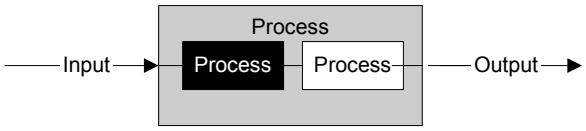


Figure 25. Outline of Grey-Box Verification.

As with the black-box approach, the top level interfaces are used to trigger and control the verification efforts. Test cases may or may not be useful on other implementations.

C. SIMULATION

Most implementations of testing and verification involve some form of simulation activity. In VHDL, simulation refers to the implementation of a discrete event and is normally conducted through the implementation of the box-approaches from the previous section. The discrete event simulator executes the VHDL code, modeling the passage of time and the occurrence of events at certain times or after certain delays. Discrete event simulations utilize an event list data structure that maintains an ordered list of all future

events in the circuit. Each event is described by its type, for instance, a transition from 1 to 0 and the time when it occurs. The event simulator works in the following steps:

- Advance the simulation time to the event which has the smallest timestamp
- Execute all events at this timestep by updating their signal values
- Execute the simulation models of all components affected by the new values
- Schedule future events
- Repeat until event list is empty or time has expired

The result of the simulation is normally presented in a waveform window. However, signals can also be followed through a block diagram.

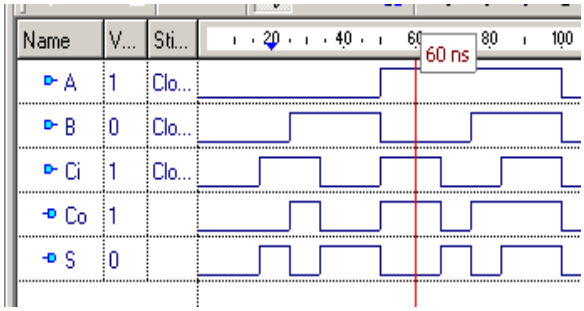


Figure 26. Simulation Result of a Simple 1-Bit Adder.

Such a simulation requires knowledge of the expected output at the abstraction level being simulated. The simulated result is then compared to the expected result which is often derived mathematically. One example of a sesult is that of the 1-bit adder in Figure 26.

One obvious drawback of this verification type is the state explosion that occurs when large, complex systems are constructed. To counteract the increase in size of the test design, a hybrid test design, i.e., the gray box, approach can be implemented. The expected output from a given input in a black box approach has to be combined with the tests of the internal composition of the system.

D. CHOSEN METHODOLOGY

The verification of the DIS was conducted through a white-box, functional verification approach and implemented through discrete event simulation.

The different areas of the design were verified stepwise and tested independently. Smaller parts were integrated into larger parts, and thereby, increasing the scope which lead to an overall bottom-up integration. After verification of the smaller components, the partially automatically generated single range-bin and 8 Range-bin implementations were tested. The final step consisted of designing and verifying the logical functions of a VHDL coded 32 Range-bin implementation.

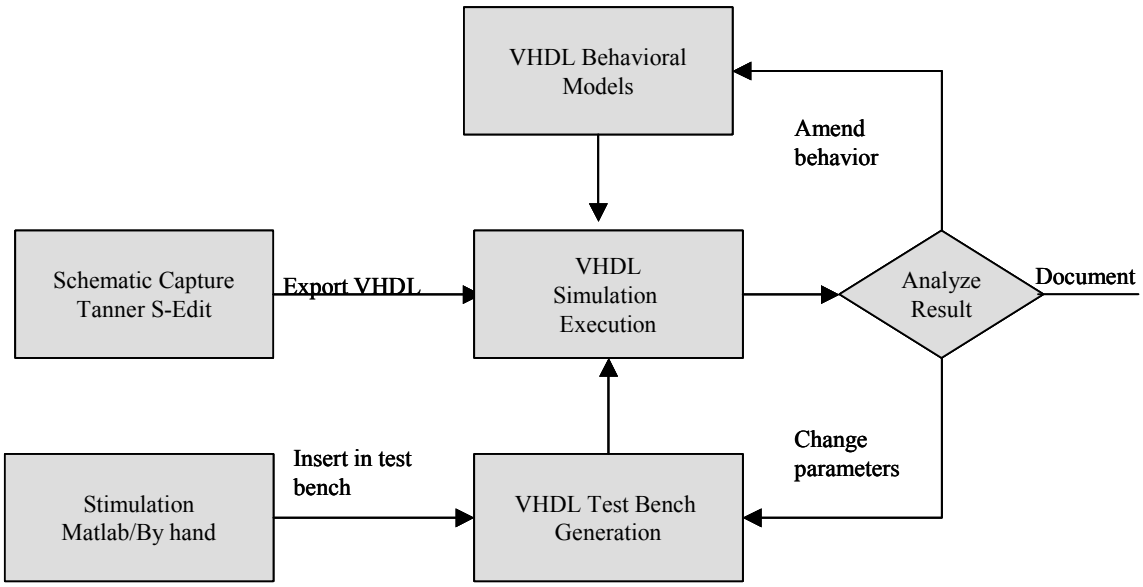


Figure 27. Verification Process Flow.

This approach verifies the interfaces between the individual system-components constituting the whole system. It is suitable for a system with stable interfaces and components developed by different teams, as is the case with the 8 Range-bin implementation which has parts designed by five different researchers. It is equally useful when testing larger entities of software created modules later. The process implemented on each level is described in Figure 27.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. VERIFICATION OF HARDWARE DESIGNS

Verifying the ability of the VHDL tool to import and simulate the Tanner S-Edit developed schematics was one of the purposes of this thesis and the subsequent supporting research. If successful, a considerable amount of time would be saved due to a decrease in simulation time by several orders of magnitude compared to T-Spice circuit simulations. After experimenting with gate level as well as transistor level implementations, the decision was made to export as complete a model as possible. As a result, the smallest entities of Ground, Power, NFETs and PFETs became the building blocks of the DIS.

A. VHDL CODE EXTRACTION

A considerable amount of time during the research process dealt with the process of exporting the graphic S-Edit design to VHDL code. Initially, it was possible to export a VHDL text file and open it with Active-HDL. However, once the files were opened they did not work as expected. It was discovered that certain bi-directional ports had to be directed in order to run the model in VHDL. Furthermore, libraries had to be initialized for each entity of the design. The entities can be viewed as the scope in which variables operate.

1. Extraction Guidelines

The following steps had to be undertaken in order to export VHDL files from Tanner S-Edit schematic diagrams:

- Ensure no modules are defined as VHDL primitives except for NFETs, PFETs and the global power and ground symbols
- For NFETs, PFETs and the global power and ground symbols, declare them as VHDL primitives by adding a property to the symbol (not the schematic) version of the module. The property should say: [VHDL PRIMITIVE=]

  When creating the property, the value field should be “blank”, the separator character should be “=“, the text size should be “2”, the value type should be “Text”, and show should be set to “none”.
- All ports must be defined as in or out, even for the global power and ground modules. For the global power and ground modules, make the ports out. For FETs, make source, gate, and bulk in and make drain out.

On the symbol version of NFETs and PFETs, it is best to indicate what is the source and what is the drain so when these are instanced at a higher level, the ports can be connected up correctly.

- Pass gates must be made unidirectional and use unidirectional (in or out) ports. It is best to supply the symbol of the pass gate with an indication of the direction so that when the symbol is instanced at a higher level, the ports will be connected up correctly.
- There must not be any networks in any module connecting to output ports and at the same time connecting to the inputs of other logic gates in the module. If such a network exists, a non-inverting buffer (2 inverters in series) must be inserted before the output port.
- Delay buffers must be inserted on the outputs of signals that control flip-flops. The VHDL programmer will adjust the amount of delay.

B. VHDL CODE MODIFICATION

The modification of the code depends on the level at which the schematic designer exports the VHDL file. VHDL is used to create the behavior of the designed entities. The behavior can be instantiated at the transistor-level, gate-level or at the combinatorial level. Given that the guidelines for code extraction are followed, the only modifications apart from behavioral instantiation, is compliance with VHDL naming rules and entity declarations.

1. Naming Conventions

Identifiers are used both as names for VHDL objects, procedures, functions, processes and design entities, and as reserved words. There are two classes of identifiers: basic identifiers and extended identifiers.

The *basic identifiers* are used for naming all named entities in VHDL. They can be of any length provided that the entire identifier is written in one line of code. Reserved words cannot be used as basic identifiers. Underscores are significant characters in an identifier and basic identifiers may contain underscores, but using an underscore as a first or last character of an identifier is not allowed. It was discovered that dashes, -, could not be used in a basic identifier early in the research effort. The rules for the basic identifiers in are:

- A basic identifier must begin with a letter
- No spaces are allowed in basic identifiers

- Basic identifiers are not case sensitive, i.e. upper- and lower-case letters are considered identical
- Basic identifiers consist of Latin letters (a..z), underscores (_) and digits (0..9)

The *extended identifiers* were included in VHDL '93 in order to make the code more compatible with tools making use of extended identifiers. The extended identifiers are braced between two backslash characters. They may contain any graphic character as well as reserved words. If a backslash is to be used as one of the graphic characters of an extended literal, it must be doubled. Upper- and lower-case letters are distinguished in extended literals.

2. Entity Declaration

For each entity declared in a design, whether it is in the same file or not, libraries need to be added. The most common declaration and the only one needed in this thesis was the following:

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
```

Std_logic_1164.all uses more memory than, for instance, a bit library. For the purpose of this research, it was never a concern.

3. Behavior

Although the designs tested are mainly of a structural nature, behavior needs to be defined for certain entities. To drive the DIS Designs, Ground, Power, NFETs and PFETs needed to be assigned a behavior.

C. CREATION OF MODELS

After the specific circuit was exported, a VHDL design representing that design was created. See the tutorial in Appendix A.

1. Inverter

The first successful exported Spice design was that of a simple inverter. Using the building blocks in Figure 28; Ground, Power, NFET and PFET a shell of the inverter was created, Figure 29.

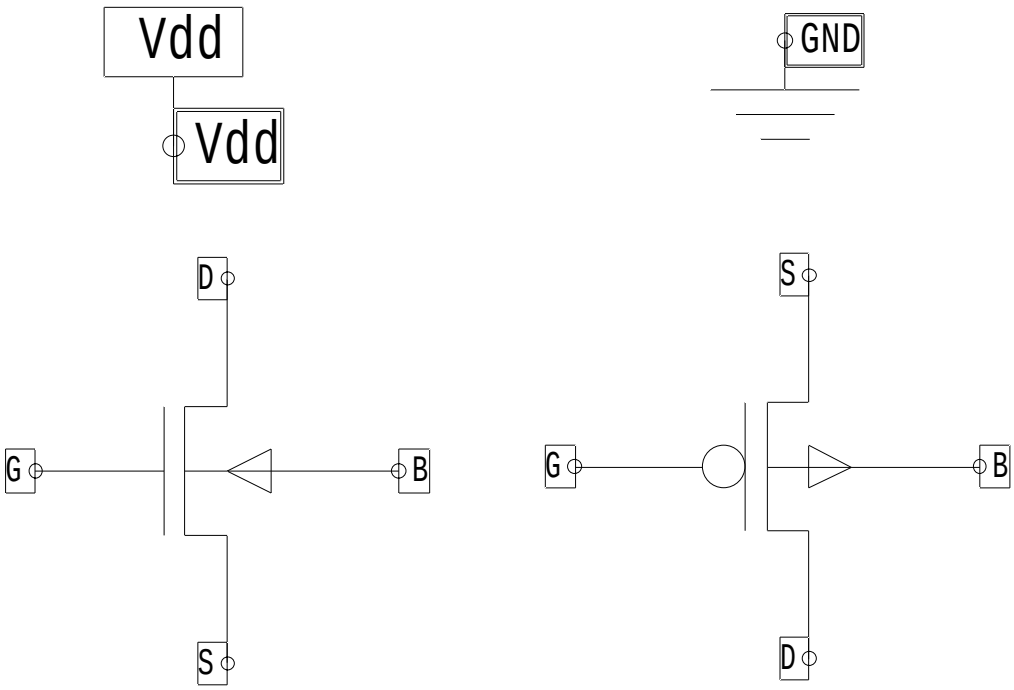


Figure 28. The Primitive Symbols of Power, Ground NFET and PFET.

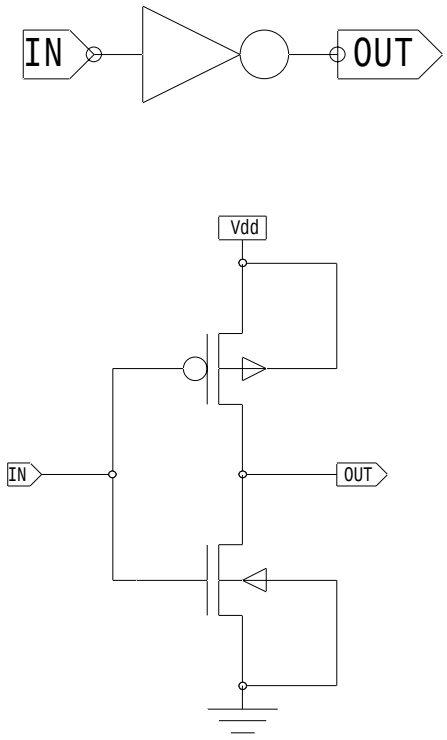


Figure 29. The Inverter Logic Gate.

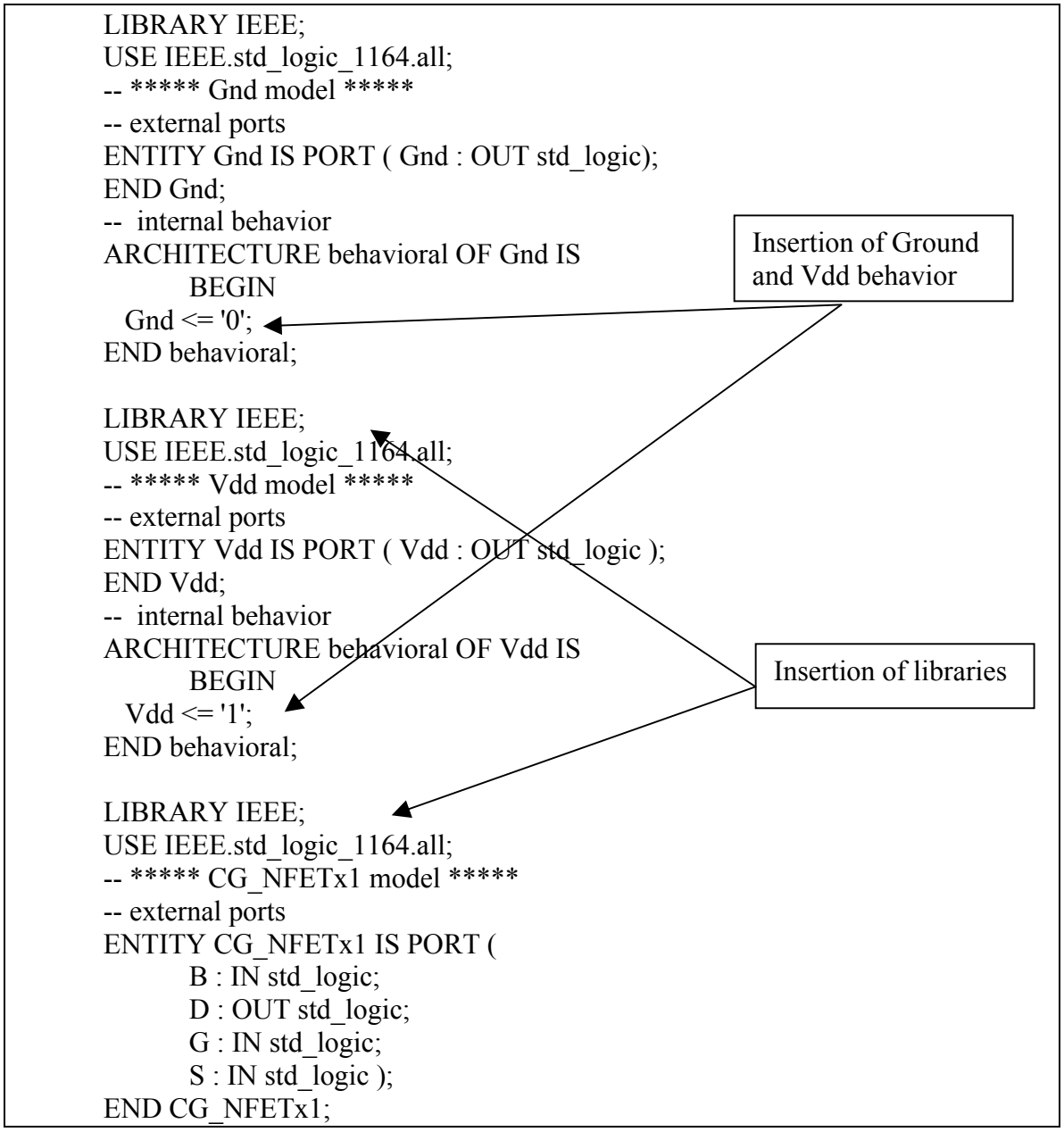
If implemented at the gate level, a behavioral model of the inverter in VHDL might have been implemented as in Figure 30: one line transferring a negated input signal to the output.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
-- ***** DJF_Inv-1x model *****
-- external ports
ENTITY \DJF_Inv_1x\ IS PORT (
        \In\ : IN std_logic;
        \Out\ : OUT std_logic );
END \DJF_Inv_1x\;

--  internal behavior
ARCHITECTURE behavioral OF \DJF_Inv_1x\ IS
begin
  \Out\ <= not(\In\);
END behavioral;
```

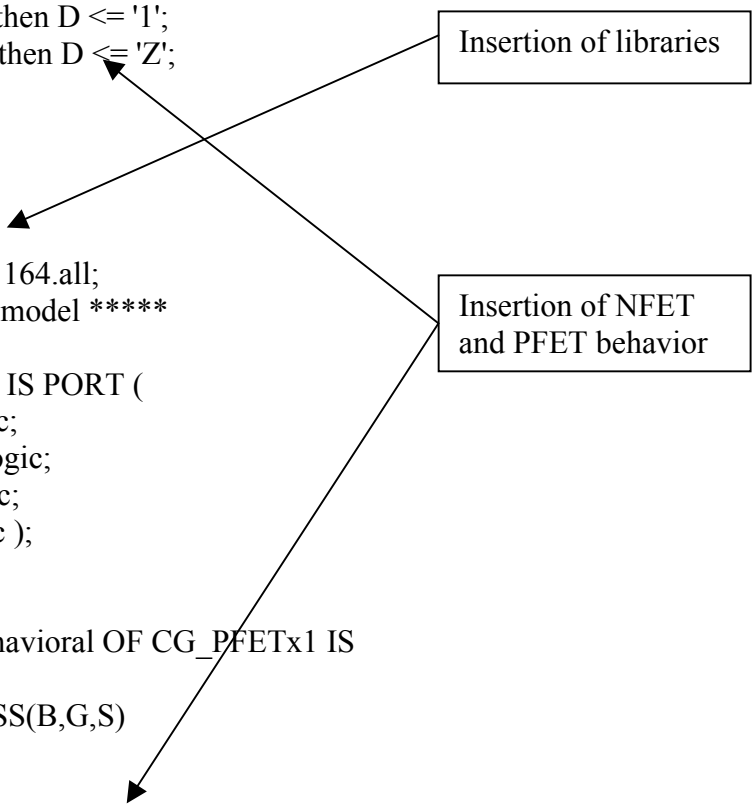
Figure 30. Description of Inverter in Behavioral VHDL.

If the inverter is allowed to serve as an example for the modifications necessary in VHDL in order to successfully simulate an implementation, Figure 31 shows necessary changes and additions to the code.



```
--  internal behavior
ARCHITECTURE behavioral OF CG_NFETx1 IS
BEGIN
        NFET:PROCESS(B,G,S)
BEGIN
if G ='0' then D<='Z';
elsif (G='1' and S='0') then D <= '0';
elsif (G='1' and S='1') then D <= '1';
elsif (G='1' and S='Z') then D <= 'Z';
end if;
end process NFET;
END behavioral;

LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
-- ***** CG_PFETx1 model *****
-- external ports
ENTITY CG_PFETx1 IS PORT (
        B : IN std_logic;
        D : OUT std_logic;
        G : IN std_logic;
        S : IN std_logic );
END CG_PFETx1;
--  internal behavior
ARCHITECTURE behavioral OF CG_PFETx1 IS
BEGIN
        PFET:PROCESS(B,G,S)
BEGIN
if G ='1' then D<='Z';
elsif (G='0' and S='0') then D <= '0';
elsif (G='0' and S='1') then D <= '1';
elsif (G='0' and S='Z') then D <= 'Z';
end if;
end process PFET;
END behavioral;
```



```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
-- *****CG_Inv_1x model *****
-- external ports
ENTITY CG_Inv_1x IS PORT (
        \In\ : IN std_logic;
        \Out\ : OUT std_logic );
END CG_Inv_1x;
-- internal structure
ARCHITECTURE structural OF CG_Inv_1x IS
-- COMPONENTS
COMPONENT Gnd
PORT (
        Gnd : OUT std_logic );
END COMPONENT;
COMPONENT CG_NFETx1
PORT (
        B : IN std_logic;
        D : OUT std_logic;
        G : IN std_logic;
        S : IN std_logic );
END COMPONENT;
COMPONENT CG_PFETx1
PORT (
        B : IN std_logic;
        D : OUT std_logic;
        G : IN std_logic;
        S : IN std_logic );
END COMPONENT;
COMPONENT Vdd
PORT (
        Vdd : OUT std_logic );
END COMPONENT;
```

Insertion of libraries

```
-- SIGNALS
SIGNAL LogVdd : std_logic;
SIGNAL LogGnd : std_logic;
-- INSTANCES
BEGIN
Gnd_1 : Gnd   PORT MAP(
        Gnd => LogGnd);
NFET_1 : CG_NFETx1   PORT MAP(
        B => LogGnd,
        D => \Out\,
        G => \In\,
        S => LogGnd);
PFET_1 : CG_PFETx1   PORT MAP(
        B => LogVdd,
        D => \Out\,
        G => \In\,
        S => LogVdd);
Vdd_1 : Vdd   PORT MAP(
        Vdd => LogVdd);
END structural;
```

Renaming of signals to avoid global conflicts.

Figure 31. Description of an Inverter in Structural VHDL.

Functional verification of the inverter was easily conducted by running a simple stimulus on the in-port and verifying that the output signal, as in Figure 32, was the opposite.

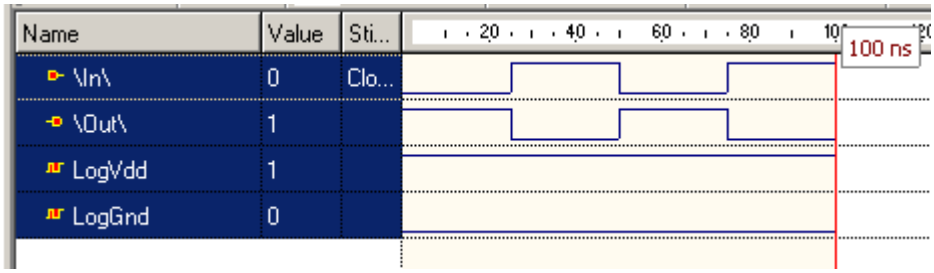


Figure 32. Screen Capture of the Waveform Window for an Inverter.

Implementations of NAND gates with two to five input ports and a two input NOR gate were verified in a similar manner.

2. Subsequent Models

After gate level verification, type specific components of the DIS were tested. These components were the 1-bit adder, the 5-bit adder, the 16-bit adder, the 4-bit register, the sine-cosine look up table and the Pass Gate. The Block Diagram, Figure 33, and Waveform, Figure 34, were used in order to be able to follow and verify the correctness of the signals. A short time increment comparing the drivers allowed incremented inputs. For the VHDL source code for the 1-bit adder, see Appendix A.

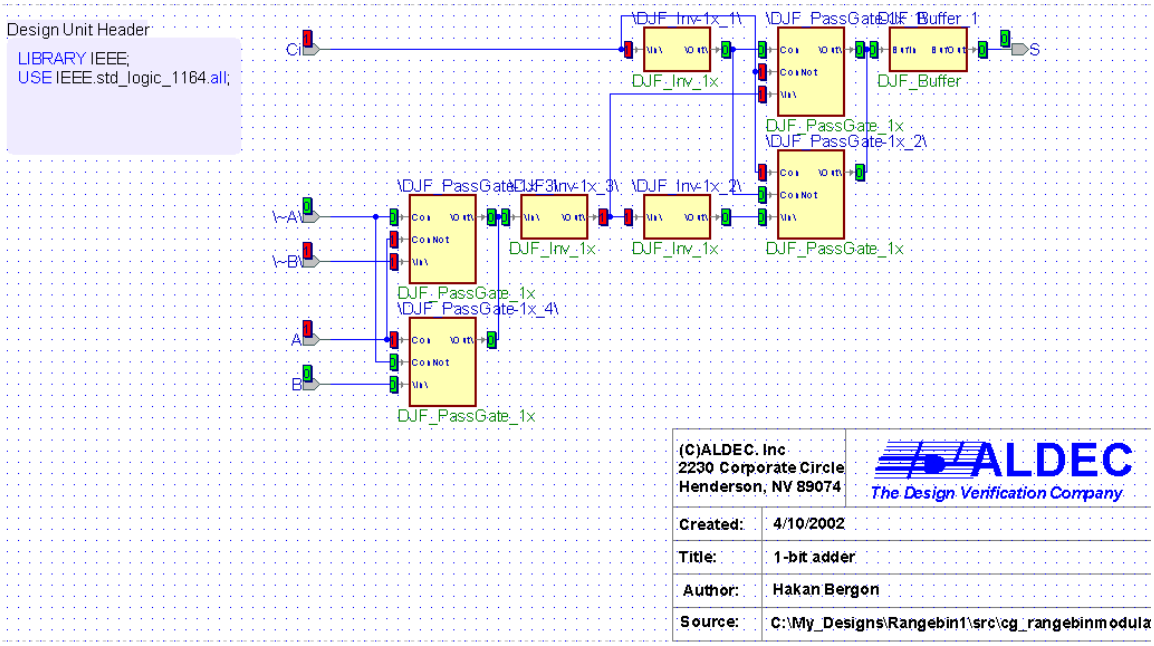


Figure 33. Block Diagram of a 1-Bit Adder without Carry Out.

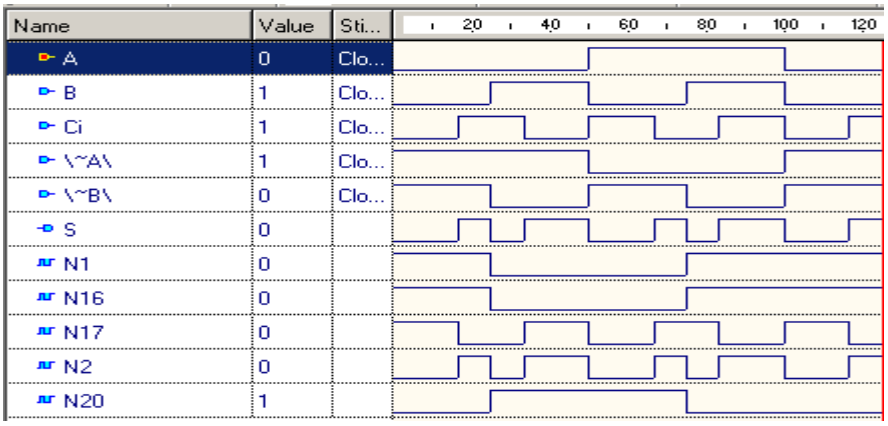


Figure 34. The 1-Bit Adder in the Waveform Window.

D. VERIFICATION OF SINGLE RANGE BIN MODULATOR

The first complete circuit layout verified was that of a Single Range Bin Modulator designed by Major Christian Guillaume (Ref. [2]). Verification was obtained by testing 31 vectors and comparing them to hand calculated, Matlab calculated and Spice simulated results.

In order to verify the basic function of the Range Bin Modulator, vectors (DRFM phase values) are applied to the modulator together with different values of phase increment and gain. Initially, the I and Q inputs, i.e., the values from a previous Range Bin Modulator, are set to zero.

Control signals are tested and their effects on the output data from the Range Bin Modulator verified.

When VHDL was used, simulation time for the single Range Bin was a matter of a few seconds. Tanner Tools Pro, on the other hand, needed approximately 30 minutes to conduct the same simulation.

1. Underlying Mathematics

The output result for each range Bin can be calculated mathematically and then added to the result of a potential previous Range Bin.

Initially, an unsigned 5-bit representation of the phase of the signal leads to increments of 11.25º when the 360º of possible phase is divided by 32. The same is true for the incremented phase coefficient.

The gain is implemented through a gain shifter, essentially using a 4-bit control code to apply the gain multiple.

Table 1 shows the truth table for the gain shifter. The "Control code" corresponds to the decimal value of the unsigned 4-bit word applied to the gain input (Gain0 - Gain3). The "Multiplication factor" is the effective decimal value by which the input of the gain shifter is multiplied. The "Size of shift" gives the corresponding number of bits to the left by which the input is shifted. The "Sin/Cos wave resolution" gives the resolution in bits of the I and Q signals at the output of the gain shifter (Ref.[2]).

Control Code	Multiplication Factor	Size of Shift	Sin/Cos Wave Resolution
0	1	0	3
1	2	1	4
2	4	2	5
3	8	3	6
4	8	3	6
5	16	4	7
6	32	5	8
7	64	6	8
8	16	4	7
9	32	5	8
10	64	6	8
11	128	7	8
12	128	7	8
13	256	8	8
14	512	9	8
15	1024	10	8

Table 1. The Capabilities of the Gain Multiplier.

The output from the Single Range Bin Layout can be mathematically calculated as follows as an example:

$$I_{Out} = GAIN * Cos(DRFM + INC)$$

$$Q_{Out} = GAIN * Sin(DRFM + INC)$$

A quantized DRFM phase of 5 corresponds to 56.25º, a Phase Increment of 1 corresponds to 11.25º and a Gain code of 6 equals a multiplication by 32.

The following results can be calculated:

$$I_{Out} = 32 * Cos(56.25 + 11.25) = 12.25$$

$$Q_{Out} = 32 * Sin(56.25 + 11.25) = 29.56$$

These results are well in unison with the results for test vector five in the test result table following later.

The I and Q values are stored in a 16-bit, two's complement format, with a decimal point two positions in. Thus, the I and Q values range from approximately + 8200 to – 8200.

E. LAYOUT

The schematic design of the Single Rang Bin, Figure 35, is as follows:

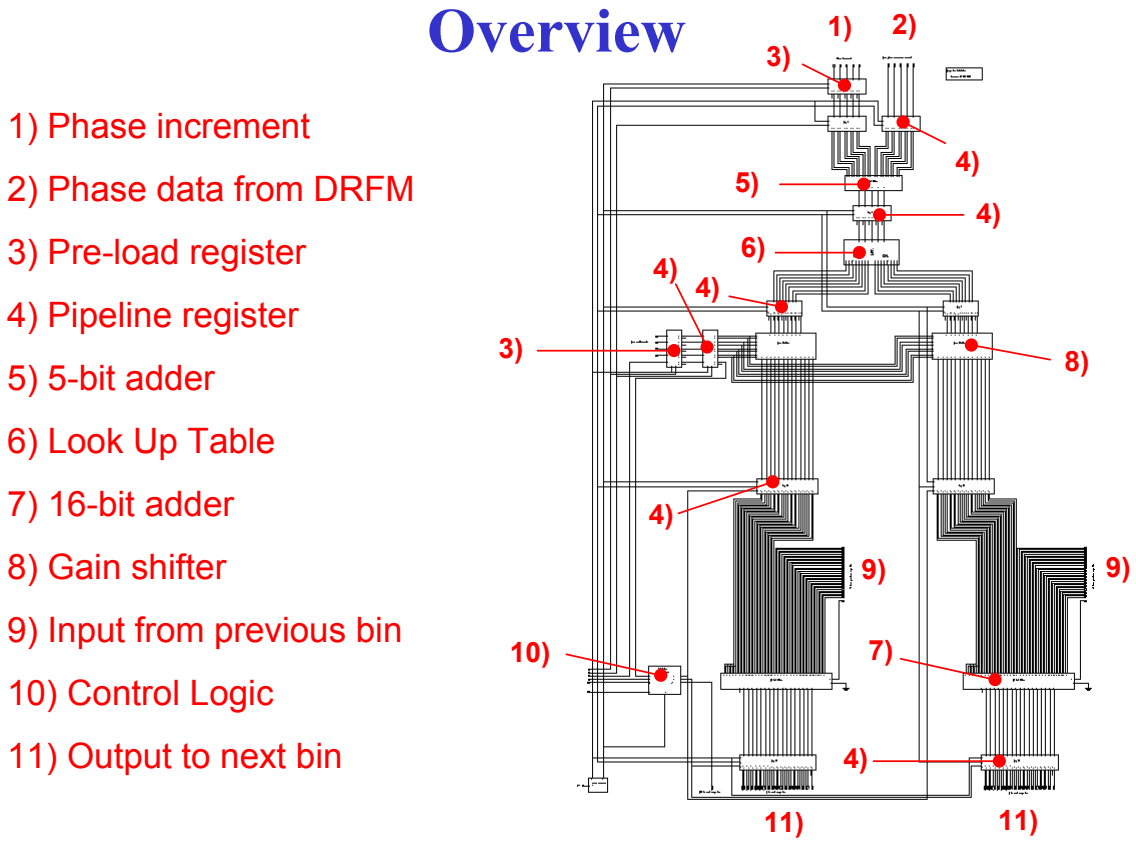


Figure 35. Overview of the Range Bin Modulator Schematic (From.[2]).

F. CONTROL SIGNALS

A number of control signals and phase signals have to be instantiated with certain values at certain times in order to drive the simulation. The functions of these signals are as follows:

Use Range Bin (URB): when a logic one is present on the rising clock edge, the selected Range Bin Modulator operates normally. The result of the modulator is added to the result of the modulator of the immediate preceding range bin, and the sum is provided

to the next Range Bin Modulator in the chain. When a logic zero is present on the rising clock edge, the current modulator is not used for computation, and its output reflects only the value of the previous modulator in the chain, assuming that one is in use.

Phase Sample Valid (PSV): when a logic one is present on the rising clock edge, which is the normal case, the current modulator computes output values given its programming and the present DRFM phase. The result is added to the result of the previous modulator. When a logic zero is present on the rising clock edge, the result of the current modulator is forced to zero, and only the result of the previous modulator is applied to the output of the current modulator.

Output Data Valid In (ODVin): when a logic one is present on the rising clock edge, the Range Bin Modulator works normally. When a logic zero is applied, two outcomes may arise depending on the result of the current modulator computation. In the first case, if the result of the current modulator is valid, then it will be added to the result of the previous Range Bin Modulator and provided to the next one in the chain. In the second case, if the result of the current modulator is invalid, then the output register is forced to zero, and the "Output Data Valid Out" (ODVout) is set to zero as well.

Program Range Bin (PRB): a logic one makes the pre-load buffer registers on the gain inputs and the phase rotation inputs load the data.

Use New Programming (UNP): a logic one makes the gain and phase rotation registers load the data in the pre-load buffer registers.

Gain (Gain0-Gain3): an unsigned 4-bit representation of the control code used in order to implement the correct multiplication factor.

Phase Increment (Inc0-Inc4): an unsigned 5-bit representation of the 32 different pre set phase increments.

DRFM (DRFM0-DRFM4): an unsigned 5-bit representation of the 32 different signal phase values.

G. DRIVER INPUT METHODOLOGY AND EXPECTED OUTPUT

The design of the Range Bin Modulator operates on the rising edge of a clock cycle. The following algorithm was developed in order to ensure a correct sequence of initialization of registers.

H. TEST ALGORITHM:

At same time:

Set Phase Inc to desired value for Rbi

Set Gain to desired value for Rbi

Set Use Range Bin to "1"

Set Phase Sample Valid to "0"

Set Operate/Main to "1"

Set Program Range Bin to "1"

Clock, rising edge = 1 ms at 500MHz

At same time:

Set Use New Programming ="1"

Set Phase Sample Valid ="1"

Set Program Range Bin to "0"

All else don't care

Clock, rising edge = 3 ms at 500MHz

At same time:

Set Use New Programming ="0"

Set Use Range Bin =“0”

Set phase sample from DRFM to desired value

All else don’t care

Clock, rising edge = 5ms at 500MHz

At same time:

Change phase sample from DRFM all other signal the same

Clock

Repeat

Continue until last phase sample

After last phase sample at same time:

Phase sample valid =“0”

Phase sample from DRFM don’t care

Clock

As long as ODVout is “1”, IS and QS (The output from I and Q) are valid.

Continue to check until ODVout =“0”

ODV out goes low after 4 +n(#of range bins) clocks after last edge that loads valid DRFM sample into top of RB.

I in and Q in to next RB =“0”

I. TEST AND RESULTS

After initial tests, the VHDL design was verified using 31 different test vectors. The result was compared with results generated in Matlab by another researcher and the results obtained when simulating with T-Spice (Ref.[2]).

DRFM	INC	GAIN	Matlab Result		T-Spice Result		VHDL Result	
phase	angle	code	Iout	Qout	Iout	Qout	Iout	Qout
1	1	1	1.8	0.8	1.75	0.75	1.75	0.75
2	2	2	2.8	2.8	2.75	2.75	2.75	2.75
3	3	3	3.1	7.3	3.0	7.25	3.0	7.25
4	2	5	6.1	14.6	6.0	14.5	6.0	14.5
5	1	6	12.3	29.3	12.25	29.25	12.25	29.25
6	0	7	24.5	58.5	24.5	58.5	24.4	58.5
7	1	11	0	127	0	127	0	127
8	2	13	-98	234	-98	234	-98	234
9	3	14	-360	360	-360	360	-360	360
10	2	15	-720	720	-720	720	-720	720
11	1	0	-0.7	0.7	-0.75	0.5	-0.75	0.5
12	0	1	-1.4	1.4	-1.5	1.3	-1.5	1.25
13	1	2	-3.7	1.5	-3.75	1.5	-3.75	1.5
14	2	3	-7.9	0	-8	0	-8	0
15	3	5	-14.6	-6.1	-14.75	-6.3	-14.75	-6.25
16	2	6	-29.3	-12.3	-29.25	-12.3	-29.25	-12.25
17	1	7	-58.5	-24.5	-58.5	-24.5	-58.5	-24.5
18	0	11	-117	-49	-117	-49	-117	-49
19	1	13	-180	-180	-180	-180	-180	-180
20	2	14	-196	-468	-196	-468	-196	-468
21	3	15	0	-1016	0	-1016	0	-1016
22	2	0	0	-1	0	-1	0	-1
23	1	1	0	-2	0	-2	0	-2
24	0	2	0	-4	0	-4	0	-4
25	1	3	3.1	-7.3	3	-7.5	3	-7.5
26	2	5	11.3	-11.3	11.25	-11.3	11.25	-11.25
27	3	6	29.3	-12.3	29.25	-12.3	29.25	-12.25
28	2	7	58.5	-24.5	58.5	-24.5	58.5	-24.5
29	1	11	117	-49	117	-49	117	-49
30	0	13	234	-98	234	-98	234	-98
31	1	14	508	0	508	0	508	0

Table 2. Test Vectors and Results.

Marginal differences due to rounding implementations can be observed between the Matlab results and the two simulated results in Table 2.

J. VERIFICATION OF 8 RANGE-BIN MODULATOR

The 8 Range-bin modulator was, as the single range-bin, created schematically in S-Edit by another project researcher. The schematic was then exported as a VHDL file. A new VHDL design was created and the necessary amendments to the design were implemented.

Verification was initially obtained by simple handcrafted vectors based on the 31 vectors used for verification of the single range-bin modulator. Later, Matlab created vectors were used.

The initial values of I and Q were yet again set to zero and the effect and expected values of the control signals were verified.

1. Underlying Mathematics

The mathematics for a multiple range bin modulator works the same way as a single range-bin modulator. The output result is the result of a correlation like summation of all the range-bins. The first valid output equals the value of the first DRFM phase value affected by the programming of the last range-bin. Subsequently, the second valid output is a summation of the second DRFM value passing through the last range bin and the first DRFM value passing through the second to last range bin, and so on until there is no more phase data.

In the case of an 8 Range-bin modulator, at least eight DRFM phase values are needed to fill up the system and influence the output. An overview of the system is shown in Table 3.

RB 0	RB 1	RB 2	RB 3	RB 4	RB 5	RB 6	RB 7	Sum
Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Phase 7	Phase 8	Output 8
	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Phase 7	Output 7
		Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Output 6
			Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Output 5
				Phase 1	Phase 2	Phase 3	Phase 4	Output 4
					Phase 1	Phase 2	Phase 3	Output 3
						Phase 1	Phase 2	Output 2
							Phase 1	Output 1

Table 3. Overview of Expected Results after Eight DRFM Phase Values.

A valid output value can be expected from the last range-bin as long as phase values are being fed to the system.

2. Layout

The 8 range-bin modulator consists of eight identical single range-bin modulators fed in parallel. In addition, extra control signals have been added in order to enable programming of the individual range-bins. Figure 36 depicts the eight different range-bin entities and the control signals. It should be viewed from left to right where all the signals entering the system are applied to the left and the results can be viewed exiting the last range-bin to the right.

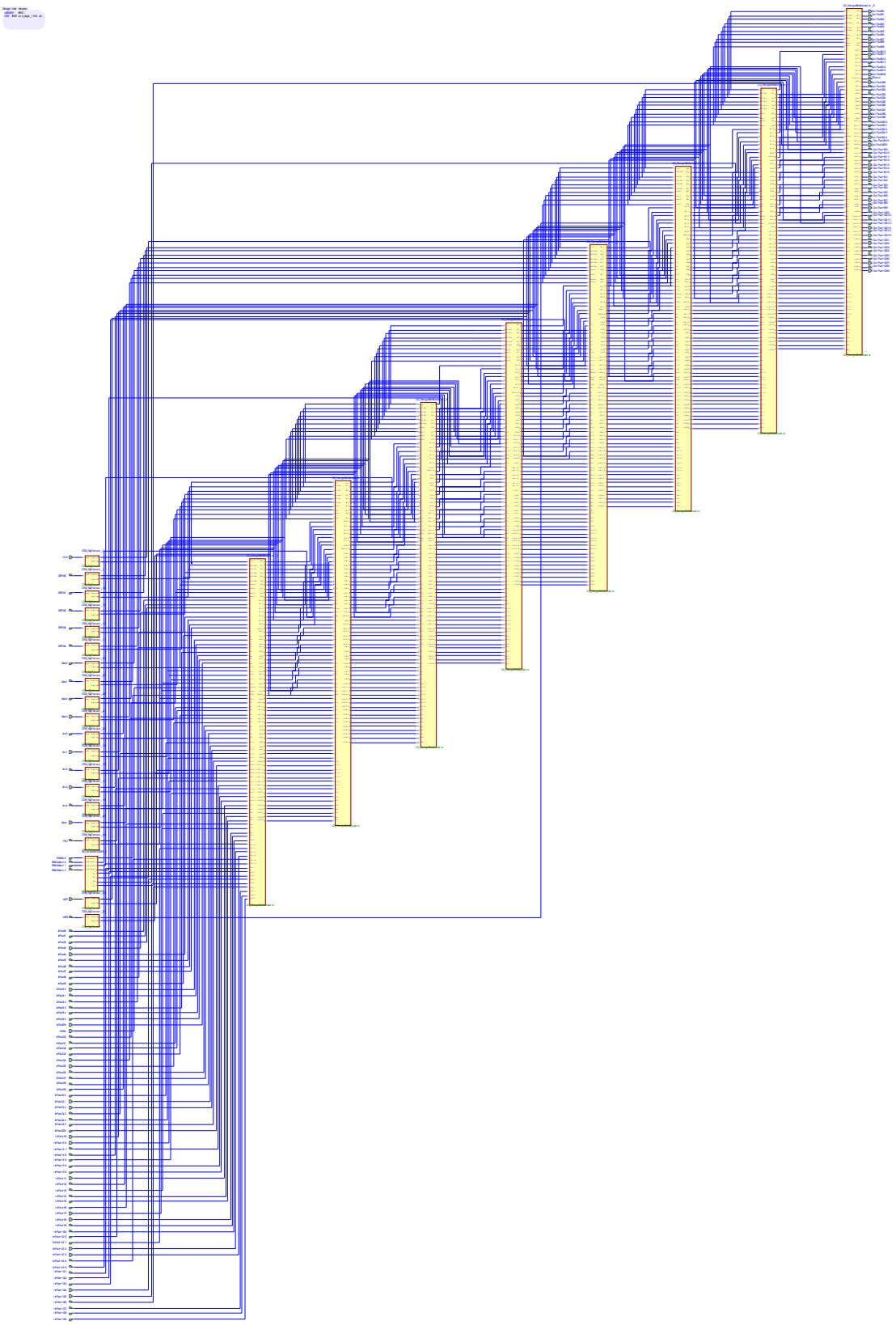


Figure 36. VHDL Block Diagram of the 8 Range Bin Modulator.

3. Additional Control Signals

New control signals were developed in order to program multiple range-bins. A block containing a 3 bit binary signal was created to choose from range-bin 0 to range-bin 7 and enable the signal. The output from this block is connected to the Program Range Bin port of the individual range-bins.

Enable : a logic one activates the block in order to transfer data from the block to the individual range-bins.

RBinSelect0—2 : steers the programming to the correct range-bin

4. Driver Input and Test Algorithm

The algorithm used to verify a multiple range bin design builds on the one used for the single range-bin. However, the number of range-bins determines the **ODVout** signal, which is one of the more interesting in order to verify correctness. The algorithm below depicts implementation of an 8 range-bin modulator using a clock speed of 500 MHz.

a. Test Algorithm

At same time:

Set Phase Inc to desired value for Rb-7

Set Gain to desired value for Rb-7

Set Enable to “1”

Set RbinSelect0 to “1”

Set RbinSelect1 to “1”

Set RbinSelect2 to “1”

Set Phase sample valid to “0”

Set Use new programming =“0”

Set Operate/Main to “1”

ODVin to =“0”

Clock rising edge 1 ms at 500MHz

At same time:

Set Phase Inc to desired value for Rb-6

Set Gain to desired value for Rb-6

Set RbinSelect0 to “0”

Set RbinSelect1 to “1”

Set RbinSelect2 to “1”

All else the same

Clock rising edge 3 ms at 500MHz

---Repeat until all Range-bins are programmed, last one is:

At same time:

Set Phase Inc to desired value for Rb-0

Set Gain to desired value for Rb-0

Set RbinSelect0 to “0”

Set RbinSelect1 to “0”

Set RbinSelect2 to “0”

All else the same

Clock rising edge 15 ms at 500MHz

At same time:

Set Enable to “0”

Set Use new programming =“1”

Phase sample valid still =“0”

All else don’t care

Clock rising edge 17 ms at 500MHz

At same time:

Set Use new programming =“0”

Set Phase sample valid =“1”

Operate/Main still “1”

Set first phase sample from DRFM to desired value

All else don’t care

Clock rising edge 19 ms at 500MHz

At same time:

Change phase sample from DRFM all other signal the same

Clock

Repeat until last valid phase sample, this example has 10 valid samples.

Clock rising edge 37 ms at 500MHz

After last phase sample at same time:

Phase sample valid =“0”

Phase sample from DRFM don't care

Clock rising edge 39 ms at 500MHz

Continue to check until ODVout from Rb-7 ="0"

4 clocks after 1st valid DRFM data is clocked into top of all RB

ODV out from RB-7 goes high, i.e. at 27 ms ODVout ="1"

ODV out goes low after 4 +n(#of range bins) clocks after last edge that loads valid DRFM sample into top of RB.

ODV out from RB-7 goes low after (37 + 4*2+8*2ms) =>

ODVout ="0" at 61 ms

Figure 37 presents a waveform window from Active HDL where these numbers can be verified. In this case, the virtual bus VBUS3 represents the DRFM phase, VBUS4 is the Gain coefficient, and VBUS5 is the Phase increment coefficient.

Furthermore, the first VBUS0 is the range-bin being programmed and the second VBUS0 is I_{out}, Q_{out} is represented by VBUS2.

The first and second window overlaps and depicts the signal values from 0ms to 80ms.

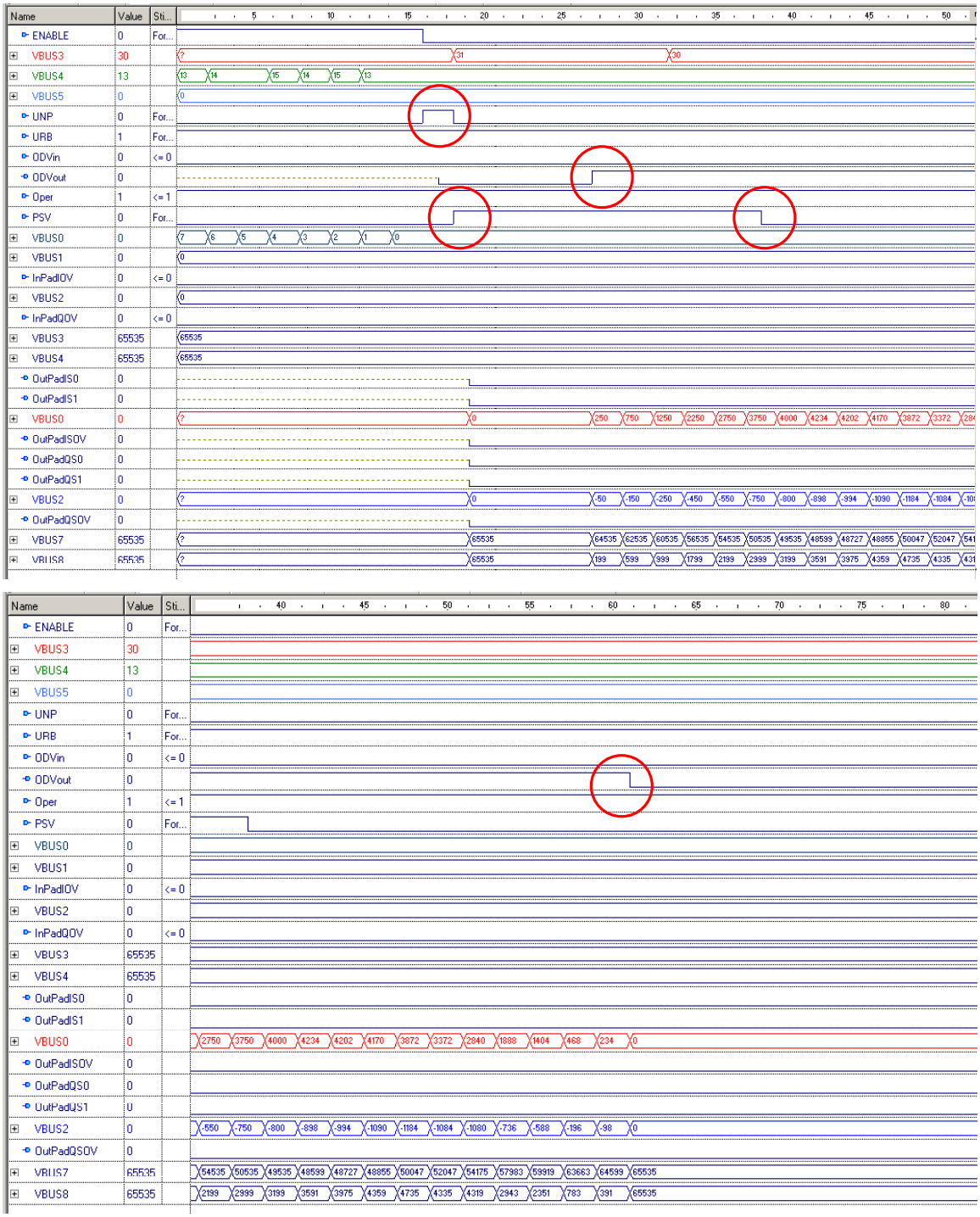


Figure 37. Waveform Window for an 8 Range-Bin Modulator.

As can be seen in the circles, the implementation of UNP and PSV results in an ODVout valid for the expected period of time.

I_{out} and Q_{out} can be viewed at the bottom of the window. In this situation, the 8^{th} through the 10^{th} output sample represents values from all range-bins. After the 10^{th} output, the value tapers down to zero when the sample no longer is valid.

5. Tests and Results

Four lengthy Matlab generated vectors were used to verify the 8 range-bin modulator. To exemplify the input and expected output, the following values were generated for two of these. Using the first 10 phase samples from each vector the programming of the 8 range-bin modulator was made with the values in Tables 4 and 6.

a. Vector 8A

Programming of range-bins:

Rb #	Multiplication Factor	Gain Code	Phase Increment
Rb 7	256	13	0
Rb 6	512	14	0
Rb 5	512	14	0
Rb 4	1024	15	0
Rb 3	512	14	0
Rb 2	1024	15	0
Rb 1	256	13	0
Rb 0	256	13	0

Table 4. Programming of Vector 8A.

Matlab describes the results as an imaginary number. The results after the first 10 DRFM samples of Vectors 8A and 8B are shown in Tables 5 and 7 respectively.

Sample	DRFM Phase	Matlab result	I_{Out}	Q_{out}
1	31	250-50j	250	-50
2	31	750-150j	750	-150
3	31	1250-250j	1250	-250
4	31	2250-450j	2250	-450
5	31	2750-550j	2750	-550
6	31	3750-750j	3750	-750
7	31	4000-800j	4000	-800
8	30	4234-898j	4234	-898
9	30	4202-994j	4202	-994
10	30	4170-1090j	4170	-1090

Table 5. Result of Vector 8A.

The results of Table 5 can be viewed in Figure 37. Given the fact that it is a form of correlation, it becomes tedious to verify by hand. However, sample one should provide a result for I of Cos 348.75* 256 which is equal to 251.1, and Q should be Sin 348.75 *256 which is equal to –49.9. After sample two, the calculation would be: for I (Cos 348.75* 256) + (Cos 348.75* 512) = 753.2 and for Q: (Sin 348.75* 256) + (Sin 348.75* 512) = -149.8. As in the case with the single range-bin, the lack of fidelity of the 16-bit adder representation creates rounding errors that account for minor differences.

b. Vector 8B

Programming of range-bins:

Rb #	Multiplication Factor	Gain Code	Phase Increment
Rb 7	256	13	28
Rb 6	256	13	26
Rb 5	256	13	22
Rb 4	256	13	16
Rb 3	512	14	16
Rb 2	256	13	13
Rb 1	256	13	16
Rb 0	256	13	15

Table 6. Programming Vector 8B.

Sample	DRFM Phase	Matlab Result	I_{Out}	Q_{out}
1	0	180-180j	180	-180
2	31	240-446j	240	-446
3	31	94-690j	94	-696
4	31	-204-674j	-204	-674
5	31	-708-624j	-708	-624
6	31	-912-382j	-912	-382
7	31	-1134-344j	-1134	-344
8	30	-1424-266j	-1424	-266
9	30	-1458-222j	-1458	-222
10	30	-1496-190j	-1496	-190

Table 7. Result of Vector 8B.

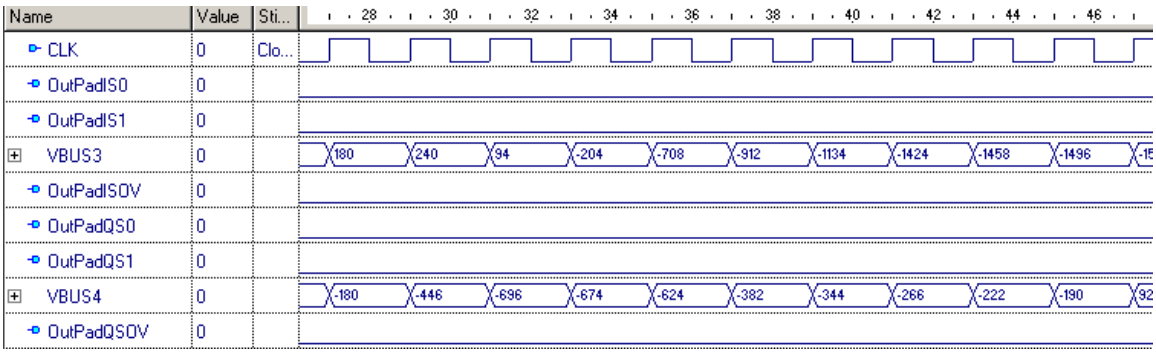


Figure 38. Result as it Appears on the Wave Form Window for Vector 8B. VBUS3 is I_{out} and VBUS4 is Q_{out}.

Figure 38 shows the Waveform window during verification of Vector 8B. The output signals have been combined into buses and converted into decimal notation for easier reading.

V. VERIFICATION OF 32 RANGE-BIN MODULATOR

A. CREATION OF 32 RANGE-BIN MODULATOR

The 32 Range-bin modulator was created by software. It was programmed in VHDL using the Active-HDL text editor. The 32 Range-bin modulator is a super class consisting of four 8 Range-bin modulators with the I and Q signals connected in series and the programming and control signals connected in parallel. Verification of the 32 Range-bin modulator again had to be conducted using Matlab generated vectors.

As an early proof of concept, a 2 Range-bin modulator was programmed using the Single Range-bin modulator as a building block. This entity was tested using a combination of the initial test vectors and the result was deemed satisfactory.

The different codes used to create and test the 32 range-bin modulator can be viewed in Appendix F.

1. Underlying Mathematics

The same effect of the individual single Range-bins can be observed in the 32 Range-bin modulator as in the case with the 8 Range-bin modulator. The difference is that 32 phase samples are needed instead of eight in order to fill the system. As in the 8 Range-bin case, the output is an added correlation of the 32 Range-bins where the sum is read after the last range bin. See Table 3.

2. Layout

Figure 39 displays the layout of the 32 Range-bin modulator. The tool, from the VHDL code produced, automatically created the block diagram. Global signals, or signals that are fed in parallel to all the range bins, are depicted without connecting wires.

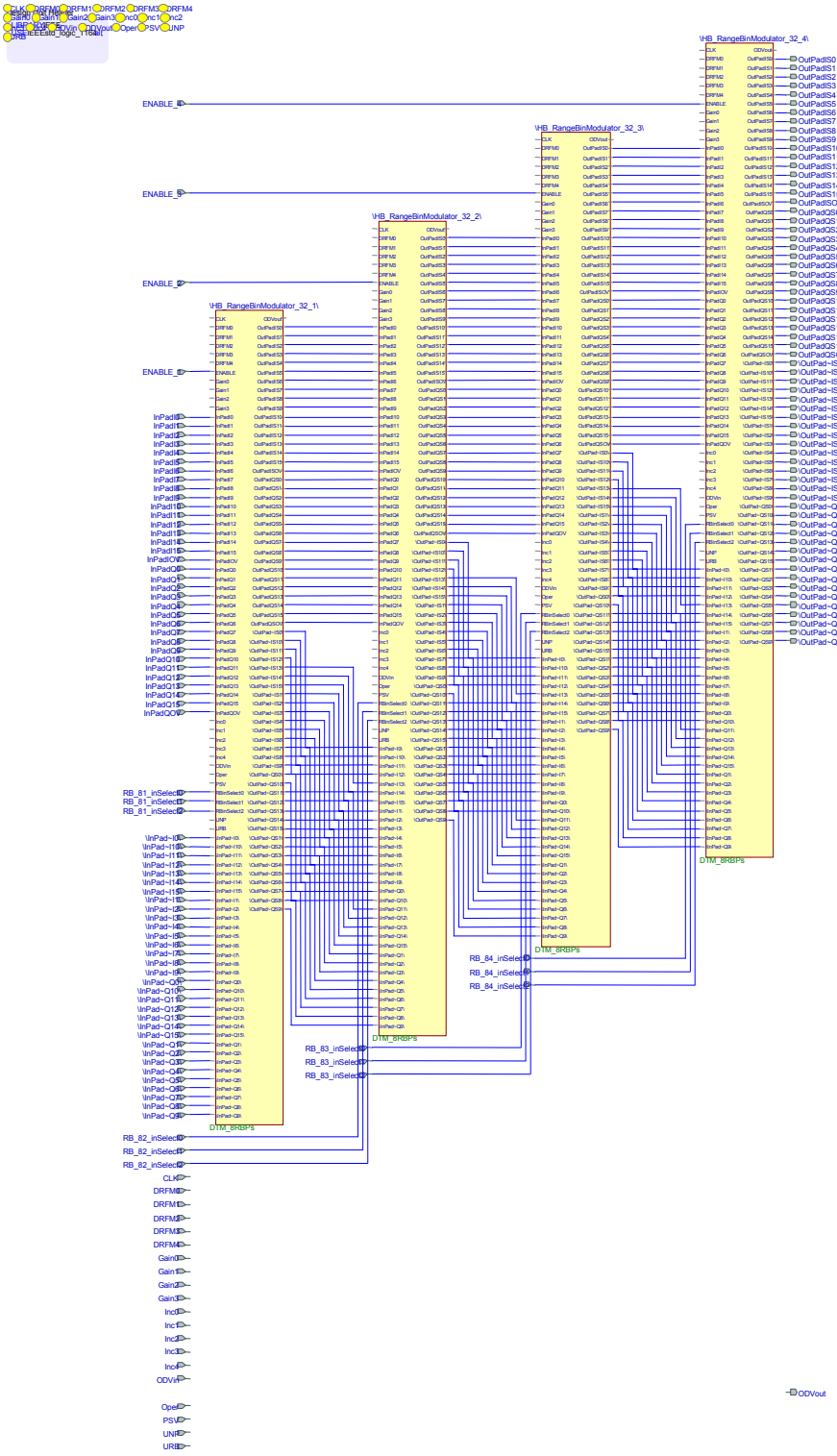


Figure 39. VHDL Block Diagram of the 32 Range–Bin Modulator.

3. Additional Control Signals

In order to program each individual range-bin, the **Enable** signal had to be split into **Enable 1—4** and the **RBinSelect0—2** was divided into **RB_81_inSelect0—2** to **RB_84inSelect0—2.**

4. Driver Input and Test Algorithm

In order to verify the 32 Range-bin modulator, the algorithm described in IV.J.4.a will be used. The only difference to the original algorithm is the expectations on **ODVout.** Since the four different modules of 8 Range-bins are fed with phase values in parallel, ODVout from the last range bin will go low after 4 (due to the single Range bin) + 8 (due to the 8 Range-bin module) clock cycles after the last loaded valid phase sample.

B. IMPLEMENTATION OF TEST CASES

Matlab was used to produce the 32 Range-bin, 32 pulse false target in Figure 40. Four of these pulses were singled out during the verification effort: the first and the last and the only two pulses generating adder overflow (Ref.[13]). In the next section, two of those pulses will be presented in vector form.

Matlab generated the following images and output signals utilizing 32 range-bins.

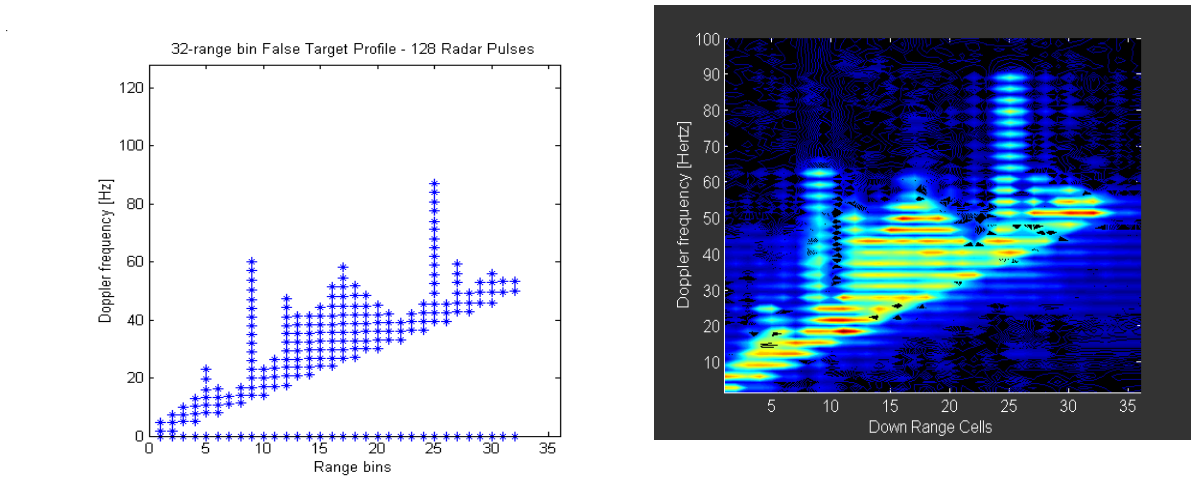


Figure 40. Matlab Created False Target, Input Template (Left) and ISAR Image (Right).

C. SIMULATION AND VERIFICATION

1. Programming of Vector 32A

Programming of range-bins:

Rb #	Multiplication Factor	Gain Code	Phase Increment
Rb 31	128	11	0
Rb 30	128	11	0
Rb 29	128	11	0
Rb 28	256	13	0
Rb 27	256	13	0
Rb 26	256	13	0
Rb 25	128	11	0
Rb 24	128	11	0
Rb 23	1024	15	0
Rb 22	256	13	0
Rb 21	256	13	0
Rb 20	512	14	0
Rb 19	512	14	0
Rb 18	512	14	0
Rb 17	512	14	0
Rb 16	512	14	0
Rb 15	512	14	0
Rb 14	512	14	0
Rb 13	512	14	0
Rb 12	256	13	0
Rb 11	256	13	0
Rb 10	128	11	0
Rb 09	128	11	0
Rb 08	256	13	0
Rb 07	1024	15	0
Rb 06	128	11	0
Rb 05	256	13	0
Rb 04	128	11	0
Rb 03	128	11	0
Rb 02	256	13	0
Rb 01	128	11	0
Rb 00	128	11	0

Table 8. Programming of Vector 32A.

2. Result of Vector 32A

Sample	DRFM Phase	Matlab Result	VHDL I_{Out}	VHDL Q_{out}
1	0	127+0j	127	0
2	0	254+0j	254	0
3	0	381+0j	381	0
4	0	635+0j	635	0
5	0	889+0j	889	0
6	0	1143+0j	1143	0
7	0	1270+0j	1270	0
8	1	1395+25j	1395	25
9	1	2409+50j	2409	50
10	1	2661+75j	2661	75
11	1	2911+125j	2911	125
12	2	3407+199j	3407	199
13	2	3903+273j	3903	273
14	3	4390+344j	4390	344
15	3	4869+439j	4869	439
16	4	5318+728j	5318	728
17	4	5768+889j	5768	889
18	5	6207+1042j	6207	1042
19	5	6626+1264j	6626	1264
20	6	6742+1643j	6742	1643
21	7	6827+1902j	6827	1902
22	7	6742+2296j	6742	2296
23	8	6660+2619j	6660	2619
24	9	6563+3065j	6563	3065
25	10	7278+3433j	7278	3433
26	10	6968+3904j	6968	3904
27	11	6848+4246j	6848	4246
28	12	6404+4641j	6404	4641
29	13	5897+5012j	5897	5012
30	14	5653+5304j	5653	5304
31	15	5074+5577j	5074	5577
32	16	4392+5946j	4392	5946
33	17	3622+5992j	3622	5992
34	18	2963+6083j	2963	6083
35	19	2162+6004j	2162	6004
36	20	1293+5985j	1293	5985
37	21	558+5741j	558	5741
38	23	-215+5550j	-215	5550
39	24	--845+5098j	-845	5098
40	25	-1494+4682j	-1494	4682
41	26	-1962+4082j	-1962	4082
42	28	-2400+3539j	-2400	3539
43	29	-2625+2879j	-2625	2879
44	30	-2894+2275j	-2894	2275
45	0	-3046+1679j	-3046	1679
46	1	-2763+990j	-2763	990
47	3	-2739+478j	-2739	478
48	4	-2585-27j	-2585	-27
49	6	-2355-454j	-2355	-454
50	7	-1791-623j	-1791	-623

Table 9. Result of Vector 32A.

The vector in Table 9 represents a straight forward approach where no overflow was expected nor detected. Programming the range bins is slightly easier and less time consuming when the phase increment values are set to zero.

By running the vector 50 samples deep, filled range bins throughout the design were achieved. Table 9 represents the results of the verification in tabular form while Figure 41 depics part of the Waveform window.

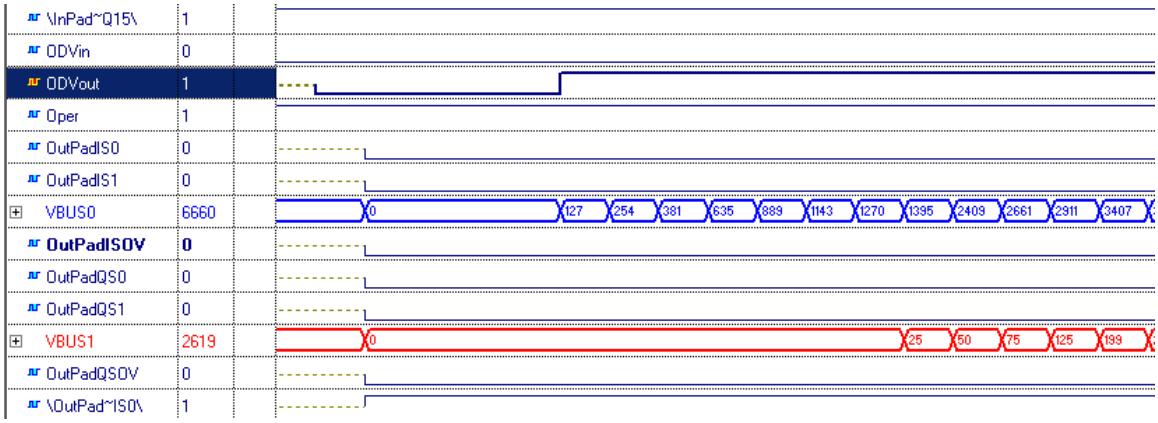


Figure 41. Portion of the Wave Form Editor Displaying the Initial I (Blue) and Q (Red) Values for Vector 32A.

3. Programming of Vector 32B

As is shown in Table 10 Vector 32B uses the same gain, i.e., the same multiplication as 32A. However, the addition of a phase increment creates an overflow in some of the range bins after a certain amount of phase samples.

Programming of range-bins:

Rb #	Multiplication Factor	Gain Code	Phase Increment
Rb 31	128	11	0
Rb 30	128	11	0
Rb 29	128	11	1
Rb 28	256	13	1
Rb 27	256	13	2
Rb 26	256	13	1
Rb 25	128	11	1
Rb 24	128	11	2
Rb 23	1024	15	5
Rb 22	256	13	2
Rb 21	256	13	3
Rb 20	512	14	5
Rb 19	512	14	4
Rb 18	512	14	5
Rb 17	512	14	5
Rb 16	512	14	6
Rb 15	512	14	6
Rb 14	512	14	6
Rb 13	512	14	6
Rb 12	256	13	6
Rb 11	256	13	6
Rb 10	128	11	5
Rb 09	128	11	6
Rb 08	256	13	6
Rb 07	1024	15	10
Rb 06	128	11	6
Rb 05	256	13	8
Rb 04	128	11	7
Rb 03	128	11	7
Rb 02	256	13	8
Rb 01	128	11	8
Rb 00	128	11	8

Table 10. Programming of Vector 32B.

4. Result of Vector 32B

Sample	DRFM Phase	Matlab Result	I_{Out}	Q_{out}
1	0	127+0j	127	0
2	0	254+0j	254	0
3	0	379+25j	379	25
4	0	629+75j	629	75
5	0	863+173j	863	173
6	0	1113+223j	1113	223
7	0	1238+248j	1238	248
8	1	1353+322j	1353	322
9	1	1919+1195j	1919	1195
10	1	2145+1317j	2145	1317
11	1	2341+1507j	2341	1507
12	2	2595+1999j	2595	1999
13	2	2931+2431j	2931	2431
14	3	3185+2923j	3185	2923
15	3	3425+3435j	3425	3435
16	4	3381+4067j	3381	4067
17	4	3485+4680j	3485	4680
18	5	3562+5272j	3562	5272
19	5	3565+5889j	3565	5889
20	6	3291+6314j	3291	6314
21	7	3122+6719j	3122	6719
22	7	2752+6965j	2752	6965
23	8	2407+7231j	2407	7231
24	9	1905+7566j	1905	7566
25	10	981-7780j	981	-7780
26	10	330-7623j	330	-7623
27	11	-256-7379j	-256	-7379
28	12	-1009-7375j	-1009	-7375
29	13	-1769-7504j	-1769	-7504
30	14	-2466-7396j	-2466	-7396
31	15	-3258-7699j	-3258	-7699
32	16	-4203+8159j	-4203	8159
33	17	-4898+7546j	-4898	7546
34	18	-5561+6960j	-5561	6960
35	19	-6120+6128j	-6120	6128
36	20	-6724+5057j	-6724	5057
37	21	-7095+4113j	-7095	4113
38	23	-7408+2904j	-7408	2904
39	24	-74621850j	-7462	1850
40	25	-7378+625j	-7378	625
41	26	-7106-428j	-7106	-428
42	28	-6655-1567j	-6655	-1567
43	29	-6050-2439j	-6050	-2439
44	30	-5317-3364j	-5317	-3364
45	0	-4422-4096j	-4422	-4096
46	1	-3356-4371j	-3356	-4371
47	3	-2318-4770j	-2318	-4770
48	4	-4770-4881j	-4770	-4881
49	6	-230-4776j	-230	-4776
50	7	562-4206j	562	-4206

Table 11. Result of Vector 32B.

Through the Matlab simulation, an overflow after sample 25 through 31 was expected. A result easily detected in Table 11 above and viewed on the waveform editor shown in Figure 42.

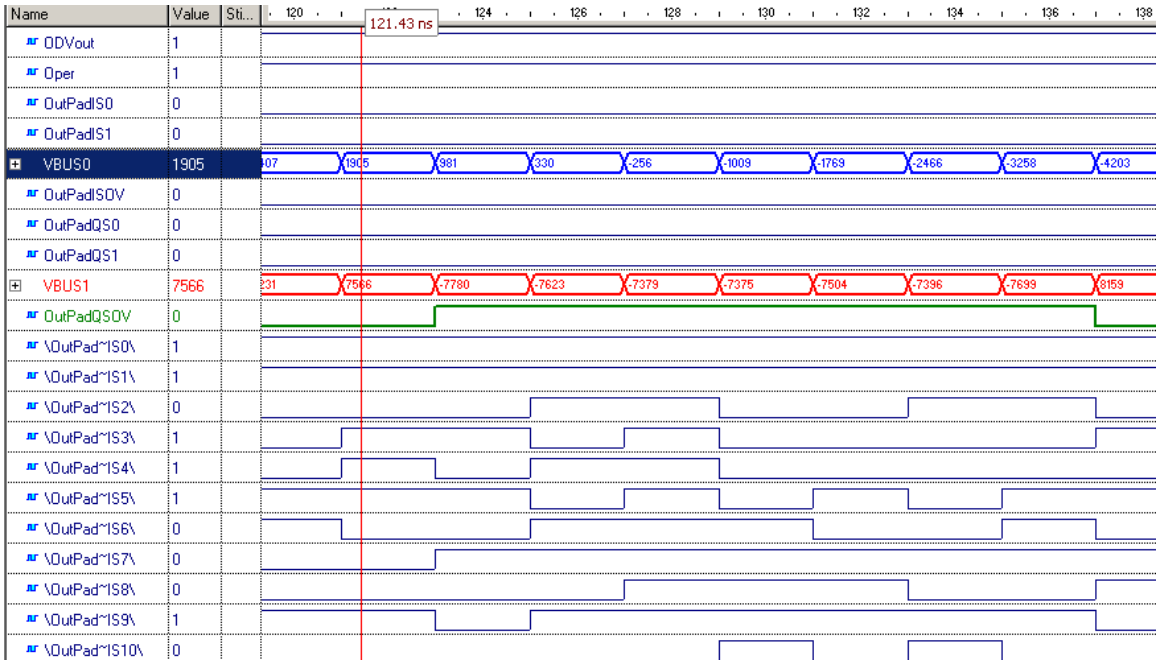


Figure 42. Portion of the Wave Form Editor Displaying I (Blue) and Q (Red) After Sample 25-31 and the Subsequent Overflow– OutpadQSOV (Green).

Equally successful results were obtained running the rest of the test vectors. A 100% correspondence with the expected, Matlab generated, results was obtained.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SUMMARY, CONCLUSION AND RECOMMENDATION

A. SUMMARY AND CONCLUSION

The main purpose of the research for this thesis was to find a method to perform simulations to verify hardware design. It has been clearly demonstrated that VHDL is a good choice of a language to model the design in order to save simulation time. Using VHDL and Active-HDL during the simulation and verification process saved several orders of magnitude of time.

Second, it has been shown that the VHDL version of the design acts as the original Tanner Tools Pro design. However, a few steps have to be remembered. When exporting the design, for instance, an attempt was made to implement bi-directional ports but unidirectional ports had to be used. Another discovery was that VHDL did not accept network output ports connected to inputs of other logic gates within the module. Such cases needed to be buffered by two inverters in series.

Third, VHDL was a good choice to create "super classes" with more Range-bin modulators than existed in the exported design. Thus, it was possible to verify the logic of a larger design. In this thesis, a design of 32 Range-bins has been verified. Two additional super classes of 128 and 512 Range-bins have also been programmed but not yet tested.

Fourth, using a functional white box approach to verify the design was successful. The algorithm developed in combination with Matlab vectors for the larger designs proved to be a good combination.

B. RECOMMENDATION

VHDL can be used in the future work of this project. The current 128 and 512 Range-bin designs can be verified to confirm their logic as well. New VHDL designs can easily be constructed when a new hardware design is created.

VHDL can also be used more actively, given time and resources. Currently, the different researchers doing the hardware design have named the same entity with different names. For instance, several inverters and other logic gates possess the same

functionality but with different names. This leads to Active-HDL generating "spaghetti code". With an early-implemented naming convention, this can be avoided. Adding a synthesizer, from a chip vendor, to Active-HDL, would make it possible for the project to use the VHDL design as a basis for manufacturing.

Increased cooperation with input from the Software Engineering Program, in a project such as this, could lead to opportunities to apply model checking or other software verification methodologies.

APPENDIX A. VHDL IMPLEMENTATION TUTORIAL

The purpose of this tutorial is to acquaint the reader with the methods involved in creating a VHDL design from an externally generated source file. It also reiterates the steps involved in library updates, naming conventions and behavior descriptions.

A. CREATING A NEW DESIGN

In this tutorial a VHDL design will be created using an externally generated source file.

1. Start Active-HDL. When the **Getting Started** dialog opens, select **Create new design,** click OK, Figure 43.

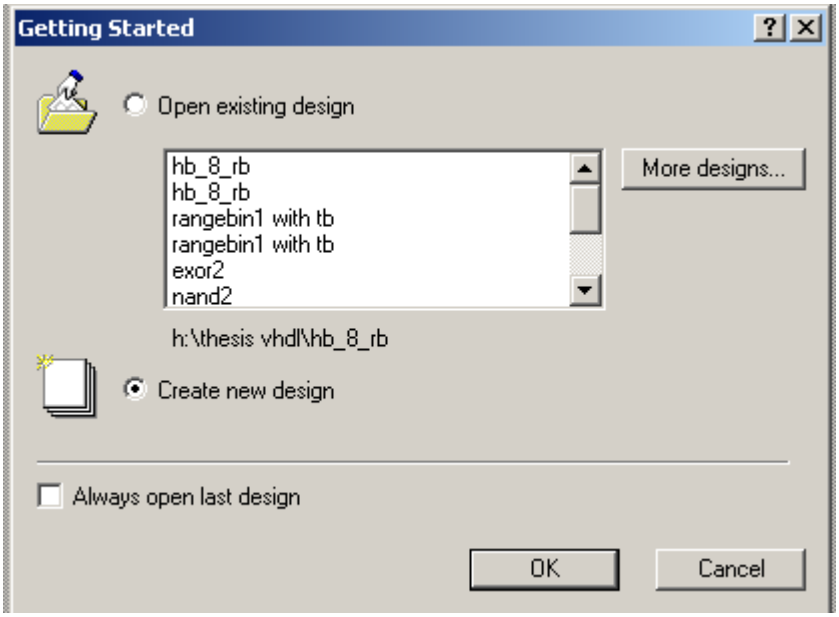


Figure 43. Getting Started Window in Active-HDL.

2. In the **New Design Wizard** first window, choose **Add existing resource file,** click **Next,** Figure 44.

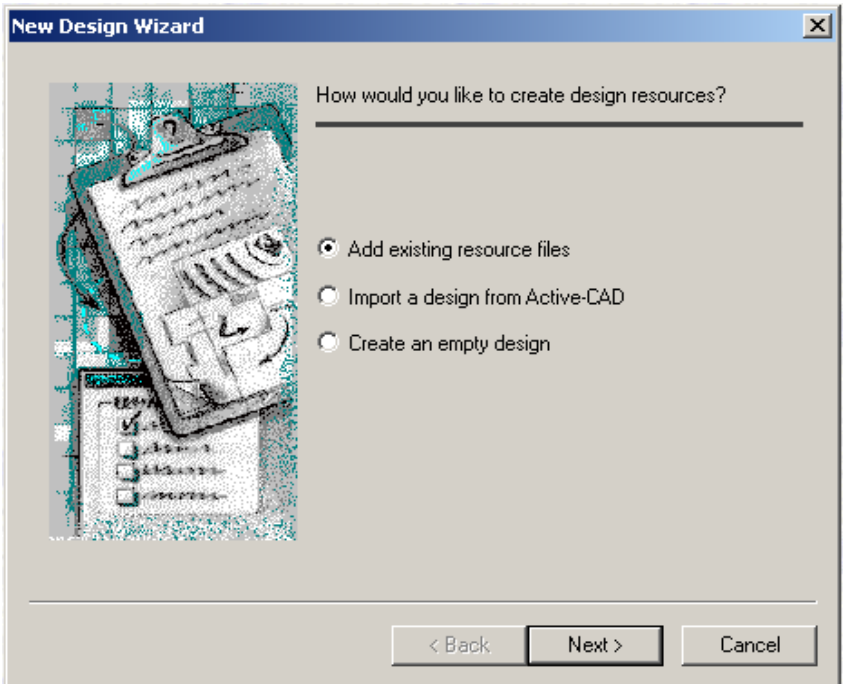


Figure 44. New Design Window in Active-HDL.

3. In the **New Design Wizard** second window Figure 45, click **Add files.**

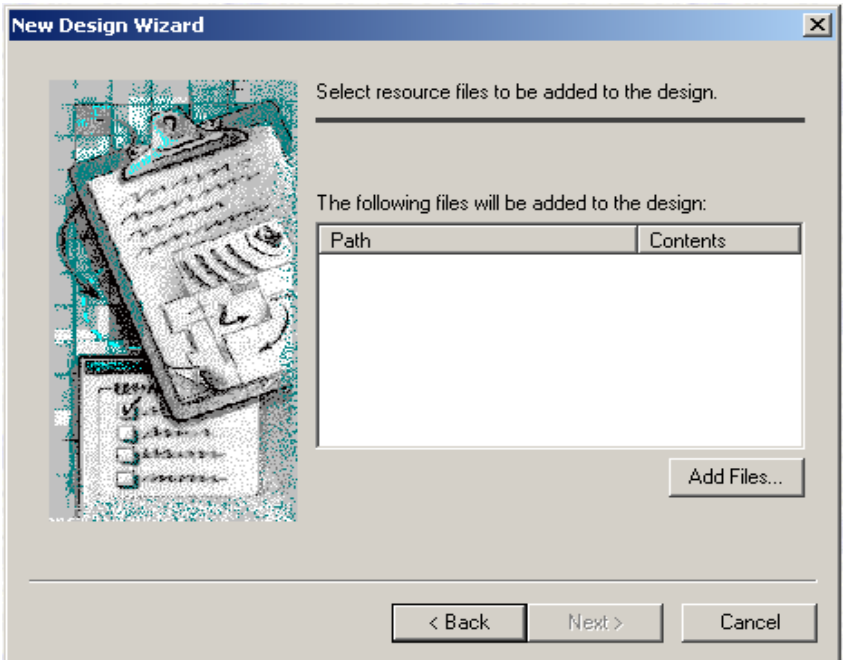


Figure 45. New Design Window in Active-HDL.

4. Find the appropriate file, Figure 46. Double click or select and click **Add.**

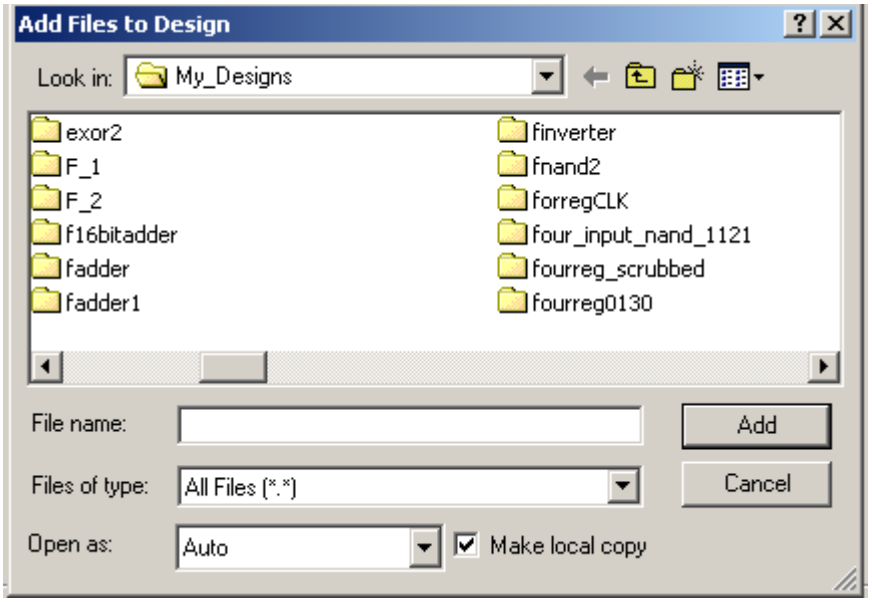


Figure 46. Find File Window in Active-HDL.

5. If file is correct click **Add Files,** shown in Figure 47**.**

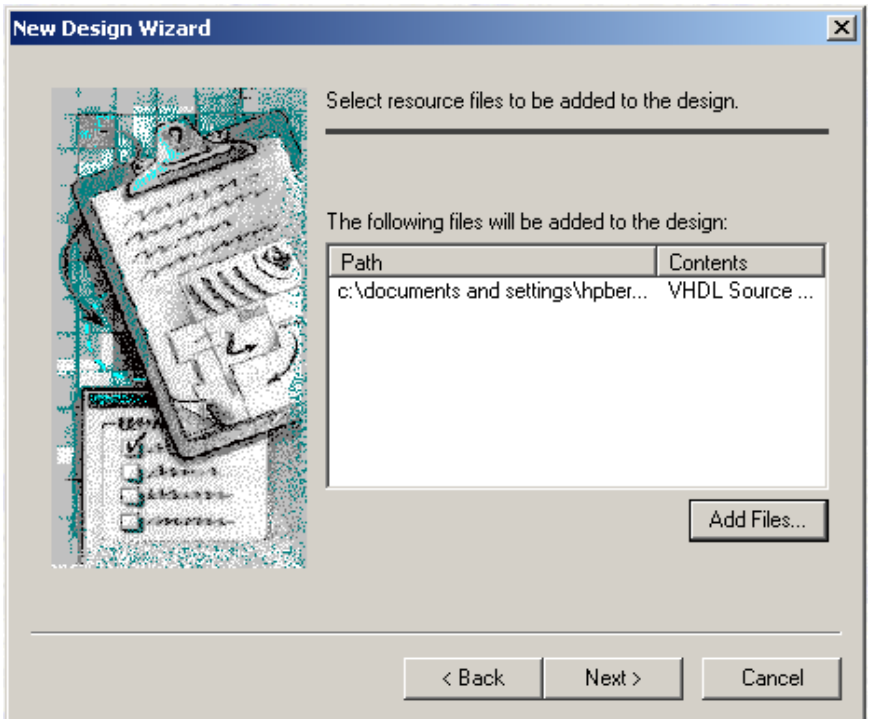


Figure 47. Chosen File in Active-HDL.

6. In the next window,Figure 48, make sure **HDL** is the Block Diagram Configuration. If other implementation tools are in use, check appropriate ones. In this tutorial, the default settings should be correct, click **Next.**

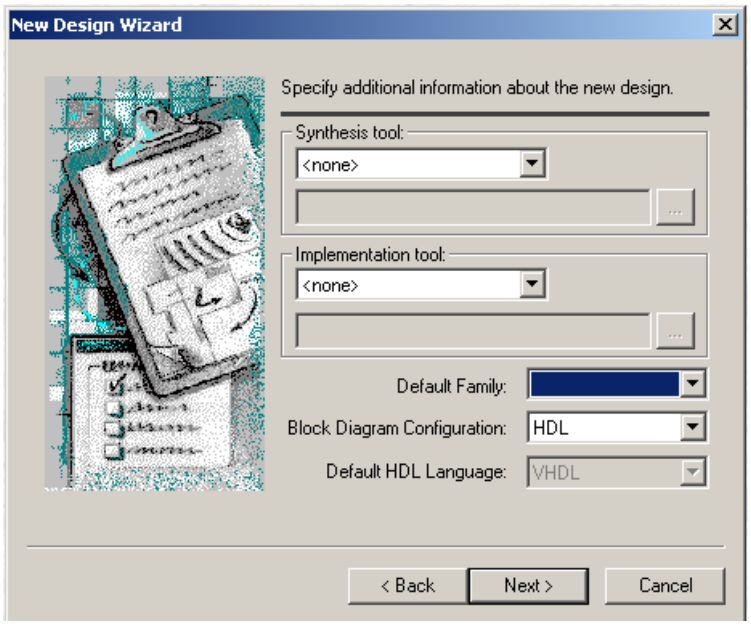


Figure 48. Configuration of Active-HDL.

7. The new design is displayed with its address in Figure 49.

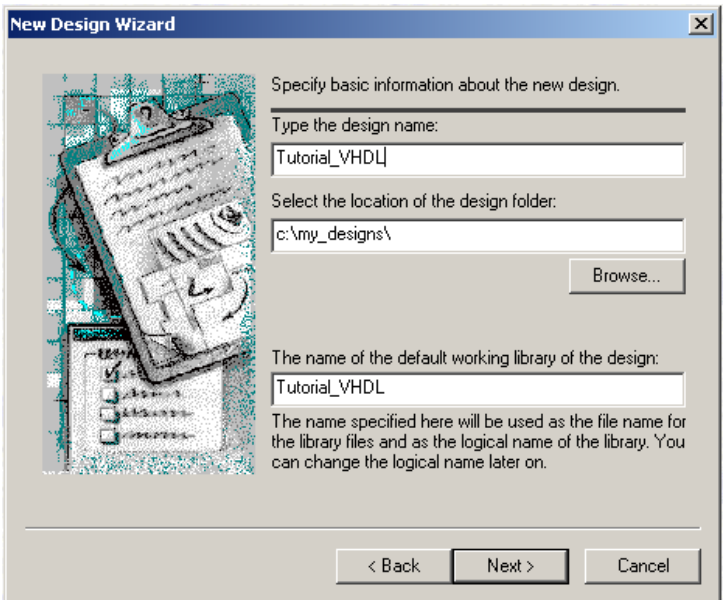


Figure 49. File Information in Active-HDL.

8. Specifications for the new design: make sure **Compile source files after creation** is checked in Figure 50. Click **Finish.**

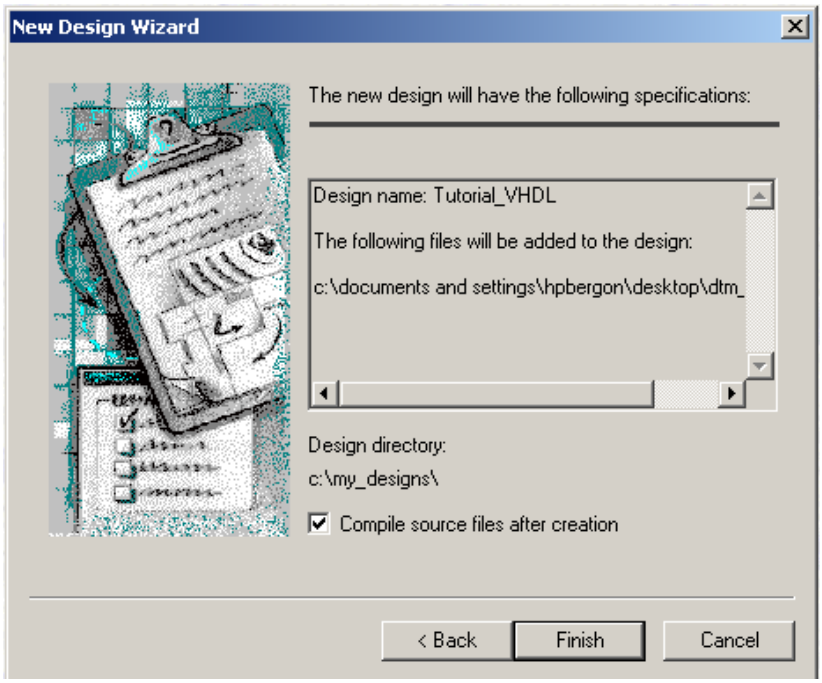


Figure 50. Design Specifications in Active-HDL.

9. At this point, the new design is launched. Note that the source file is compiled but that it contains errors. The errors in this example stem from three different sources:

- Lack of Library addition to each entity
- Lack of behavioral implementation to applicable entities
- Faulty component names

Depending on the file size, the errors may be more or less frequent. Each incorrect programming may also lead to more than one specific error.

Figure 51 will address these errors.

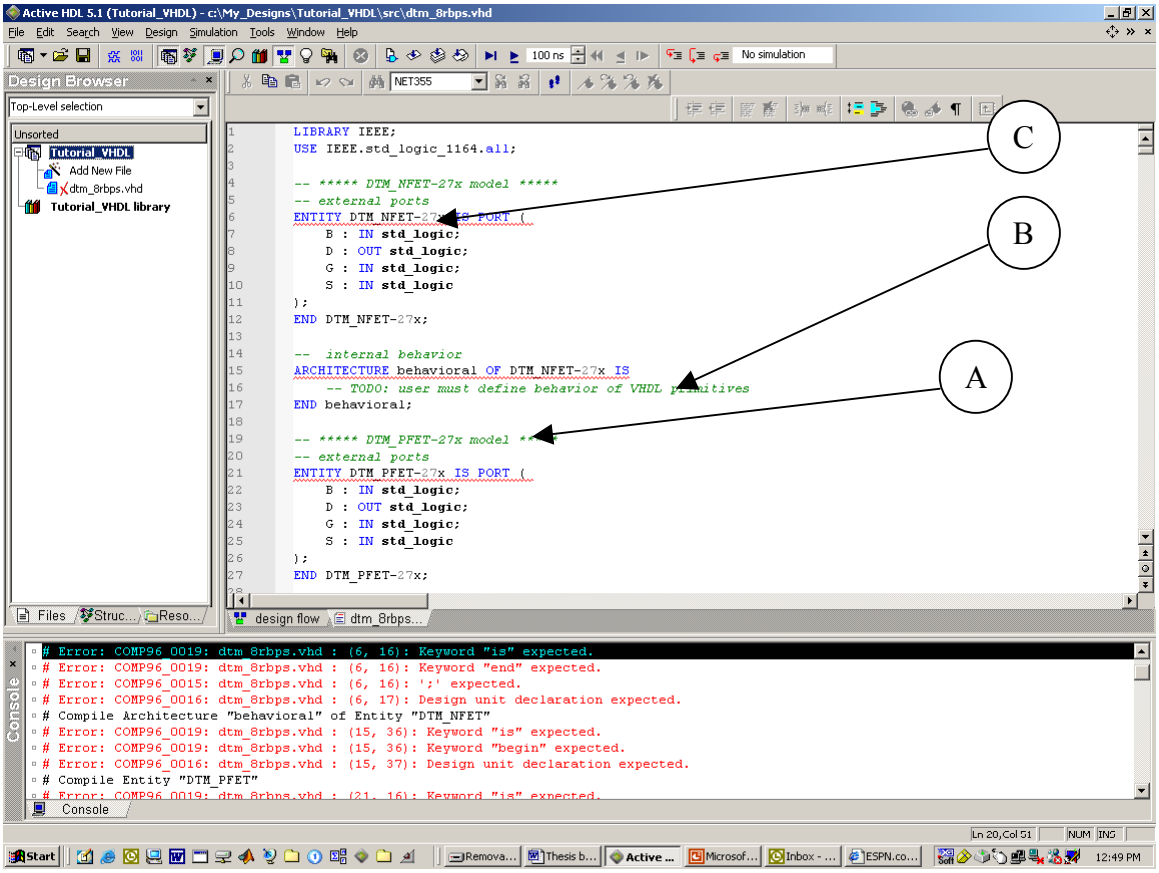


Figure 51. Active-HDL Design Launched from External Source File. Initial Errors According to Previous Page.

10. After all errors have been corrected, the file compiles correctly and it is now possible to open waveforms or create block diagrams.

APPENDIX B. TEST BENCH GENERATION TUTORIAL

In order to more easily test specific input signals, a test bench should be created. Usually, the user performs the functional simulation and defines test vectors required to verify operation of the design before generating a test bench. This tutorial will use a saved waveform file to generate the test bench, and then perform the functional simulation using the test bench macro.

After creating a waveform, running a simulation and saving the waveform:

1. Right-click the top-level design entity shown in Figure 52, and then choose **Generate Test Bench** from the shortcut menu to start the **Test Bench Wizard**.

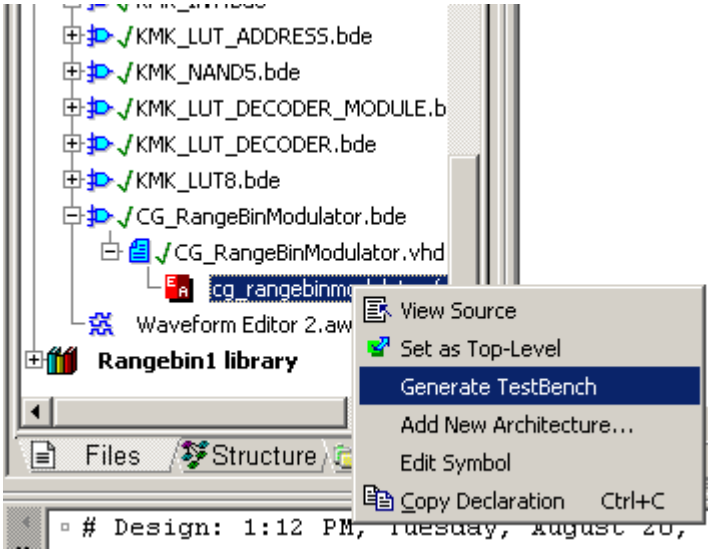


Figure 52. Test Bench Generation in Active-HDL.

2. For most purposes, select **Single Process** and click next in Figure 53.

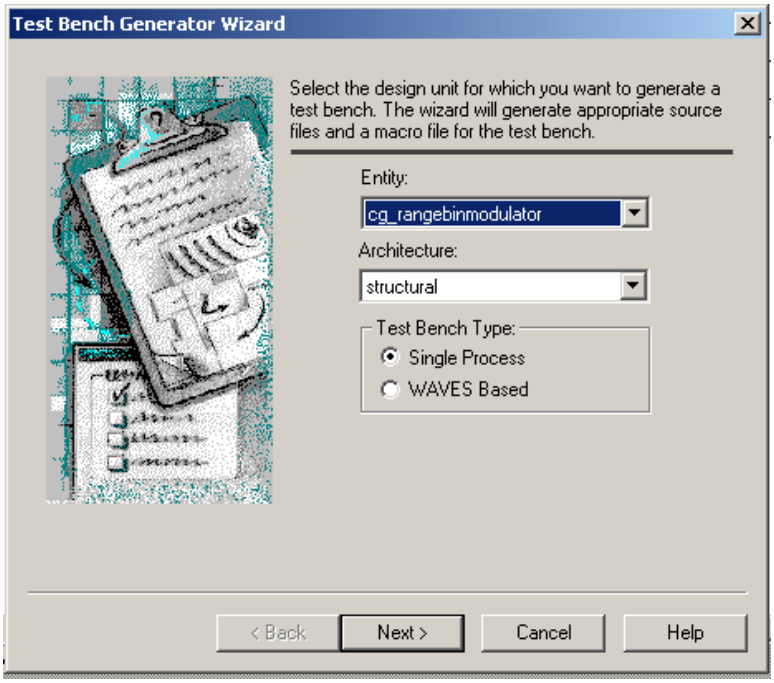


Figure 53. Test Bench Generation in Active-HDL.

3. Chose **Test vectors from file,** click **Browse** in Figure 54**.**

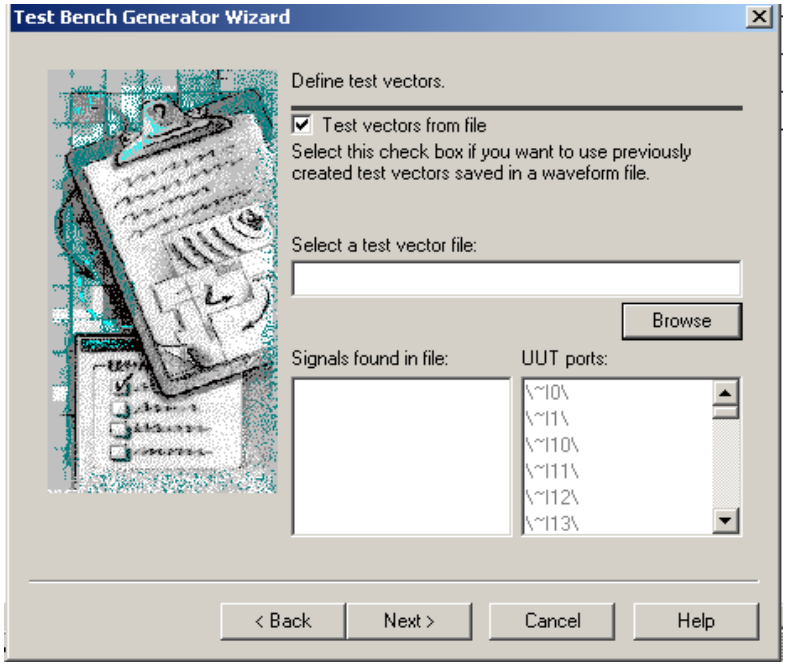


Figure 54. Test Bench Generation in Active-HDL.

4. Chose appropriate, saved waveform in Figure 55, click **Open.**

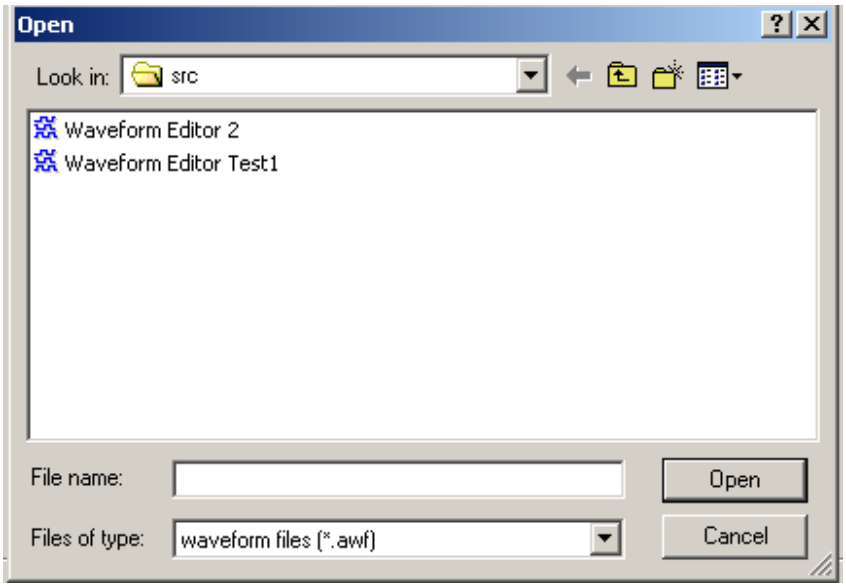


Figure 55. Test Bench Generation in Active-HDL.

5. Click, **Next** in Figure 56**.**

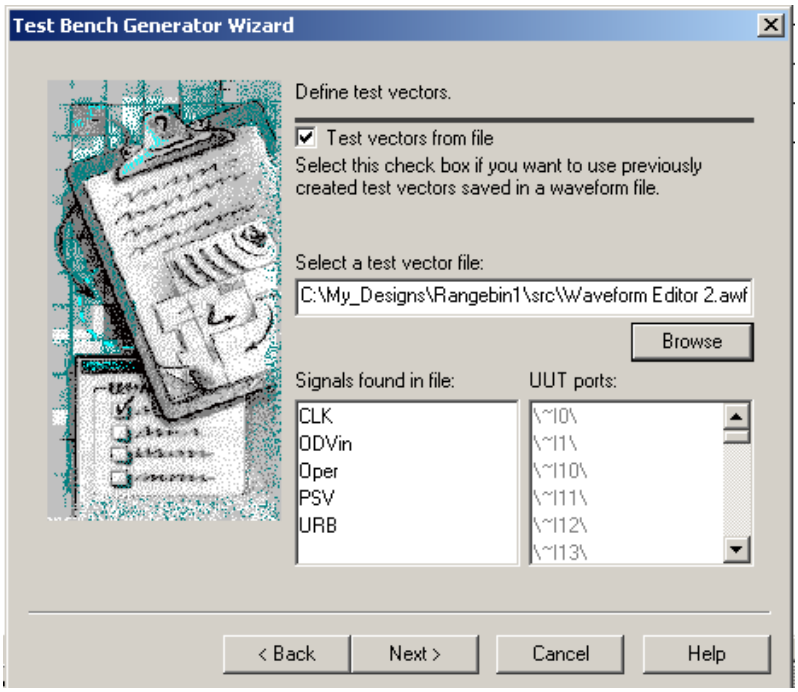


Figure 56. Test Bench Generation in Active-HDL.

6. Edit name or use default in Figure 57, click, **Next.**

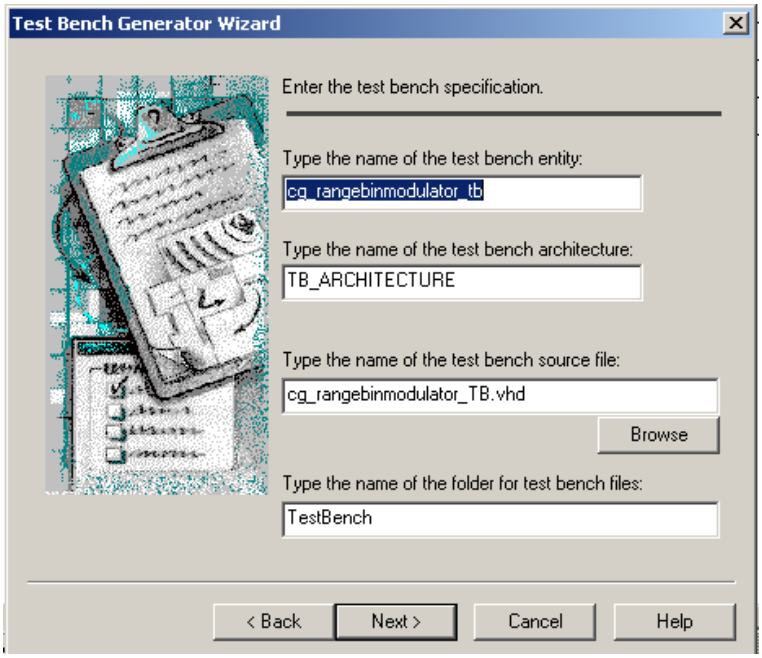


Figure 57. Test Bench Generation in Active-HDL.

7. Click **Finish** in Figure 58**.**

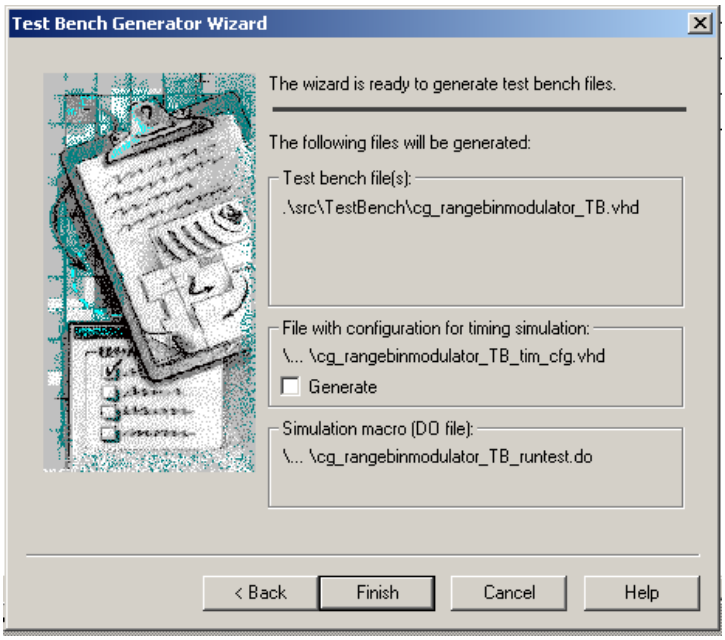


Figure 58. Test Bench Generation in Active-HDL.

8. The testbench is now complete and its file icon is shown in Figure 59.

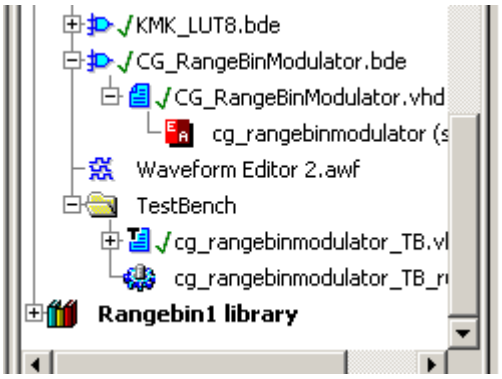


Figure 59. Test Bench Generation in Active-HDL.

Changes in the test bench are implemented in the test bench file. Initially, it will resemble the waveform used during generation, but it can be manually changed and executed again and again. Each time a new wave form is created it can be saved for future reference. The test bench is run by executing its macro.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. TOP-LEVEL VHDL CODE FOR A 1-BIT ADDER

```
-------------------------------------------------
-- Title      :
-- Design     : Rangebin1
-- Author     : Hakan Bergon
-- Company    : NPS
-------------------------------------------------
--      File                    :
c:\My_Designs\Rangebin1\compile\DJF_1BitAd
der.vhd
-- Generated   : Mon May 20 16:38:22
2002
--      From                    :
c:\My_Designs\Rangebin1\src\DJF_1BitAdder.b
de
-- By         : Bde2Vhdl ver. 2.01
-------------------------------------------------
-- Description :
-------------------------------------------------
-- Design unit header --
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

entity DJF_1BitAdder is
  port(
      A : in std_logic;
      B : in std_logic;
      Ci : in std_logic;
      \~A\ : in std_logic;
      \~B\ : in std_logic;
      S : out std_logic);
end DJF_1BitAdder;

architecture structural of
DJF_1BitAdder is
---- Component declarations -----
component DJF_Buffer
  port (
      BufIn : in STD_LOGIC;
      BufOut : out STD_LOGIC);
end component;
component DJF_Inv_1x
  port (
      \In\ : in STD_LOGIC;
      \Out\ : out STD_LOGIC);
end component;

component DJF_PassGate_1x
  port (
      Con : in STD_LOGIC;
      ConNot : in STD_LOGIC;
      \In\ : in STD_LOGIC;
      \Out\ : out STD_LOGIC);
end component;

---- Signal declarations used on the
diagram ----
signal N1 : std_logic;
signal N16 : std_logic;
signal N17 : std_logic;
signal N2 : std_logic;
signal N20 : std_logic;

begin

---- Component instantiations  ----
DJF_Buffer_1 : DJF_Buffer
  port map(
      BufIn => N2,
      BufOut => S);

\DJF_Inv-1x_1\ : DJF_Inv_1x
  port map(
      \In\ => Ci,
      \Out\ => N17);

\DJF_Inv-1x_2\ : DJF_Inv_1x
  port map(
      \In\ => N20,
      \Out\ => N16);

\DJF_Inv-1x_3\ : DJF_Inv_1x
  port map(
      \In\ => N1,
      \Out\ => N20);

\DJF_PassGate-1x_1\:
DJF_PassGate_1x
  port map(
      Con => N17,
      ConNot => Ci,
      \In\ => N20,
      \Out\ => N2);

\DJF_PassGate-1x_2\:
DJF_PassGate_1x
  port map(
      Con => Ci,
      ConNot => N17,
      \In\ => N16,
      \Out\ => N2);

\DJF_PassGate-1x_3\:
DJF_PassGate_1x
  port map(
```

```
            Con => \~A\,
            ConNot => A,                                    end structural;
            \In\ => \~B\,
            \Out\ => N1);

        \DJF_PassGate-1x_4\:
DJF_PassGate_1x
         port map(
            Con => A,
            ConNot => \~A\,
            \In\ => B,
            \Out\ => N1);
```

APPENDIX D. VHDL CODE FOR THE SINGLE RANGE BIN

A. TOP LEVEL VHDL CODE

```
        ----------------------------------------------
        -- Title      :
        -- Design     : Rangebin1
        -- Author     : Hakan Bergon
        -- Company    : NPS
        --
        ----------------------------------------------
        --
        --File:
c:\My_Designs\Rangebin1\compile\CG_RangeB
inModulator.vhd
        -- Generated: Mon May 20 16:39:08
2002
        --From:
c:\My_Designs\Rangebin1\src\CG_RangeBinMo
dulator.bde
        -- By : Bde2Vhdl ver. 2.01
        --
        ------------------------------------------
        --
        -- Description :
        --
        ------------------------------------------------
        -- Design unit header --
        LIBRARY IEEE;
        USE IEEE.std_logic_1164.all;

        entity CG_RangeBinModulator is
          port(
            CLK : in std_logic;
            DRFM0 : in std_logic;
            DRFM1 : in std_logic;
            DRFM2 : in std_logic;
            DRFM3 : in std_logic;
            DRFM4 : in std_logic;
            Gain0 : in std_logic;
            Gain1 : in std_logic;
            Gain2 : in std_logic;
            Gain3 : in std_logic;
            I0 : in std_logic;
            I1 : in std_logic;
            I10 : in std_logic;
            I11 : in std_logic;
            I12 : in std_logic;
            I13 : in std_logic;
            I14 : in std_logic;
            I15 : in std_logic;
            I2 : in std_logic;
            I3 : in std_logic;
            I4 : in std_logic;
            I5 : in std_logic;
            I6 : in std_logic;
            I7 : in std_logic;
            I8 : in std_logic;
            I9 : in std_logic;
            IOV : in std_logic;
            Inc0 : in std_logic;
            Inc1 : in std_logic;
            Inc2 : in std_logic;
            Inc3 : in std_logic;
            Inc4 : in std_logic;
            ODVin : in std_logic;
            Oper : in std_logic;
            PRB : in std_logic;
            PSV : in std_logic;
            Q0 : in std_logic;
            Q1 : in std_logic;
            Q10 : in std_logic;
            Q11 : in std_logic;
            Q12 : in std_logic;
            Q13 : in std_logic;
            Q14 : in std_logic;
            Q15 : in std_logic;
            Q2 : in std_logic;
            Q3 : in std_logic;
            Q4 : in std_logic;
            Q5 : in std_logic;
            Q6 : in std_logic;
            Q7 : in std_logic;
            Q8 : in std_logic;
            Q9 : in std_logic;
            QOV : in std_logic;
            UNP : in std_logic;
            URB : in std_logic;
            \~I0\ : in std_logic;
            \~I10\ : in std_logic;
            \~I11\ : in std_logic;
            \~I12\ : in std_logic;
            \~I13\ : in std_logic;
            \~I14\ : in std_logic;
            \~I15\ : in std_logic;
            \~I1\ : in std_logic;
            \~I2\ : in std_logic;
            \~I3\ : in std_logic;
            \~I4\ : in std_logic;
            \~I5\ : in std_logic;
            \~I6\ : in std_logic;
            \~I7\ : in std_logic;
            \~I8\ : in std_logic;
            \~I9\ : in std_logic;
```

```
\~Q0\ : in std_logic;
\~Q10\ : in std_logic;
\~Q11\ : in std_logic;
\~Q12\ : in std_logic;
\~Q13\ : in std_logic;
\~Q14\ : in std_logic;
\~Q15\ : in std_logic;
\~Q1\ : in std_logic;
\~Q2\ : in std_logic;
\~Q3\ : in std_logic;
\~Q4\ : in std_logic;
\~Q5\ : in std_logic;
\~Q6\ : in std_logic;
\~Q7\ : in std_logic;
\~Q8\ : in std_logic;
\~Q9\ : in std_logic;
IS0 : out std_logic;
IS1 : out std_logic;
IS10 : out std_logic;
IS11 : out std_logic;
IS12 : out std_logic;
IS13 : out std_logic;
IS14 : out std_logic;
IS15 : out std_logic;
IS2 : out std_logic;
IS3 : out std_logic;
IS4 : out std_logic;
IS5 : out std_logic;
IS6 : out std_logic;
IS7 : out std_logic;
IS8 : out std_logic;
IS9 : out std_logic;
ISOV : out std_logic;
ODVout : out std_logic;
QS0 : out std_logic;
QS1 : out std_logic;
QS10 : out std_logic;
QS11 : out std_logic;
QS12 : out std_logic;
QS13 : out std_logic;
QS14 : out std_logic;
QS15 : out std_logic;
QS2 : out std_logic;
QS3 : out std_logic;
QS4 : out std_logic;
QS5 : out std_logic;
QS6 : out std_logic;
QS7 : out std_logic;
QS8 : out std_logic;
QS9 : out std_logic;
QSOV : out std_logic;
\~IS0\ : out std_logic;
\~IS10\ : out std_logic;
\~IS11\ : out std_logic;
\~IS12\ : out std_logic;
\~IS13\ : out std_logic;
\~IS14\ : out std_logic;
\~IS15\ : out std_logic;
\~IS1\ : out std_logic;
\~IS2\ : out std_logic;
\~IS3\ : out std_logic;
\~IS4\ : out std_logic;
\~IS5\ : out std_logic;
\~IS6\ : out std_logic;
\~IS7\ : out std_logic;
\~IS8\ : out std_logic;
\~IS9\ : out std_logic;
\~QS0\ : out std_logic;
\~QS10\ : out std_logic;
\~QS11\ : out std_logic;
\~QS12\ : out std_logic;
\~QS13\ : out std_logic;
\~QS14\ : out std_logic;
\~QS15\ : out std_logic;
\~QS1\ : out std_logic;
\~QS2\ : out std_logic;
\~QS3\ : out std_logic;
\~QS4\ : out std_logic;
\~QS5\ : out std_logic;
\~QS6\ : out std_logic;
\~QS7\ : out std_logic;
\~QS8\ : out std_logic;
\~QS9\ : out std_logic
  );
end CG_RangeBinModulator;

architecture structural of
CG_RangeBinModulator is

---- Component declarations -----

component CG_5bitAdder_1x
  port (
    A0 : in STD_LOGIC;
    A1 : in STD_LOGIC;
    A2 : in STD_LOGIC;
    A3 : in STD_LOGIC;
    A4 : in STD_LOGIC;
    B0 : in STD_LOGIC;
    B1 : in STD_LOGIC;
    B2 : in STD_LOGIC;
    B3 : in STD_LOGIC;
    B4 : in STD_LOGIC;
    \~A0\ : in STD_LOGIC;
    \~A1\ : in STD_LOGIC;
    \~A2\ : in STD_LOGIC;
    \~A3\ : in STD_LOGIC;
    \~A4\ : in STD_LOGIC;
    \~B0\ : in STD_LOGIC;
    \~B1\ : in STD_LOGIC;
    \~B2\ : in STD_LOGIC;
    \~B3\ : in STD_LOGIC;
```

```
      \~B4\ : in STD_LOGIC;
      S0 : out STD_LOGIC;
      S1 : out STD_LOGIC;
      S2 : out STD_LOGIC;
      S3 : out STD_LOGIC;
      S4 : out STD_LOGIC
  );
end component;
component CG_Clock
  port (
      CLK : in STD_LOGIC;
      CLK1 : out STD_LOGIC;
      CLK2 : out STD_LOGIC
  );
end component;
component CG_DMSFFPGreg17_1x
  port (
      CLK : in STD_LOGIC;
      CLR : in STD_LOGIC;
      D0 : in STD_LOGIC;
      D1 : in STD_LOGIC;
      D10 : in STD_LOGIC;
      D11 : in STD_LOGIC;
      D12 : in STD_LOGIC;
      D13 : in STD_LOGIC;
      D14 : in STD_LOGIC;
      D15 : in STD_LOGIC;
      D16 : in STD_LOGIC;
      D2 : in STD_LOGIC;
      D3 : in STD_LOGIC;
      D4 : in STD_LOGIC;
      D5 : in STD_LOGIC;
      D6 : in STD_LOGIC;
      D7 : in STD_LOGIC;
      D8 : in STD_LOGIC;
      D9 : in STD_LOGIC;
      LD : in STD_LOGIC;
      Q0 : out STD_LOGIC;
      Q1 : out STD_LOGIC;
      Q10 : out STD_LOGIC;
      Q11 : out STD_LOGIC;
      Q12 : out STD_LOGIC;
      Q13 : out STD_LOGIC;
      Q14 : out STD_LOGIC;
      Q15 : out STD_LOGIC;
      Q16 : out STD_LOGIC;
      Q2 : out STD_LOGIC;
      Q3 : out STD_LOGIC;
      Q4 : out STD_LOGIC;
      Q5 : out STD_LOGIC;
      Q6 : out STD_LOGIC;
      Q7 : out STD_LOGIC;
      Q8 : out STD_LOGIC;
      Q9 : out STD_LOGIC;
      \~Q0\ : out STD_LOGIC;
      \~Q10\ : out STD_LOGIC;
      \~Q11\ : out STD_LOGIC;
      \~Q12\ : out STD_LOGIC;
      \~Q13\ : out STD_LOGIC;
      \~Q14\ : out STD_LOGIC;
      \~Q15\ : out STD_LOGIC;
      \~Q16\ : out STD_LOGIC;
      \~Q1\ : out STD_LOGIC;
      \~Q2\ : out STD_LOGIC;
      \~Q3\ : out STD_LOGIC;
      \~Q4\ : out STD_LOGIC;
      \~Q5\ : out STD_LOGIC;
      \~Q6\ : out STD_LOGIC;
      \~Q7\ : out STD_LOGIC;
      \~Q8\ : out STD_LOGIC;
      \~Q9\ : out STD_LOGIC
  );
end component;
component CG_DMSFFPGreg5_1x
  port (
      CLK : in STD_LOGIC;
      D0 : in STD_LOGIC;
      D1 : in STD_LOGIC;
      D2 : in STD_LOGIC;
      D3 : in STD_LOGIC;
      D4 : in STD_LOGIC;
      LD : in STD_LOGIC;
      Q0 : out STD_LOGIC;
      Q1 : out STD_LOGIC;
      Q2 : out STD_LOGIC;
      Q3 : out STD_LOGIC;
      Q4 : out STD_LOGIC;
      \~Q0\ : out STD_LOGIC;
      \~Q1\ : out STD_LOGIC;
      \~Q2\ : out STD_LOGIC;
      \~Q3\ : out STD_LOGIC;
      \~Q4\ : out STD_LOGIC
  );
end component;
component CG_DMSFFPGreg8_1x
  port (
      CLK : in STD_LOGIC;
      D0 : in STD_LOGIC;
      D1 : in STD_LOGIC;
      D2 : in STD_LOGIC;
      D3 : in STD_LOGIC;
      D4 : in STD_LOGIC;
      D5 : in STD_LOGIC;
      D6 : in STD_LOGIC;
      D7 : in STD_LOGIC;
      LD : in STD_LOGIC;
      Q0 : out STD_LOGIC;
      Q1 : out STD_LOGIC;
      Q2 : out STD_LOGIC;
      Q3 : out STD_LOGIC;
      Q4 : out STD_LOGIC;
      Q5 : out STD_LOGIC;
```

```
            Q6 : out STD_LOGIC;
            Q7 : out STD_LOGIC;
            \~Q0\ : out STD_LOGIC;
            \~Q1\ : out STD_LOGIC;
            \~Q2\ : out STD_LOGIC;
            \~Q3\ : out STD_LOGIC;
            \~Q4\ : out STD_LOGIC;
            \~Q5\ : out STD_LOGIC;
            \~Q6\ : out STD_LOGIC;
            \~Q7\ : out STD_LOGIC
        );
    end component;
    component
CG_DMSFFPG_CLRreg13_1x
        port (
            CLK : in STD_LOGIC;
            CLR : in STD_LOGIC;
            D0 : in STD_LOGIC;
            D1 : in STD_LOGIC;
            D10 : in STD_LOGIC;
            D11 : in STD_LOGIC;
            D12 : in STD_LOGIC;
            D2 : in STD_LOGIC;
            D3 : in STD_LOGIC;
            D4 : in STD_LOGIC;
            D5 : in STD_LOGIC;
            D6 : in STD_LOGIC;
            D7 : in STD_LOGIC;
            D8 : in STD_LOGIC;
            D9 : in STD_LOGIC;
            LD : in STD_LOGIC;
            Q0 : out STD_LOGIC;
            Q1 : out STD_LOGIC;
            Q10 : out STD_LOGIC;
            Q11 : out STD_LOGIC;
            Q12 : out STD_LOGIC;
            Q2 : out STD_LOGIC;
            Q3 : out STD_LOGIC;
            Q4 : out STD_LOGIC;
            Q5 : out STD_LOGIC;
            Q6 : out STD_LOGIC;
            Q7 : out STD_LOGIC;
            Q8 : out STD_LOGIC;
            Q9 : out STD_LOGIC;
            \~Q0\ : out STD_LOGIC;
            \~Q10\ : out STD_LOGIC;
            \~Q11\ : out STD_LOGIC;
            \~Q12\ : out STD_LOGIC;
            \~Q1\ : out STD_LOGIC;
            \~Q2\ : out STD_LOGIC;
            \~Q3\ : out STD_LOGIC;
            \~Q4\ : out STD_LOGIC;
            \~Q5\ : out STD_LOGIC;
            \~Q6\ : out STD_LOGIC;
            \~Q7\ : out STD_LOGIC;
            \~Q8\ : out STD_LOGIC;
            \~Q9\ : out STD_LOGIC
        );
    end component;
    component CG_Gain_Shifter_1x
        port (
            Gain0 : in STD_LOGIC;
            Gain1 : in STD_LOGIC;
            Gain2 : in STD_LOGIC;
            Gain3 : in STD_LOGIC;
            I0 : in STD_LOGIC;
            I1 : in STD_LOGIC;
            I2 : in STD_LOGIC;
            I3 : in STD_LOGIC;
            I4 : in STD_LOGIC;
            I5 : in STD_LOGIC;
            I6 : in STD_LOGIC;
            I7 : in STD_LOGIC;
            \~Gain0\ : in STD_LOGIC;
            \~Gain1\ : in STD_LOGIC;
            \~Gain2\ : in STD_LOGIC;
            \~Gain3\ : in STD_LOGIC;
            O10 : out STD_LOGIC;
            O11 : out STD_LOGIC;
            O12 : out STD_LOGIC;
            O13 : out STD_LOGIC;
            O14 : out STD_LOGIC;
            O15 : out STD_LOGIC;
            O16 : out STD_LOGIC;
            O17 : out STD_LOGIC;
            O5 : out STD_LOGIC;
            O6 : out STD_LOGIC;
            O7 : out STD_LOGIC;
            O8 : out STD_LOGIC;
            O9 : out STD_LOGIC
        );
    end component;
    component CG_RangeBinControl
        port (
            CLK : in STD_LOGIC;
            ODVin : in STD_LOGIC;
            Oper : in STD_LOGIC;
            PSV : in STD_LOGIC;
            URB : in STD_LOGIC;
            CLR13 : out STD_LOGIC;
            CLR17 : out STD_LOGIC;
            ODVout : out STD_LOGIC
        );
    end component;
    component DJF_16BitAdder
        port (
            A0 : in STD_LOGIC;
            A1 : in STD_LOGIC;
            A10 : in STD_LOGIC;
            A11 : in STD_LOGIC;
            A12 : in STD_LOGIC;
            A13 : in STD_LOGIC;
```

```
A14 : in STD_LOGIC;
A15 : in STD_LOGIC;
A2 : in STD_LOGIC;
A3 : in STD_LOGIC;
A4 : in STD_LOGIC;
A5 : in STD_LOGIC;
A6 : in STD_LOGIC;
A7 : in STD_LOGIC;
A8 : in STD_LOGIC;
A9 : in STD_LOGIC;
B0 : in STD_LOGIC;
B1 : in STD_LOGIC;
B10 : in STD_LOGIC;
B11 : in STD_LOGIC;
B12 : in STD_LOGIC;
B13 : in STD_LOGIC;
B14 : in STD_LOGIC;
B15 : in STD_LOGIC;
B2 : in STD_LOGIC;
B3 : in STD_LOGIC;
B4 : in STD_LOGIC;
B5 : in STD_LOGIC;
B6 : in STD_LOGIC;
B7 : in STD_LOGIC;
B8 : in STD_LOGIC;
B9 : in STD_LOGIC;
C0 : in STD_LOGIC;
OFin : in STD_LOGIC;
\~A0\ : in STD_LOGIC;
\~A10\ : in STD_LOGIC;
\~A11\ : in STD_LOGIC;
\~A12\ : in STD_LOGIC;
\~A13\ : in STD_LOGIC;
\~A14\ : in STD_LOGIC;
\~A15\ : in STD_LOGIC;
\~A1\ : in STD_LOGIC;
\~A2\ : in STD_LOGIC;
\~A3\ : in STD_LOGIC;
\~A4\ : in STD_LOGIC;
\~A5\ : in STD_LOGIC;
\~A6\ : in STD_LOGIC;
\~A7\ : in STD_LOGIC;
\~A8\ : in STD_LOGIC;
\~A9\ : in STD_LOGIC;
\~B0\ : in STD_LOGIC;
\~B10\ : in STD_LOGIC;
\~B11\ : in STD_LOGIC;
\~B12\ : in STD_LOGIC;
\~B13\ : in STD_LOGIC;
\~B14\ : in STD_LOGIC;
\~B15\ : in STD_LOGIC;
\~B1\ : in STD_LOGIC;
\~B2\ : in STD_LOGIC;
\~B3\ : in STD_LOGIC;
\~B4\ : in STD_LOGIC;
\~B5\ : in STD_LOGIC;
\~B6\ : in STD_LOGIC;
\~B7\ : in STD_LOGIC;
\~B8\ : in STD_LOGIC;
\~B9\ : in STD_LOGIC;
C16 : out STD_LOGIC;
OFout : out STD_LOGIC;
S0 : out STD_LOGIC;
S1 : out STD_LOGIC;
S10 : out STD_LOGIC;
S11 : out STD_LOGIC;
S12 : out STD_LOGIC;
S13 : out STD_LOGIC;
S14 : out STD_LOGIC;
S15 : out STD_LOGIC;
S2 : out STD_LOGIC;
S3 : out STD_LOGIC;
S4 : out STD_LOGIC;
S5 : out STD_LOGIC;
S6 : out STD_LOGIC;
S7 : out STD_LOGIC;
S8 : out STD_LOGIC;
S9 : out STD_LOGIC
);
end component;
component Gnd
port (
Gnd : out STD_LOGIC
);
end component;
component KMK_LUT8
port (
A0 : in STD_LOGIC;
A1 : in STD_LOGIC;
A2 : in STD_LOGIC;
A3 : in STD_LOGIC;
A4 : in STD_LOGIC;
COS0 : out STD_LOGIC;
COS1 : out STD_LOGIC;
COS2 : out STD_LOGIC;
COS3 : out STD_LOGIC;
COS4 : out STD_LOGIC;
COS5 : out STD_LOGIC;
COS6 : out STD_LOGIC;
COS7 : out STD_LOGIC;
SIN0 : out STD_LOGIC;
SIN1 : out STD_LOGIC;
SIN2 : out STD_LOGIC;
SIN3 : out STD_LOGIC;
SIN4 : out STD_LOGIC;
SIN5 : out STD_LOGIC;
SIN6 : out STD_LOGIC;
SIN7 : out STD_LOGIC
);
end component;
```

```
        ---- Signal declarations used on the
diagram ----

        signal LogGnd : std_logic;
        signal N1 : std_logic;
        signal N10 : std_logic;
        signal N100 : std_logic;
        signal N101 : std_logic;
        signal N102 : std_logic;
        signal N103 : std_logic;
        signal N104 : std_logic;
        signal N105 : std_logic;
        signal N106 : std_logic;
        signal N107 : std_logic;
        signal N108 : std_logic;
        signal N109 : std_logic;
        signal N11 : std_logic;
        signal N110 : std_logic;
        signal N111 : std_logic;
        signal N112 : std_logic;
        signal N113 : std_logic;
        signal N114 : std_logic;
        signal N115 : std_logic;
        signal N116 : std_logic;
        signal N117 : std_logic;
        signal N118 : std_logic;
        signal N119 : std_logic;
        signal N12 : std_logic;
        signal N120 : std_logic;
        signal N121 : std_logic;
        signal N122 : std_logic;
        signal N123 : std_logic;
        signal N124 : std_logic;
        signal N125 : std_logic;
        signal N126 : std_logic;
        signal N127 : std_logic;
        signal N128 : std_logic;
        signal N129 : std_logic;
        signal N13 : std_logic;
        signal N130 : std_logic;
        signal N131 : std_logic;
        signal N132 : std_logic;
        signal N133 : std_logic;
        signal N134 : std_logic;
        signal N135 : std_logic;
        signal N136 : std_logic;
        signal N137 : std_logic;
        signal N138 : std_logic;
        signal N139 : std_logic;
        signal N14 : std_logic;
        signal N140 : std_logic;
        signal N141 : std_logic;
        signal N142 : std_logic;
        signal N143 : std_logic;
        signal N144 : std_logic;
        signal N145 : std_logic;
        signal N146 : std_logic;
        signal N147 : std_logic;
        signal N148 : std_logic;
        signal N149 : std_logic;
        signal N15 : std_logic;
        signal N150 : std_logic;
        signal N151 : std_logic;
        signal N152 : std_logic;
        signal N16 : std_logic;
        signal N17 : std_logic;
        signal N18 : std_logic;
        signal N187 : std_logic;
        signal N188 : std_logic;
        signal N189 : std_logic;
        signal N19 : std_logic;
        signal N190 : std_logic;
        signal N191 : std_logic;
        signal N192 : std_logic;
        signal N193 : std_logic;
        signal N194 : std_logic;
        signal N195 : std_logic;
        signal N196 : std_logic;
        signal N197 : std_logic;
        signal N198 : std_logic;
        signal N199 : std_logic;
        signal N2 : std_logic;
        signal N20 : std_logic;
        signal N200 : std_logic;
        signal N201 : std_logic;
        signal N202 : std_logic;
        signal N203 : std_logic;
        signal N204 : std_logic;
        signal N205 : std_logic;
        signal N206 : std_logic;
        signal N207 : std_logic;
        signal N208 : std_logic;
        signal N209 : std_logic;
        signal N21 : std_logic;
        signal N210 : std_logic;
        signal N211 : std_logic;
        signal N212 : std_logic;
        signal N213 : std_logic;
        signal N214 : std_logic;
        signal N215 : std_logic;
        signal N216 : std_logic;
        signal N217 : std_logic;
        signal N218 : std_logic;
        signal N22 : std_logic;
        signal N220 : std_logic;
        signal N221 : std_logic;
        signal N222 : std_logic;
        signal N223 : std_logic;
        signal N224 : std_logic;
        signal N225 : std_logic;
        signal N226 : std_logic;
        signal N227 : std_logic;
```

```
signal N228 : std_logic;
signal N229 : std_logic;
signal N23 : std_logic;
signal N230 : std_logic;
signal N231 : std_logic;
signal N232 : std_logic;
signal N24 : std_logic;
signal N25 : std_logic;
signal N26 : std_logic;
signal N263 : std_logic;
signal N264 : std_logic;
signal N265 : std_logic;
signal N266 : std_logic;
signal N267 : std_logic;
signal N268 : std_logic;
signal N269 : std_logic;
signal N27 : std_logic;
signal N270 : std_logic;
signal N271 : std_logic;
signal N272 : std_logic;
signal N273 : std_logic;
signal N274 : std_logic;
signal N275 : std_logic;
signal N277 : std_logic;
signal N278 : std_logic;
signal N279 : std_logic;
signal N28 : std_logic;
signal N280 : std_logic;
signal N281 : std_logic;
signal N282 : std_logic;
signal N283 : std_logic;
signal N284 : std_logic;
signal N285 : std_logic;
signal N286 : std_logic;
signal N287 : std_logic;
signal N288 : std_logic;
signal N289 : std_logic;
signal N29 : std_logic;
signal N3 : std_logic;
signal N30 : std_logic;
signal N306 : std_logic;
signal N31 : std_logic;
signal N32 : std_logic;
signal N33 : std_logic;
signal N339 : std_logic;
signal N34 : std_logic;
signal N344 : std_logic;
signal N345 : std_logic;
signal N346 : std_logic;
signal N347 : std_logic;
signal N348 : std_logic;
signal N349 : std_logic;
signal N35 : std_logic;
signal N350 : std_logic;
signal N351 : std_logic;
signal N352 : std_logic;
signal N36 : std_logic;
signal N37 : std_logic;
signal N38 : std_logic;
signal N39 : std_logic;
signal N4 : std_logic;
signal N40 : std_logic;
signal N41 : std_logic;
signal N42 : std_logic;
signal N43 : std_logic;
signal N44 : std_logic;
signal N45 : std_logic;
signal N46 : std_logic;
signal N47 : std_logic;
signal N48 : std_logic;
signal N49 : std_logic;
signal N5 : std_logic;
signal N50 : std_logic;
signal N51 : std_logic;
signal N52 : std_logic;
signal N53 : std_logic;
signal N54 : std_logic;
signal N55 : std_logic;
signal N56 : std_logic;
signal N57 : std_logic;
signal N58 : std_logic;
signal N59 : std_logic;
signal N6 : std_logic;
signal N60 : std_logic;
signal N61 : std_logic;
signal N62 : std_logic;
signal N63 : std_logic;
signal N64 : std_logic;
signal N65 : std_logic;
signal N66 : std_logic;
signal N67 : std_logic;
signal N68 : std_logic;
signal N69 : std_logic;
signal N7 : std_logic;
signal N70 : std_logic;
signal N71 : std_logic;
signal N72 : std_logic;
signal N73 : std_logic;
signal N74 : std_logic;
signal N75 : std_logic;
signal N76 : std_logic;
signal N77 : std_logic;
signal N78 : std_logic;
signal N79 : std_logic;
signal N8 : std_logic;
signal N80 : std_logic;
signal N81 : std_logic;
signal N82 : std_logic;
signal N83 : std_logic;
signal N84 : std_logic;
signal N85 : std_logic;
signal N86 : std_logic;
```

```
        signal N87 : std_logic;
        signal N88 : std_logic;
        signal N89 : std_logic;
        signal N9 : std_logic;
        signal N90 : std_logic;
        signal N91 : std_logic;
        signal N92 : std_logic;
        signal N93 : std_logic;
        signal N94 : std_logic;
        signal N95 : std_logic;
        signal N96 : std_logic;
        signal N97 : std_logic;
        signal N98 : std_logic;
        signal N99 : std_logic;

        begin

        ----  Component instantiations  ----

        CG_Clock_1 : CG_Clock
         port map(
            CLK => CLK,
            CLK1 => N152,
            CLK2 => N43
         );

        CG_RangeBinControl_1                    :
CG_RangeBinControl
         port map(
            CLK => N43,
            CLR13 => N44,
            CLR17 => N339,
            ODVin => ODVin,
            ODVout => ODVout,
            Oper => Oper,
            PSV => PSV,
            URB => N352
         );

        DJF_16BitAdder_1 : DJF_16BitAdder
         port map(
            A0 => Q0,
            A1 => Q1,
            A10 => Q10,
            A11 => Q11,
            A12 => Q12,
            A13 => Q13,
            A14 => Q14,
            A15 => Q15,
            A2 => Q2,
            A3 => Q3,
            A4 => Q4,
            A5 => Q5,
            A6 => Q6,
            A7 => Q7,
            A8 => Q8,
            A9 => Q9,
            B0 => N232,
            B1 => N231,
            B10 => N222,
            B11 => N221,
            B12 => N220,
            B13 => N220,
            B14 => N220,
            B15 => N220,
            B2 => N230,
            B3 => N229,
            B4 => N228,
            B5 => N227,
            B6 => N226,
            B7 => N225,
            B8 => N224,
            B9 => N223,
            C0 => LogGnd,
            C16 => N11,
            OFin => QOV,
            OFout => N73,
            S0 => N203,
            S1 => N204,
            S10 => N210,
            S11 => N209,
            S12 => N208,
            S13 => N207,
            S14 => N206,
            S15 => N205,
            S2 => N218,
            S3 => N217,
            S4 => N216,
            S5 => N215,
            S6 => N214,
            S7 => N213,
            S8 => N212,
            S9 => N211,
            \~A0\ => \~Q0\,
            \~A10\ => \~Q10\,
            \~A11\ => \~Q11\,
            \~A12\ => \~Q12\,
            \~A13\ => \~Q13\,
            \~A14\ => \~Q14\,
            \~A15\ => \~Q15\,
            \~A1\ => \~Q1\,
            \~A2\ => \~Q2\,
            \~A3\ => \~Q3\,
            \~A4\ => \~Q4\,
            \~A5\ => \~Q5\,
            \~A6\ => \~Q6\,
            \~A7\ => \~Q7\,
            \~A8\ => \~Q8\,
            \~A9\ => \~Q9\,
            \~B0\ => N42,
            \~B10\ => N32,
            \~B11\ => N31,
```

```
    \~B12\ => N30,
    \~B13\ => N30,
    \~B14\ => N30,
    \~B15\ => N30,
    \~B1\ => N41,
    \~B2\ => N40,
    \~B3\ => N39,
    \~B4\ => N38,
    \~B5\ => N37,
    \~B6\ => N36,
    \~B7\ => N35,
    \~B8\ => N34,
    \~B9\ => N33
 );

DJF_16BitAdder_2 : DJF_16BitAdder
  port map(
    A0 => I0,
    A1 => I1,
    A10 => I10,
    A11 => I11,
    A12 => I12,
    A13 => I13,
    A14 => I14,
    A15 => I15,
    A2 => I2,
    A3 => I3,
    A4 => I4,
    A5 => I5,
    A6 => I6,
    A7 => I7,
    A8 => I8,
    A9 => I9,
    B0 => N288,
    B1 => N287,
    B10 => N278,
    B11 => N277,
    B12 => N275,
    B13 => N275,
    B14 => N275,
    B15 => N275,
    B2 => N286,
    B3 => N285,
    B4 => N284,
    B5 => N283,
    B6 => N282,
    B7 => N281,
    B8 => N280,
    B9 => N279,
    C0 => N9,
    C16 => N10,
    OFin => IOV,
    OFout => N306,
    S0 => N187,
    S1 => N188,
    S10 => N194,
    S11 => N193,
    S12 => N192,
    S13 => N191,
    S14 => N190,
    S15 => N189,
    S2 => N202,
    S3 => N201,
    S4 => N200,
    S5 => N199,
    S6 => N198,
    S7 => N197,
    S8 => N196,
    S9 => N195,
    \~A0\ => \~I0\,
    \~A10\ => \~I10\,
    \~A11\ => \~I11\,
    \~A12\ => \~I12\,
    \~A13\ => \~I13\,
    \~A14\ => \~I14\,
    \~A15\ => \~I15\,
    \~A1\ => \~I1\,
    \~A2\ => \~I2\,
    \~A3\ => \~I3\,
    \~A4\ => \~I4\,
    \~A5\ => \~I5\,
    \~A6\ => \~I6\,
    \~A7\ => \~I7\,
    \~A8\ => \~I8\,
    \~A9\ => \~I9\,
    \~B0\ => N274,
    \~B10\ => N264,
    \~B11\ => N263,
    \~B12\ => N289,
    \~B13\ => N289,
    \~B14\ => N289,
    \~B15\ => N289,
    \~B1\ => N273,
    \~B2\ => N272,
    \~B3\ => N271,
    \~B4\ => N270,
    \~B5\ => N269,
    \~B6\ => N268,
    \~B7\ => N267,
    \~B8\ => N266,
    \~B9\ => N265
  );

Gnd_1 : Gnd
  port map(
    Gnd => LogGnd
  );

Gnd_2 : Gnd
  port map(
    Gnd => N9
  );
```

```
        KMK_LUT8_1 : KMK_LUT8
         port map(
            A0 => N79,
            A1 => N78,
            A2 => N77,
            A3 => N76,
            A4 => N75,
            COS0 => N92,
            COS1 => N91,
            COS2 => N90,
            COS3 => N89,
            COS4 => N88,
            COS5 => N87,
            COS6 => N86,
            COS7 => N85,
            SIN0 => N8,
            SIN1 => N7,
            SIN2 => N6,
            SIN3 => N5,
            SIN4 => N4,
            SIN5 => N3,
            SIN6 => N2,
            SIN7 => N1
         );

        \CG_5bitAdder-1x_1\ :
CG_5bitAdder_1x
         port map(
            A0 => N120,
            A1 => N129,
            A2 => N127,
            A3 => N125,
            A4 => N123,
            B0 => N109,
            B1 => N111,
            B2 => N113,
            B3 => N115,
            B4 => N117,
            S0 => N139,
            S1 => N140,
            S2 => N141,
            S3 => N142,
            S4 => N143,
            \~A0\ => N121,
            \~A1\ => N128,
            \~A2\ => N126,
            \~A3\ => N124,
            \~A4\ => N122,
            \~B0\ => N110,
            \~B1\ => N112,
            \~B2\ => N114,
            \~B3\ => N116,
            \~B4\ => N118
         );

                \CG_DMSFFPG_CLRreg13-1x_1\ :
CG_DMSFFPG_CLRreg13_1x
            port map(
               CLK => N43,
               CLR => N44,
               D0 => N70,
               D1 => N69,
               D10 => N60,
               D11 => N59,
               D12 => N58,
               D2 => N68,
               D3 => N67,
               D4 => N66,
               D5 => N65,
               D6 => N64,
               D7 => N63,
               D8 => N62,
               D9 => N61,
               LD => Oper,
               Q0 => N232,
               Q1 => N231,
               Q10 => N222,
               Q11 => N221,
               Q12 => N220,
               Q2 => N230,
               Q3 => N229,
               Q4 => N228,
               Q5 => N227,
               Q6 => N226,
               Q7 => N225,
               Q8 => N224,
               Q9 => N223,
               \~Q0\ => N42,
               \~Q10\ => N32,
               \~Q11\ => N31,
               \~Q12\ => N30,
               \~Q1\ => N41,
               \~Q2\ => N40,
               \~Q3\ => N39,
               \~Q4\ => N38,
               \~Q5\ => N37,
               \~Q6\ => N36,
               \~Q7\ => N35,
               \~Q8\ => N34,
               \~Q9\ => N33
            );

            \CG_DMSFFPG_CLRreg13-1x_2\ :
CG_DMSFFPG_CLRreg13_1x
            port map(
               CLK => N152,
               CLR => N44,
               D0 => N57,
               D1 => N56,
               D10 => N47,
               D11 => N46,
```

```
            D12 => N45,
            D2 => N55,
            D3 => N54,
            D4 => N53,
            D5 => N52,
            D6 => N51,
            D7 => N50,
            D8 => N49,
            D9 => N48,
            LD => Oper,
            Q0 => N288,
            Q1 => N287,
            Q10 => N278,
            Q11 => N277,
            Q12 => N275,
            Q2 => N286,
            Q3 => N285,
            Q4 => N284,
            Q5 => N283,
            Q6 => N282,
            Q7 => N281,
            Q8 => N280,
            Q9 => N279,
            \~Q0\ => N274,
            \~Q10\ => N264,
            \~Q11\ => N263,
            \~Q12\ => N289,
            \~Q1\ => N273,
            \~Q2\ => N272,
            \~Q3\ => N271,
            \~Q4\ => N270,
            \~Q5\ => N269,
            \~Q6\ => N268,
            \~Q7\ => N267,
            \~Q8\ => N266,
            \~Q9\ => N265
        );

    \CG_DMSFFPGreg17-1x_1\ : CG_DMSFFPGreg17_1x
        port map(
            CLK => N152,
            CLR => N339,
            D0 => N306,
            D1 => N187,
            D10 => N195,
            D11 => N194,
            D12 => N193,
            D13 => N192,
            D14 => N191,
            D15 => N190,
            D16 => N189,
            D2 => N188,
            D3 => N202,
            D4 => N201,
            D5 => N200,
            D6 => N199,
            D7 => N198,
            D8 => N197,
            D9 => N196,
            LD => Oper,
            Q0 => ISOV,
            Q1 => IS0,
            Q10 => IS9,
            Q11 => IS10,
            Q12 => IS11,
            Q13 => IS12,
            Q14 => IS13,
            Q15 => IS14,
            Q16 => IS15,
            Q2 => IS1,
            Q3 => IS2,
            Q4 => IS3,
            Q5 => IS4,
            Q6 => IS5,
            Q7 => IS6,
            Q8 => IS7,
            Q9 => IS8,
            \~Q0\ => N151,
            \~Q10\ => \~IS9\,
            \~Q11\ => \~IS10\,
            \~Q12\ => \~IS11\,
            \~Q13\ => \~IS12\,
            \~Q14\ => \~IS13\,
            \~Q15\ => \~IS14\,
            \~Q16\ => \~IS15\,
            \~Q1\ => \~IS0\,
            \~Q2\ => \~IS1\,
            \~Q3\ => \~IS2\,
            \~Q4\ => \~IS3\,
            \~Q5\ => \~IS4\,
            \~Q6\ => \~IS5\,
            \~Q7\ => \~IS6\,
            \~Q8\ => \~IS7\,
            \~Q9\ => \~IS8\
        );

    \CG_DMSFFPGreg17-1x_2\ : CG_DMSFFPGreg17_1x
        port map(
            CLK => N43,
            CLR => N339,
            D0 => N73,
            D1 => N203,
            D10 => N211,
            D11 => N210,
            D12 => N209,
            D13 => N208,
            D14 => N207,
            D15 => N206,
            D16 => N205,
            D2 => N204,
```

```
            D3 => N218,
            D4 => N217,
            D5 => N216,
            D6 => N215,
            D7 => N214,
            D8 => N213,
            D9 => N212,
            LD => Oper,
            Q0 => QSOV,
            Q1 => QS0,
            Q10 => QS9,
            Q11 => QS10,
            Q12 => QS11,
            Q13 => QS12,
            Q14 => QS13,
            Q15 => QS14,
            Q16 => QS15,
            Q2 => QS1,
            Q3 => QS2,
            Q4 => QS3,
            Q5 => QS4,
            Q6 => QS5,
            Q7 => QS6,
            Q8 => QS7,
            Q9 => QS8,
            \~Q0\ => N150,
            \~Q10\ => \~QS9\,
            \~Q11\ => \~QS10\,
            \~Q12\ => \~QS11\,
            \~Q13\ => \~QS12\,
            \~Q14\ => \~QS13\,
            \~Q15\ => \~QS14\,
            \~Q16\ => \~QS15\,
            \~Q1\ => \~QS0\,
            \~Q2\ => \~QS1\,
            \~Q3\ => \~QS2\,
            \~Q4\ => \~QS3\,
            \~Q5\ => \~QS4\,
            \~Q6\ => \~QS5\,
            \~Q7\ => \~QS6\,
            \~Q8\ => \~QS7\,
            \~Q9\ => \~QS8\
        );

    \CG_DMSFFPGreg5-1x_1\ : CG_DMSFFPGreg5_1x
        port map(
            CLK => N43,
            D0 => N139,
            D1 => N140,
            D2 => N141,
            D3 => N142,
            D4 => N143,
            LD => Oper,
            Q0 => N79,
            Q1 => N78,
            Q2 => N77,
            Q3 => N76,
            Q4 => N75,
            \~Q0\ => N149,
            \~Q1\ => N148,
            \~Q2\ => N147,
            \~Q3\ => N146,
            \~Q4\ => N145
        );

    \CG_DMSFFPGreg5-1x_2\ : CG_DMSFFPGreg5_1x
        port map(
            CLK => N152,
            D0 => DRFM0,
            D1 => DRFM1,
            D2 => DRFM2,
            D3 => DRFM3,
            D4 => DRFM4,
            LD => Oper,
            Q0 => N120,
            Q1 => N129,
            Q2 => N127,
            Q3 => N125,
            Q4 => N123,
            \~Q0\ => N121,
            \~Q1\ => N128,
            \~Q2\ => N126,
            \~Q3\ => N124,
            \~Q4\ => N122
        );

    \CG_DMSFFPGreg5-1x_3\ : CG_DMSFFPGreg5_1x
        port map(
            CLK => N152,
            D0 => N134,
            D1 => N135,
            D2 => N136,
            D3 => N137,
            D4 => N138,
            LD => UNP,
            Q0 => N109,
            Q1 => N111,
            Q2 => N113,
            Q3 => N115,
            Q4 => N117,
            \~Q0\ => N110,
            \~Q1\ => N112,
            \~Q2\ => N114,
            \~Q3\ => N116,
            \~Q4\ => N118
        );

    \CG_DMSFFPGreg5-1x_4\ : CG_DMSFFPGreg5_1x
```

```
        port map(
            CLK => N43,
            D0 => Inc0,
            D1 => Inc1,
            D2 => Inc2,
            D3 => Inc3,
            D4 => Inc4,
            LD => PRB,
            Q0 => N134,
            Q1 => N135,
            Q2 => N136,
            Q3 => N137,
            Q4 => N138,
            \~Q0\ => N144,
            \~Q1\ => N133,
            \~Q2\ => N132,
            \~Q3\ => N131,
            \~Q4\ => N130
        );

    \CG_DMSFFPGreg5-1x_5\ : CG_DMSFFPGreg5_1x
        port map(
            CLK => N43,
            D0 => Gain0,
            D1 => Gain1,
            D2 => Gain2,
            D3 => Gain3,
            D4 => URB,
            LD => PRB,
            Q0 => N347,
            Q1 => N348,
            Q2 => N349,
            Q3 => N350,
            Q4 => N351,
            \~Q0\ => N119,
            \~Q1\ => N74,
            \~Q2\ => N72,
            \~Q3\ => N71,
            \~Q4\ => N29
        );

    \CG_DMSFFPGreg5-1x_6\ : CG_DMSFFPGreg5_1x
        port map(
            CLK => N152,
            D0 => N347,
            D1 => N348,
            D2 => N349,
            D3 => N350,
            D4 => N351,
            LD => UNP,
            Q0 => N84,
            Q1 => N82,
            Q2 => N81,
            Q3 => N345,
            Q4 => N352,
            \~Q0\ => N83,
            \~Q1\ => N344,
            \~Q2\ => N80,
            \~Q3\ => N346,
            \~Q4\ => N28
        );

    \CG_DMSFFPGreg8-1x_1\ : CG_DMSFFPGreg8_1x
        port map(
            CLK => N43,
            D0 => N8,
            D1 => N7,
            D2 => N6,
            D3 => N5,
            D4 => N4,
            D5 => N3,
            D6 => N2,
            D7 => N1,
            LD => Oper,
            Q0 => N100,
            Q1 => N99,
            Q2 => N98,
            Q3 => N97,
            Q4 => N96,
            Q5 => N95,
            Q6 => N94,
            Q7 => N93,
            \~Q0\ => N27,
            \~Q1\ => N26,
            \~Q2\ => N25,
            \~Q3\ => N24,
            \~Q4\ => N23,
            \~Q5\ => N22,
            \~Q6\ => N21,
            \~Q7\ => N20
        );

    \CG_DMSFFPGreg8-1x_2\ : CG_DMSFFPGreg8_1x
        port map(
            CLK => N152,
            D0 => N92,
            D1 => N91,
            D2 => N90,
            D3 => N89,
            D4 => N88,
            D5 => N87,
            D6 => N86,
            D7 => N85,
            LD => Oper,
            Q0 => N108,
            Q1 => N107,
            Q2 => N106,
            Q3 => N105,
```

```
            Q4 => N104,
            Q5 => N103,
            Q6 => N102,
            Q7 => N101,
            \~Q0\ => N19,
            \~Q1\ => N18,
            \~Q2\ => N17,
            \~Q3\ => N16,
            \~Q4\ => N15,
            \~Q5\ => N14,
            \~Q6\ => N13,
            \~Q7\ => N12
        );

    \CG_Gain_Shifter-1x_1\ : CG_Gain_Shifter_1x
        port map(
            Gain0 => N84,
            Gain1 => N82,
            Gain2 => N81,
            Gain3 => N345,
            I0 => N100,
            I1 => N99,
            I2 => N98,
            I3 => N97,
            I4 => N96,
            I5 => N95,
            I6 => N94,
            I7 => N93,
            O10 => N65,
            O11 => N64,
            O12 => N63,
            O13 => N62,
            O14 => N61,
            O15 => N60,
            O16 => N59,
            O17 => N58,
            O5 => N70,
            O6 => N69,
            O7 => N68,
            O8 => N67,
            O9 => N66,
            \~Gain0\ => N83,
            \~Gain1\ => N344,
            \~Gain2\ => N80,
            \~Gain3\ => N346
        );

    \CG_Gain_Shifter-1x_2\ : CG_Gain_Shifter_1x
        port map(
            Gain0 => N84,
            Gain1 => N82,
            Gain2 => N81,
            Gain3 => N345,
            I0 => N108,
            I1 => N107,
            I2 => N106,
            I3 => N105,
            I4 => N104,
            I5 => N103,
            I6 => N102,
            I7 => N101,
            O10 => N52,
            O11 => N51,
            O12 => N50,
            O13 => N49,
            O14 => N48,
            O15 => N47,
            O16 => N46,
            O17 => N45,
            O5 => N57,
            O6 => N56,
            O7 => N55,
            O8 => N54,
            O9 => N53,
            \~Gain0\ => N83,
            \~Gain1\ => N344,
            \~Gain2\ => N80,
            \~Gain3\ => N346
        );


end structural;
```

B. TEST BENCH FOR THE SINGLE RANGE BIN

```
        ------------------------------------------------
--
        --
        -- Title: Test Bench for
cg_rangebinmodulator
        -- Design      : Rangebin1 with tb
        -- Author      : Hakan Bergon
        -- Company     : NPS
        --
        ------------------------------------------------
        --
        --File:
$DSN\src\TestBench\cg_rangebinmodulator_TB
.vhd
        -- Generated   : 7/11/2002, 9:07 AM
        --From:
$DSN\src\cg_rangebinmodulator.vhd
        -- By: Active-HDL Built-in Test Bench
Generator ver. 1.2s
        --
        ------------------------------------------------
        --
        -- Description : Automatically
generated Test Bench for
cg_rangebinmodulator_tb
        --
        ------------------------------------------------

        library ieee;
        use ieee.std_logic_1164.all;

                -- Add your library and
packages declaration here ...

        entity cg_rangebinmodulator_tb is
        end cg_rangebinmodulator_tb;

        architecture TB_ARCHITECTURE of
cg_rangebinmodulator_tb is
                -- Component declaration of
the tested unit
                component
cg_rangebinmodulator
        port(
                CLK : in std_logic;
                DRFM0 : in std_logic;
                DRFM1 : in std_logic;
                DRFM2 : in std_logic;
                DRFM3 : in std_logic;
                DRFM4 : in std_logic;
                Gain0 : in std_logic;
                Gain1 : in std_logic;
                Gain2 : in std_logic;
                Gain3 : in std_logic;
                I0 : in std_logic;
                I1 : in std_logic;
                I2 : in std_logic;
                I3 : in std_logic;
                I4 : in std_logic;
                I5 : in std_logic;
                I6 : in std_logic;
                I7 : in std_logic;
                I8 : in std_logic;
                I9 : in std_logic;
                I10 : in std_logic;
                I11 : in std_logic;
                I12 : in std_logic;
                I13 : in std_logic;
                I14 : in std_logic;
                I15 : in std_logic;
                Inc0 : in std_logic;
                Inc1 : in std_logic;
                Inc2 : in std_logic;
                Inc3 : in std_logic;
                Inc4 : in std_logic;
                IOV : in std_logic;
                IS0 : out std_logic;
                IS1 : out std_logic;
                IS2 : out std_logic;
                IS3 : out std_logic;
                IS4 : out std_logic;
                IS5 : out std_logic;
                IS6 : out std_logic;
                IS7 : out std_logic;
                IS8 : out std_logic;
                IS9 : out std_logic;
                IS10 : out std_logic;
                IS11 : out std_logic;
                IS12 : out std_logic;
                IS13 : out std_logic;
                IS14 : out std_logic;
                IS15 : out std_logic;
                ISOV : out std_logic;
                ODVin : in std_logic;
                ODVout : out std_logic;
                Oper : in std_logic;
                PRB : in std_logic;
                PSV : in std_logic;
                Q0 : in std_logic;
                Q1 : in std_logic;
                Q2 : in std_logic;
                Q3 : in std_logic;
                Q4 : in std_logic;
                Q5 : in std_logic;
                Q6 : in std_logic;
                Q7 : in std_logic;
                Q8 : in std_logic;
                Q9 : in std_logic;
```

```
Q10 : in std_logic;
Q11 : in std_logic;
Q12 : in std_logic;
Q13 : in std_logic;
Q14 : in std_logic;
Q15 : in std_logic;
QOV : in std_logic;
QS0 : out std_logic;
QS1 : out std_logic;
QS2 : out std_logic;
QS3 : out std_logic;
QS4 : out std_logic;
QS5 : out std_logic;
QS6 : out std_logic;
QS7 : out std_logic;
QS8 : out std_logic;
QS9 : out std_logic;
QS10 : out std_logic;
QS11 : out std_logic;
QS12 : out std_logic;
QS13 : out std_logic;
QS14 : out std_logic;
QS15 : out std_logic;
QSOV : out std_logic;
UNP : in std_logic;
URB : in std_logic;
\~I0\ : in std_logic;
\~I1\ : in std_logic;
\~I2\ : in std_logic;
\~I3\ : in std_logic;
\~I4\ : in std_logic;
\~I5\ : in std_logic;
\~I6\ : in std_logic;
\~I7\ : in std_logic;
\~I8\ : in std_logic;
\~I9\ : in std_logic;
\~I10\ : in std_logic;
\~I11\ : in std_logic;
\~I12\ : in std_logic;
\~I13\ : in std_logic;
\~I14\ : in std_logic;
\~I15\ : in std_logic;
\~IS0\ : out std_logic;
\~IS1\ : out std_logic;
\~IS2\ : out std_logic;
\~IS3\ : out std_logic;
\~IS4\ : out std_logic;
\~IS5\ : out std_logic;
\~IS6\ : out std_logic;
\~IS7\ : out std_logic;
\~IS8\ : out std_logic;
\~IS9\ : out std_logic;
\~IS10\ : out std_logic;
\~IS11\ : out std_logic;
\~IS12\ : out std_logic;
\~IS13\ : out std_logic;
\~IS14\ : out std_logic;
\~IS15\ : out std_logic;
\~Q0\ : in std_logic;
\~Q1\ : in std_logic;
\~Q2\ : in std_logic;
\~Q3\ : in std_logic;
\~Q4\ : in std_logic;
\~Q5\ : in std_logic;
\~Q6\ : in std_logic;
\~Q7\ : in std_logic;
\~Q8\ : in std_logic;
\~Q9\ : in std_logic;
\~Q10\ : in std_logic;
\~Q11\ : in std_logic;
\~Q12\ : in std_logic;
\~Q13\ : in std_logic;
\~Q14\ : in std_logic;
\~Q15\ : in std_logic;
\~QS0\ : out std_logic;
\~QS1\ : out std_logic;
\~QS2\ : out std_logic;
\~QS3\ : out std_logic;
\~QS4\ : out std_logic
\~QS5\ : out std_logic;
\~QS6\ : out std_logic;
\~QS7\ : out std_logic;
\~QS8\ : out std_logic;
\~QS9\ : out std_logic;
\~QS10\ : out std_logic;
\~QS11\ : out std_logic;
\~QS12\ : out std_logic;
\~QS13\ : out std_logic;
\~QS14\ : out std_logic;
\~QS15\ : out std_logic );
end component;

        -- Stimulus signals - signals mapped to
the input and inout ports of tested entity
signal CLK : std_logic;
signal DRFM0 : std_logic;
signal DRFM1 : std_logic;
signal DRFM2 : std_logic;
signal DRFM3 : std_logic;
signal DRFM4 : std_logic;
signal Gain0 : std_logic;
signal Gain1 : std_logic;
signal Gain2 : std_logic;
signal Gain3 : std_logic;
signal I0 : std_logic;
signal I1 : std_logic;
signal I2 : std_logic;
signal I3 : std_logic;
signal I4 : std_logic;
signal I5 : std_logic;
signal I6 : std_logic;
signal I7 : std_logic;
```

```
signal I8 : std_logic;
signal I9 : std_logic;
signal I10 : std_logic;
signal I11 : std_logic;
signal I12 : std_logic;
signal I13 : std_logic;
signal I14 : std_logic;
signal I15 : std_logic;
signal Inc0 : std_logic;
signal Inc1 : std_logic;
signal Inc2 : std_logic;
signal Inc3 : std_logic;
signal Inc4 : std_logic;
signal IOV : std_logic;
signal ODVin : std_logic;
signal Oper : std_logic;
signal PRB : std_logic;
signal PSV : std_logic;
signal Q0 : std_logic;
signal Q1 : std_logic;
signal Q2 : std_logic;
signal Q3 : std_logic;
signal Q4 : std_logic;
signal Q5 : std_logic;
signal Q6 : std_logic;
signal Q7 : std_logic;
signal Q8 : std_logic;
signal Q9 : std_logic;
signal Q10 : std_logic;
signal Q11 : std_logic;
signal Q12 : std_logic;
signal Q13 : std_logic;
signal Q14 : std_logic;
signal Q15 : std_logic;
signal QOV : std_logic;
signal UNP : std_logic;
signal URB : std_logic;
signal \~I0\ : std_logic;
signal \~I1\ : std_logic;
signal \~I2\ : std_logic;
signal \~I3\ : std_logic;
signal \~I4\ : std_logic;
signal \~I5\ : std_logic;
signal \~I6\ : std_logic;
signal \~I7\ : std_logic;
signal \~I8\ : std_logic;
signal \~I9\ : std_logic;
signal \~I10\ : std_logic;
signal \~I11\ : std_logic;
signal \~I12\ : std_logic;
signal \~I13\ : std_logic;
signal \~I14\ : std_logic;
signal \~I15\ : std_logic;
signal \~Q0\ : std_logic;
signal \~Q1\ : std_logic;
signal \~Q2\ : std_logic;
signal \~Q3\ : std_logic;
signal \~Q4\ : std_logic;
signal \~Q5\ : std_logic;
signal \~Q6\ : std_logic;
signal \~Q7\ : std_logic;
signal \~Q8\ : std_logic;
signal \~Q9\ : std_logic;
signal \~Q10\ : std_logic;
signal \~Q11\ : std_logic;
signal \~Q12\ : std_logic;
signal \~Q13\ : std_logic;
signal \~Q14\ : std_logic;
signal \~Q15\ : std_logic;
-- Observed signals - signals mapped to the output ports of tested entity
signal IS0 : std_logic;
signal IS1 : std_logic;
signal IS2 : std_logic;
signal IS3 : std_logic;
signal IS4 : std_logic;
signal IS5 : std_logic;
signal IS6 : std_logic;
signal IS7 : std_logic;
signal IS8 : std_logic;
signal IS9 : std_logic;
signal IS10 : std_logic;
signal IS11 : std_logic;
signal IS12 : std_logic;
signal IS13 : std_logic;
signal IS14 : std_logic;
signal IS15 : std_logic;
signal ISOV : std_logic;
signal ODVout : std_logic;
signal QS0 : std_logic;
signal QS1 : std_logic;
signal QS2 : std_logic;
signal QS3 : std_logic;
signal QS4 : std_logic;
signal QS5 : std_logic;
signal QS6 : std_logic;
signal QS7 : std_logic;
signal QS8 : std_logic;
signal QS9 : std_logic;
signal QS10 : std_logic;
signal QS11 : std_logic;
signal QS12 : std_logic;
signal QS13 : std_logic;
signal QS14 : std_logic;
signal QS15 : std_logic;
signal QSOV : std_logic;
signal \~IS0\ : std_logic;
signal \~IS1\ : std_logic;
signal \~IS2\ : std_logic;
signal \~IS3\ : std_logic;
signal \~IS4\ : std_logic;
signal \~IS5\ : std_logic;
```

```
        signal \~IS6\ : std_logic;
        signal \~IS7\ : std_logic;
        signal \~IS8\ : std_logic;
        signal \~IS9\ : std_logic;
        signal \~IS10\ : std_logic;
        signal \~IS11\ : std_logic;
        signal \~IS12\ : std_logic;
        signal \~IS13\ : std_logic;
        signal \~IS14\ : std_logic;
        signal \~IS15\ : std_logic;
        signal \~QS0\ : std_logic;
        signal \~QS1\ : std_logic;
        signal \~QS2\ : std_logic;
        signal \~QS3\ : std_logic;
        signal \~QS4\ : std_logic;
        signal \~QS5\ : std_logic;
        signal \~QS6\ : std_logic;
        signal \~QS7\ : std_logic;
        signal \~QS8\ : std_logic;
        signal \~QS9\ : std_logic;
        signal \~QS10\ : std_logic;
        signal \~QS11\ : std_logic;
        signal \~QS12\ : std_logic;
        signal \~QS13\ : std_logic;
        signal \~QS14\ : std_logic;
        signal \~QS15\ : std_logic;

        -- Add your code here ...

begin

- Unit Under Test port map
UUT : cg_rangebinmodulator
        port map (
                CLK => CLK,
                DRFM0 => DRFM0,
                DRFM1 => DRFM1,
                DRFM2 => DRFM2,
                DRFM3 => DRFM3,
                DRFM4 => DRFM4,
                Gain0 => Gain0,
                Gain1 => Gain1,
                Gain2 => Gain2,
                Gain3 => Gain3,
                I0 => I0,
                I1 => I1,
                I2 => I2,
                I3 => I3,
                I4 => I4,
                I5 => I5,
                I6 => I6,
                I7 => I7,
                I8 => I8,
                I9 => I9,
                I10 => I10,
                I11 => I11,
                I12 => I12,
                I13 => I13,
                I14 => I14,
                I15 => I15,
                Inc0 => Inc0,
                Inc1 => Inc1,
                Inc2 => Inc2,
                Inc3 => Inc3,
                Inc4 => Inc4,
                IOV => IOV,
                IS0 => IS0,
                IS1 => IS1,
                IS2 => IS2,
                IS3 => IS3,
                IS4 => IS4,
                IS5 => IS5,
                IS6 => IS6,
                IS7 => IS7,
                IS8 => IS8,
                IS9 => IS9,
                IS10 => IS10,
                IS11 => IS11,
                IS12 => IS12,
                IS13 => IS13,
                IS14 => IS14,
                IS15 => IS15,
                ISOV => ISOV,
                ODVin => ODVin,
                ODVout => ODVout,
                Oper => Oper,
                PRB => PRB,
                PSV => PSV,
                Q0 => Q0,
                Q1 => Q1,
                Q2 => Q2,
                Q3 => Q3,
                Q4 => Q4,
                Q5 => Q5,
                Q6 => Q6,
                Q7 => Q7,
                Q8 => Q8,
                Q9 => Q9,
                Q10 => Q10,
                Q11 => Q11,
                Q12 => Q12,
                Q13 => Q13,
                Q14 => Q14,
                Q15 => Q15,
                QOV => QOV,
                QS0 => QS0,
                QS1 => QS1,
                QS2 => QS2,
                QS3 => QS3,
                QS4 => QS4,
                QS5 => QS5,
                QS6 => QS6,
```

```
QS7 => QS7,
QS8 => QS8,
QS9 => QS9,
QS10 => QS10,
QS11 => QS11,
QS12 => QS12,
QS13 => QS13,
QS14 => QS14,
QS15 => QS15,
QSOV => QSOV,
UNP => UNP,
URB => URB,
\~I0\ => \~I0\,
\~I1\ => \~I1\,
\~I2\ => \~I2\,
\~I3\ => \~I3\,
\~I4\ => \~I4\,
\~I5\ => \~I5\,
\~I6\ => \~I6\,
\~I7\ => \~I7\,
\~I8\ => \~I8\,
\~I9\ => \~I9\,
\~I10\ => \~I10\,
\~I11\ => \~I11\,
\~I12\ => \~I12\,
\~I13\ => \~I13\,
\~I14\ => \~I14\,
\~I15\ => \~I15\,
\~IS0\ => \~IS0\,
\~IS1\ => \~IS1\,
\~IS2\ => \~IS2\,
\~IS3\ => \~IS3\,
\~IS4\ => \~IS4\,
\~IS5\ => \~IS5\,
\~IS6\ => \~IS6\,
\~IS7\ => \~IS7\,
\~IS8\ => \~IS8\,
\~IS9\ => \~IS9\,
\~IS10\ => \~IS10\,
\~IS11\ => \~IS11\,
\~IS12\ => \~IS12\,
\~IS13\ => \~IS13\,
\~IS14\ => \~IS14\,
\~IS15\ => \~IS15\,
\~Q0\ => \~Q0\,
\~Q1\ => \~Q1\,
\~Q2\ => \~Q2\,
\~Q3\ => \~Q3\,
\~Q4\ => \~Q4\,
\~Q5\ => \~Q5\,
\~Q6\ => \~Q6\,
\~Q7\ => \~Q7\,
\~Q8\ => \~Q8\,
\~Q9\ => \~Q9\,
\~Q10\ => \~Q10\,
\~Q11\ => \~Q11\,
\~Q12\ => \~Q12\,
\~Q13\ => \~Q13\,
\~Q14\ => \~Q14\,
\~Q15\ => \~Q15\,
\~QS0\ => \~QS0\,
\~QS1\ => \~QS1\,
\~QS2\ => \~QS2\,
\~QS3\ => \~QS3\,
\~QS4\ => \~QS4\,
\~QS5\ => \~QS5\,
\~QS6\ => \~QS6\,
\~QS7\ => \~QS7\,
\~QS8\ => \~QS8\,
\~QS9\ => \~QS9\,
\~QS10\ => \~QS10\,
\~QS11\ => \~QS11\,
\~QS12\ => \~QS12\,
\~QS13\ => \~QS13\,
\~QS14\ => \~QS14\,
\~QS15\ => \~QS15\
        );

        --Below VHDL code is an inserted .\compile\Waveform Editor 4.vhs
        --User can modify it ....

        STIMULUS: process
        begin  -- of stimulus process
        --wait for <time to next event>; -- <current time>
                DRFM0 <= '0';
                DRFM1 <= '0';
                DRFM2 <= '0';
                DRFM3 <= '1';
                DRFM4 <= '0';
                Inc0 <= '1';
                Inc1 <= '0';
                Inc2 <= '0';
                Inc3 <= '0';
                Inc4 <= '0';
                Gain0 <= '1';
                Gain1 <= '0';
                Gain2 <= '0';
                Gain3 <= '0';
                \~Q11\ <= '1';
                \~Q10\ <= '1';
                \~Q9\ <= '1';
                \~Q5\ <= '1';
                \~I6\ <= '1';
                \~I5\ <= '1';
                \~I4\ <= '1';
                \~I3\ <= '1';
                \~I2\ <= '1';
                QOV <= '0';
                Q15 <= '0';
                \~I15\ <= '1';
```

```
Q10 <= '0';
Q9 <= '0';
Q8 <= '0';
\~Q2\ <= '1';
I13 <= '0';
I12 <= '0';
I11 <= '0';
\~Q13\ <= '1';
IOV <= '0';
Q14 <= '0';
Q13 <= '0';
Q12 <= '0';
Q11 <= '0';
ODVin <= '0';
\~Q1\ <= '1';
\~Q0\ <= '1';
I6 <= '0';
I5 <= '0';
I4 <= '0';
I15 <= '0';
I14 <= '0';
Q2 <= '0';
CLK <= '0';
I10 <= '0';
I9 <= '0';
I8 <= '0';
I7 <= '0';
\~I9\ <= '1';
\~I8\ <= '1';
\~I7\ <= '1';
\~Q8\ <= '1';
\~Q7\ <= '1';
\~Q6\ <= '1';
\~I14\ <= '1';
\~I13\ <= '1';
\~I12\ <= '1';
\~I11\ <= '1';
\~I10\ <= '1';
\~I1\ <= '1';
\~I0\ <= '1';
URB <= '1';
UNP <= '0';
Q1 <= '0';
Q0 <= '0';
PSV <= '0';
PRB <= '1';
Oper <= '1';
Q7 <= '0';
Q6 <= '0';
Q5 <= '0';
Q4 <= '0';
Q3 <= '0';
I3 <= '0';
I2 <= '0';
I1 <= '0';
I0 <= '0';
        \~Q4\ <= '1';
        \~Q3\ <= '1';
        \~Q15\ <= '1';
        \~Q14\ <= '1';
        \~Q12\ <= '1';
wait for 1 ns; --0 fs
        CLK <= '1';
wait for 1 ns; --1 ns
        CLK <= '0';
        UNP <= '1';
        PSV <= '1';
        PRB <= '0';
wait for 1 ns; --2 ns
        CLK <= '1';
        URB <= '0';
wait for 1 ns; --3 ns
        CLK <= '0';
        UNP <= '0';
        PSV <= '0';
wait for 1 ns; --4 ns
        CLK <= '1';
wait for 1 ns; --5 ns
        CLK <= '0';
wait for 1 ns; --6 ns
        CLK <= '1';
wait for 1 ns; --7 ns
        CLK <= '0';
wait for 1 ns; --8 ns
        CLK <= '1';
wait for 1 ns; --9 ns
        CLK <= '0';
wait for 1 ns; --10 ns
        CLK <= '1';
wait for 1 ns; --11 ns
        CLK <= '0';
wait for 1 ns; --12 ns
        CLK <= '1';
wait for 1 ns; --13 ns
        CLK <= '0';
wait for 1 ns; --14 ns
        CLK <= '1';
wait for 1 ns; --15 ns
        CLK <= '0';
wait for 1 ns; --16 ns
        CLK <= '1';
wait for 1 ns; --17 ns
        CLK <= '0';
wait for 1 ns; --18 ns
        CLK <= '1';
wait for 1 ns; --19 ns
        CLK <= '0';
wait for 1 ns; --20 ns
        CLK <= '1';
wait for 1 ns; --21 ns
        CLK <= '0';
wait for 1 ns; --22 ns
```

```
        CLK <= '1';
    wait for 1 ns; --23 ns
        CLK <= '0';
    wait for 76 ns; --24 ns
--      end of stimulus events
        wait;
end process; -- end of stimulus process

        -- Add your stimulus here ...

end TB_ARCHITECTURE;
```

```
        configuration
TESTBENCH_FOR_cg_rangebinmodulator of
cg_rangebinmodulator_tb is
                for TB_ARCHITECTURE
                        for UUT :
cg_rangebinmodulator
                                use entity
work.cg_rangebinmodulator(structural);
                        end for;
                end for;
        end
TESTBENCH_FOR_cg_rangebinmodulator;
```

C. EXECUTING MACRO FOR THE ONE RANGE-BIN TEST BENCH

```
SetActiveLib
-workcomp
-include
"$DSN\src\cg_rangebinmodulator.vhd"
comp
 -include
"$DSN\src\TestBench\cg_rangebinmodulator_T
B.vhd"
asim
TESTBENCH_FOR_cg_rangebinmodulator
wave
wave -noreg CLK
wave -noreg DRFM0
wave -noreg DRFM1
wave -noreg DRFM2
wave -noreg DRFM3
wave -noreg DRFM4
wave -noreg Gain0
wave -noreg Gain1
wave -noreg Gain2
wave -noreg Gain3
wave -noreg I0
wave -noreg I1
wave -noreg I2
wave -noreg I3
wave -noreg I4
wave -noreg I5
wave -noreg I6
wave -noreg I7
wave -noreg I8
wave -noreg I9
wave -noreg I10
wave -noreg I11
wave -noreg I12
wave -noreg I13
wave -noreg I14
wave -noreg I15
wave -noreg Inc0
wave -noreg Inc1
wave -noreg Inc2
wave -noreg Inc3
wave -noreg Inc4
wave -noreg IOV
wave -noreg IS0
wave -noreg IS1
wave -noreg IS2
wave -noreg IS3
wave -noreg IS4
wave -noreg IS5
wave -noreg IS6
wave -noreg IS7
wave -noreg IS8
wave -noreg IS9
wave -noreg IS10
wave -noreg IS11
wave -noreg IS12
wave -noreg IS13
wave -noreg IS14
wave -noreg IS15
wave -noreg ISOV
wave -noreg ODVin
wave -noreg ODVout
wave -noreg Oper
wave -noreg PRB
wave -noreg PSV
wave -noreg Q0
wave -noreg Q1
wave -noreg Q2
wave -noreg Q3
wave -noreg Q4
wave -noreg Q5
wave -noreg Q6
wave -noreg Q7
wave -noreg Q8
wave -noreg Q9
wave -noreg Q10
wave -noreg Q11
wave -noreg Q12
wave -noreg Q13
wave -noreg Q14
wave -noreg Q15
wave -noreg QOV
wave -noreg QS0
wave -noreg QS1
wave -noreg QS2
wave -noreg QS3
wave -noreg QS4
wave -noreg QS5
wave -noreg QS6
wave -noreg QS7
wave -noreg QS8
wave -noreg QS9
wave -noreg QS10
wave -noreg QS11
wave -noreg QS12
wave -noreg QS13
wave -noreg QS14
wave -noreg QS15
wave -noreg QSOV
wave -noreg UNP
wave -noreg URB
wave -noreg {\~I0\}
wave -noreg {\~I1\}
wave -noreg {\~I2\}
wave -noreg {\~I3\}
wave -noreg {\~I4\}
wave -noreg {\~I5\}
wave -noreg {\~I6\}
```

```
wave -noreg {\~I7\}
wave -noreg {\~I8\}
wave -noreg {\~I9\}
wave -noreg {\~I10\}
wave -noreg {\~I11\}
wave -noreg {\~I12\}
wave -noreg {\~I13\}
wave -noreg {\~I14\}
wave -noreg {\~I15\}
wave -noreg {\~IS0\}
wave -noreg {\~IS1\}
wave -noreg {\~IS2\}
wave -noreg {\~IS3\}
wave -noreg {\~IS4\}
wave -noreg {\~IS5\}
wave -noreg {\~IS6\}
wave -noreg {\~IS7\}
wave -noreg {\~IS8\}
wave -noreg {\~IS9\}
wave -noreg {\~IS10\}
wave -noreg {\~IS11\}
wave -noreg {\~IS12\}
wave -noreg {\~IS13\}
wave -noreg {\~IS14\}
wave -noreg {\~IS15\}
wave -noreg {\~Q0\}
wave -noreg {\~Q1\}
wave -noreg {\~Q2\}
wave -noreg {\~Q3\}
wave -noreg {\~Q4\}
wave -noreg {\~Q5\}
wave -noreg {\~Q6\}
wave -noreg {\~Q7\}
wave -noreg {\~Q8\}
wave -noreg {\~Q9\}
wave -noreg {\~Q10\}
wave -noreg {\~Q11\}
wave -noreg {\~Q12\}
wave -noreg {\~Q13\}
wave -noreg {\~Q14\}
wave -noreg {\~Q15\}
wave -noreg {\~QS0\}
wave -noreg {\~QS1\}
wave -noreg {\~QS2\}
wave -noreg {\~QS3\}
wave -noreg {\~QS4\}
wave -noreg {\~QS5\}
wave -noreg {\~QS6\}
wave -noreg {\~QS7\}
wave -noreg {\~QS8\}
wave -noreg {\~QS9\}
wave -noreg {\~QS10\}
wave -noreg {\~QS11\}
wave -noreg {\~QS12\}
wave -noreg {\~QS13\}
wave -noreg {\~QS14\}
wave -noreg {\~QS15\}
run 100.00 ns
# The following lines can be used for timing simulation
#acom backannotated_vhdl_file_name>
#comp-include "$DSN\src\TestBench\cg_rangebinmodulator_TB_tim_cfg.vhd"
#asim TIMING_FOR_cg_rangebinmodulator
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. VHDL CODE FOR THE 8 RANGE-BIN MODULATOR

A. TOP LEVEL VHDL CODE

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
-- ***** DTM_8RBPs model *****
-- external ports
ENTITY DTM_8RBPs IS PORT (
        CLK : IN std_logic;
        DRFM0 : IN std_logic;
        DRFM1 : IN std_logic;
        DRFM2 : IN std_logic;
        DRFM3 : IN std_logic;
        DRFM4 : IN std_logic;
        ENABLE : IN std_logic;
        Gain0 : IN std_logic;
        Gain1 : IN std_logic;
        Gain2 : IN std_logic;
        Gain3 : IN std_logic;
        Inc0 : IN std_logic;
        Inc1 : IN std_logic;
        Inc2 : IN std_logic;
        Inc3 : IN std_logic;
        Inc4 : IN std_logic;
        InPadI0 : IN std_logic;
        InPadI1 : IN std_logic;
        InPadI2 : IN std_logic;
        InPadI3 : IN std_logic;
        InPadI4 : IN std_logic;
        InPadI5 : IN std_logic;
        InPadI6 : IN std_logic;
        InPadI7 : IN std_logic;
        InPadI8 : IN std_logic;
        InPadI9 : IN std_logic;
        InPadI10 : IN std_logic;
        InPadI11 : IN std_logic;
        InPadI12 : IN std_logic;
        InPadI13 : IN std_logic;
        InPadI14 : IN std_logic;
        InPadI15 : IN std_logic;
        InPadIOV : IN std_logic;
        InPadQ0 : IN std_logic;
        InPadQ1 : IN std_logic;
        InPadQ2 : IN std_logic;
        InPadQ3 : IN std_logic;
        InPadQ4 : IN std_logic;
        InPadQ5 : IN std_logic;
        InPadQ6 : IN std_logic;
        InPadQ7 : IN std_logic;
        InPadQ8 : IN std_logic;
        InPadQ9 : IN std_logic;
        InPadQ10 : IN std_logic;
        InPadQ11 : IN std_logic;
        InPadQ12 : IN std_logic;
        InPadQ13 : IN std_logic;
        InPadQ14 : IN std_logic;
        InPadQ15 : IN std_logic;
        InPadQOV : IN std_logic;
        \InPad~I0\ : IN std_logic;
        \InPad~I1\ : IN std_logic;
        \InPad~I2\ : IN std_logic;
        \InPad~I3\ : IN std_logic;
        \InPad~I4\ : IN std_logic;
        \InPad~I5\ : IN std_logic;
        \InPad~I6\ : IN std_logic;
        \InPad~I7\ : IN std_logic;
        \InPad~I8\ : IN std_logic;
        \InPad~I9\ : IN std_logic;
        \InPad~I10\ : IN std_logic;
        \InPad~I11\ : IN std_logic;
        \InPad~I12\ : IN std_logic;
        \InPad~I13\ : IN std_logic;
        \InPad~I14\ : IN std_logic;
        \InPad~I15\ : IN std_logic;
        \InPad~Q0\ : IN std_logic;
        \InPad~Q1\ : IN std_logic;
        \InPad~Q2\ : IN std_logic;
        \InPad~Q3\ : IN std_logic;
        \InPad~Q4\ : IN std_logic;
        \InPad~Q5\ : IN std_logic;
        \InPad~Q6\ : IN std_logic;
        \InPad~Q7\ : IN std_logic;
        \InPad~Q8\ : IN std_logic;
        \InPad~Q9\ : IN std_logic;
        \InPad~Q10\ : IN std_logic;
        \InPad~Q11\ : IN std_logic;
        \InPad~Q12\ : IN std_logic;
        \InPad~Q13\ : IN std_logic;
        \InPad~Q14\ : IN std_logic;
        \InPad~Q15\ : IN std_logic;
        ODVin : IN std_logic;
        ODVout : OUT std_logic;
        Oper : IN std_logic;
        OutPadIS0 : OUT std_logic;
        OutPadIS1 : OUT std_logic;
        OutPadIS2 : OUT std_logic;
        OutPadIS3 : OUT std_logic;
        OutPadIS4 : OUT std_logic;
        OutPadIS5 : OUT std_logic;
```

```
        OutPadIS6 : OUT std_logic;
        OutPadIS7 : OUT std_logic;
        OutPadIS8 : OUT std_logic;
        OutPadIS9 : OUT std_logic;
        OutPadIS10 : OUT std_logic;
        OutPadIS11 : OUT std_logic;
        OutPadIS12 : OUT std_logic;
        OutPadIS13 : OUT std_logic;
        OutPadIS14 : OUT std_logic;
        OutPadIS15 : OUT std_logic;
        OutPadISOV : OUT std_logic;
        OutPadQS0 : OUT std_logic;
        OutPadQS1 : OUT std_logic;
        OutPadQS2 : OUT std_logic;
        OutPadQS3 : OUT std_logic;
        OutPadQS4 : OUT std_logic;
        OutPadQS5 : OUT std_logic;
        OutPadQS6 : OUT std_logic;
        OutPadQS7 : OUT std_logic;
        OutPadQS8 : OUT std_logic;
        OutPadQS9 : OUT std_logic;
        OutPadQS10 : OUT std_logic;
        OutPadQS11 : OUT std_logic;
        OutPadQS12 : OUT std_logic;
        OutPadQS13 : OUT std_logic;
        OutPadQS14 : OUT std_logic;
        OutPadQS15 : OUT std_logic;
        OutPadQSOV: OUT std_logic;
        \OutPad~IS0\ : OUT std_logic;
        \OutPad~IS1\ : OUT std_logic;
        \OutPad~IS2\ : OUT std_logic;
        \OutPad~IS3\ : OUT std_logic;
        \OutPad~IS4\ : OUT std_logic;
        \OutPad~IS5\ : OUT std_logic;
        \OutPad~IS6\ : OUT std_logic;
        \OutPad~IS7\ : OUT std_logic;
        \OutPad~IS8\ : OUT std_logic;
        \OutPad~IS9\ : OUT std_logic;
        \OutPad~IS10\:OUT std_logic;
        \OutPad~IS11\:OUT std_logic;
        \OutPad~IS12\:OUT std_logic;
        \OutPad~IS13\:OUT std_logic;
        \OutPad~IS14\:OUT std_logic;
        \OutPad~IS15\:OUT std_logic;
        \OutPad~QS0\:OUT std_logic;
        \OutPad~QS1\:OUT std_logic;
        \OutPad~QS2\:OUT std_logic;
        \OutPad~QS3\:OUT std_logic;
        \OutPad~QS4\:OUT std_logic;
        \OutPad~QS5\:OUT std_logic;
        \OutPad~QS6\:OUT std_logic;
        \OutPad~QS7\:OUT std_logic;
        \OutPad~QS8\:OUT std_logic;
        \OutPad~QS9\:OUT std_logic;
\OutPad~QS10\:      OUT      std_logic;
\OutPad~QS11\ : OUT std_logic;
        \OutPad~QS12\ : OUT std_logic;
        \OutPad~QS13\ : OUT std_logic;
        \OutPad~QS14\ : OUT std_logic;
        \OutPad~QS15\ : OUT std_logic;
        PSV : IN std_logic;
        RBinSelect0 : IN std_logic;
        RBinSelect1 : IN std_logic;
        RBinSelect2 : IN std_logic;
        UNP : IN std_logic;
        URB : IN std_logic
        );
        END DTM_8RBPs;

        -- internal structure
        ARCHITECTURE      structural      OF
DTM_8RBPs IS

        -- COMPONENTS

        COMPONENT DTM_SigFanout
        PORT (
                SigIn : IN std_logic;
                SigOut1 : OUT std_logic;
                SigOut2 : OUT std_logic
        );
        END COMPONENT;

        COMPONENT
CG_RangeBinModulator
        PORT (
                CLK : IN std_logic;
                DRFM0 : IN std_logic;
                DRFM1 : IN std_logic;
                DRFM2 : IN std_logic;
                DRFM3 : IN std_logic;
                DRFM4 : IN std_logic;
                Gain0 : IN std_logic;
                Gain1 : IN std_logic;
                Gain2 : IN std_logic;
                Gain3 : IN std_logic;
                I0 : IN std_logic;
                I1 : IN std_logic;
                I2 : IN std_logic;
                I3 : IN std_logic;
                I4 : IN std_logic;
                I5 : IN std_logic;
                I6 : IN std_logic;
                I7 : IN std_logic;
                I8 : IN std_logic;
                I9 : IN std_logic;
                I10 : IN std_logic;
                I11 : IN std_logic;
                I12 : IN std_logic;
                I13 : IN std_logic;
                I14 : IN std_logic;
                I15 : IN std_logic;
```

```
Inc0 : IN std_logic;
Inc1 : IN std_logic;
Inc2 : IN std_logic;
Inc3 : IN std_logic;
Inc4 : IN std_logic;
IOV : IN std_logic;
IS0 : OUT std_logic;
IS1 : OUT std_logic;
IS2 : OUT std_logic;
IS3 : OUT std_logic;
IS4 : OUT std_logic;
IS5 : OUT std_logic;
IS6 : OUT std_logic;
IS7 : OUT std_logic;
IS8 : OUT std_logic;
IS9 : OUT std_logic;
IS10 : OUT std_logic;
IS11 : OUT std_logic;
IS12 : OUT std_logic;
IS13 : OUT std_logic;
IS14 : OUT std_logic;
IS15 : OUT std_logic;
ISOV : OUT std_logic;
ODVin : IN std_logic;
ODVout : OUT std_logic;
Oper : IN std_logic;
PRB : IN std_logic;
PSV : IN std_logic;
Q0 : IN std_logic;
Q1 : IN std_logic;
Q2 : IN std_logic;
Q3 : IN std_logic;
Q4 : IN std_logic;
Q5 : IN std_logic;
Q6 : IN std_logic;
Q7 : IN std_logic;
Q8 : IN std_logic;
Q9 : IN std_logic;
Q10 : IN std_logic;
Q11 : IN std_logic;
Q12 : IN std_logic;
Q13 : IN std_logic;
Q14 : IN std_logic;
Q15 : IN std_logic;
QOV : IN std_logic;
QS0 : OUT std_logic;
QS1 : OUT std_logic;
QS2 : OUT std_logic;
QS3 : OUT std_logic;
QS4 : OUT std_logic;
QS5 : OUT std_logic;
QS6 : OUT std_logic;
QS7 : OUT std_logic;
QS8 : OUT std_logic;
QS9 : OUT std_logic;
QS10 : OUT std_logic;
QS11 : OUT std_logic;
QS12 : OUT std_logic;
QS13 : OUT std_logic;
QS14 : OUT std_logic;
QS15 : OUT std_logic;
QSOV : OUT std_logic;
UNP : IN std_logic;
URB : IN std_logic;
\~I0\ : IN std_logic;
\~I1\ : IN std_logic;
\~I2\ : IN std_logic;
\~I3\ : IN std_logic;
\~I4\ : IN std_logic;
\~I5\ : IN std_logic;
\~I6\ : IN std_logic;
\~I7\ : IN std_logic;
\~I8\ : IN std_logic;
\~I9\ : IN std_logic;
\~I10\ : IN std_logic;
\~I11\ : IN std_logic;
\~I12\ : IN std_logic;
\~I13\ : IN std_logic;
\~I14\ : IN std_logic;
\~I15\ : IN std_logic;
\~IS0\ : OUT std_logic;
\~IS1\ : OUT std_logic;
\~IS2\ : OUT std_logic;
\~IS3\ : OUT std_logic;
\~IS4\ : OUT std_logic;
\~IS5\ : OUT std_logic;
\~IS6\ : OUT std_logic;
\~IS7\ : OUT std_logic;
\~IS8\ : OUT std_logic;
\~IS9\ : OUT std_logic;
\~IS10\ : OUT std_logic;
\~IS11\ : OUT std_logic;
\~IS12\ : OUT std_logic;
\~IS13\ : OUT std_logic;
\~IS14\ : OUT std_logic;
\~IS15\ : OUT std_logic;
\~Q0\ : IN std_logic;
\~Q1\ : IN std_logic;
\~Q2\ : IN std_logic;
\~Q3\ : IN std_logic;
\~Q4\ : IN std_logic;
\~Q5\ : IN std_logic;
\~Q6\ : IN std_logic;
\~Q7\ : IN std_logic;
\~Q8\ : IN std_logic;
\~Q9\ : IN std_logic;
\~Q10\ : IN std_logic;
\~Q11\ : IN std_logic;
\~Q12\ : IN std_logic;
\~Q13\ : IN std_logic;
\~Q14\ : IN std_logic;
\~Q15\ : IN std_logic;
```

```
        \~QS0\ : OUT std_logic;
        \~QS1\ : OUT std_logic;
        \~QS2\ : OUT std_logic;
        \~QS3\ : OUT std_logic;
        \~QS4\ : OUT std_logic;
        \~QS5\ : OUT std_logic;
        \~QS6\ : OUT std_logic;
        \~QS7\ : OUT std_logic;
        \~QS8\ : OUT std_logic;
        \~QS9\ : OUT std_logic;
        \~QS10\ : OUT std_logic;
        \~QS11\ : OUT std_logic;
        \~QS12\ : OUT std_logic;
        \~QS13\ : OUT std_logic;
        \~QS14\ : OUT std_logic;
        \~QS15\ : OUT std_logic
);
END COMPONENT;

COMPONENT BO_3to8DECODER
PORT (
        D0 : OUT std_logic;
        D1 : OUT std_logic;
        D2 : OUT std_logic;
        D3 : OUT std_logic;
        D4 : OUT std_logic;
        D5 : OUT std_logic;
        D6 : OUT std_logic;
        D7 : OUT std_logic;
        Enable : IN std_logic;
        Select0 : IN std_logic;
        Select1 : IN std_logic;
        Select2 : IN std_logic
);
END COMPONENT;

-- SIGNALS

SIGNAL N1271 : std_logic;
SIGNAL N1270 : std_logic;
SIGNAL N1269 : std_logic;
SIGNAL N1268 : std_logic;
SIGNAL N1267 : std_logic;
SIGNAL N1266 : std_logic;
SIGNAL N1265 : std_logic;
SIGNAL N1264 : std_logic;
SIGNAL N1259 : std_logic;
SIGNAL N1258 : std_logic;
SIGNAL N1257 : std_logic;
SIGNAL N1256 : std_logic;
SIGNAL N1254 : std_logic;
SIGNAL N1209 : std_logic;
SIGNAL N1208 : std_logic;
SIGNAL N1207 : std_logic;
SIGNAL N1206 : std_logic;
SIGNAL N1205 : std_logic;
SIGNAL N1189 : std_logic;
SIGNAL N1188 : std_logic;
SIGNAL N1187 : std_logic;
SIGNAL N1186 : std_logic;
SIGNAL N1185 : std_logic;
SIGNAL N1184 : std_logic;
SIGNAL N1183 : std_logic;
SIGNAL N1182 : std_logic;
SIGNAL N1181 : std_logic;
SIGNAL N1175 : std_logic;
SIGNAL N1139 : std_logic;
SIGNAL N1138 : std_logic;
SIGNAL N1137 : std_logic;
SIGNAL N1136 : std_logic;
SIGNAL N1135 : std_logic;
SIGNAL N1134 : std_logic;
SIGNAL N1133 : std_logic;
SIGNAL N1132 : std_logic;
SIGNAL N1124 : std_logic;
SIGNAL N1113 : std_logic;
SIGNAL N1112 : std_logic;
SIGNAL N1111 : std_logic;
SIGNAL N1110 : std_logic;
SIGNAL N1101 : std_logic;
SIGNAL N1088 : std_logic;
SIGNAL N1087 : std_logic;
SIGNAL N1086 : std_logic;
SIGNAL N1085 : std_logic;
SIGNAL N1084 : std_logic;
SIGNAL N1083 : std_logic;
SIGNAL N1082 : std_logic;
SIGNAL N1081 : std_logic;
SIGNAL N1080 : std_logic;
SIGNAL N1079 : std_logic;
SIGNAL N1073 : std_logic;
SIGNAL N1072 : std_logic;
SIGNAL N1071 : std_logic;
SIGNAL N1061 : std_logic;
SIGNAL N1060 : std_logic;
SIGNAL N1059 : std_logic;
SIGNAL N1058 : std_logic;
SIGNAL N1057 : std_logic;
SIGNAL N1055 : std_logic;
SIGNAL N1054 : std_logic;
SIGNAL N1053 : std_logic;
SIGNAL N1052 : std_logic;
SIGNAL N1051 : std_logic;
SIGNAL N1035 : std_logic;
SIGNAL N1034 : std_logic;
SIGNAL N1033 : std_logic;
SIGNAL N1032 : std_logic;
SIGNAL N1031 : std_logic;
SIGNAL N1030 : std_logic;
SIGNAL N1029 : std_logic;
SIGNAL N1028 : std_logic;
SIGNAL N1027 : std_logic;
```

```
SIGNAL N1024 : std_logic;
SIGNAL N1021 : std_logic;
SIGNAL N1020 : std_logic;
SIGNAL N1019 : std_logic;
SIGNAL N1010 : std_logic;
SIGNAL N1009 : std_logic;
SIGNAL N1008 : std_logic;
SIGNAL N1007 : std_logic;
SIGNAL N1006 : std_logic;
SIGNAL N1005 : std_logic;
SIGNAL N1004 : std_logic;
SIGNAL N1002 : std_logic;
SIGNAL N1001 : std_logic;
SIGNAL N1000 : std_logic;
SIGNAL N999 : std_logic;
SIGNAL N998 : std_logic;
SIGNAL N997 : std_logic;
SIGNAL N996 : std_logic;
SIGNAL N995 : std_logic;
SIGNAL N994 : std_logic;
SIGNAL N993 : std_logic;
SIGNAL N988 : std_logic;
SIGNAL N985 : std_logic;
SIGNAL N984 : std_logic;
SIGNAL N983 : std_logic;
SIGNAL N982 : std_logic;
SIGNAL N981 : std_logic;
SIGNAL N980 : std_logic;
SIGNAL N979 : std_logic;
SIGNAL N978 : std_logic;
SIGNAL N970 : std_logic;
SIGNAL N959 : std_logic;
SIGNAL N958 : std_logic;
SIGNAL N957 : std_logic;
SIGNAL N956 : std_logic;
SIGNAL N955 : std_logic;
SIGNAL N954 : std_logic;
SIGNAL N952 : std_logic;
SIGNAL N934 : std_logic;
SIGNAL N933 : std_logic;
SIGNAL N932 : std_logic;
SIGNAL N931 : std_logic;
SIGNAL N930 : std_logic;
SIGNAL N929 : std_logic;
SIGNAL N928 : std_logic;
SIGNAL N927 : std_logic;
SIGNAL N926 : std_logic;
SIGNAL N925 : std_logic;
SIGNAL N919 : std_logic;
SIGNAL N918 : std_logic;
SIGNAL N917 : std_logic;
SIGNAL N907 : std_logic;
SIGNAL N906 : std_logic;
SIGNAL N905 : std_logic;
SIGNAL N904 : std_logic;
SIGNAL N903 : std_logic;
SIGNAL N902 : std_logic;
SIGNAL N896 : std_logic;
SIGNAL N895 : std_logic;
SIGNAL N894 : std_logic;
SIGNAL N893 : std_logic;
SIGNAL N892 : std_logic;
SIGNAL N891 : std_logic;
SIGNAL N890 : std_logic;
SIGNAL N889 : std_logic;
SIGNAL N888 : std_logic;
SIGNAL N886 : std_logic;
SIGNAL N885 : std_logic;
SIGNAL N884 : std_logic;
SIGNAL N883 : std_logic;
SIGNAL N882 : std_logic;
SIGNAL N872 : std_logic;
SIGNAL N870 : std_logic;
SIGNAL N867 : std_logic;
SIGNAL N865 : std_logic;
SIGNAL N856 : std_logic;
SIGNAL N855 : std_logic;
SIGNAL N854 : std_logic;
SIGNAL N853 : std_logic;
SIGNAL N852 : std_logic;
SIGNAL N851 : std_logic;
SIGNAL N850 : std_logic;
SIGNAL N848 : std_logic;
SIGNAL N847 : std_logic;
SIGNAL N846 : std_logic;
SIGNAL N845 : std_logic;
SIGNAL N844 : std_logic;
SIGNAL N843 : std_logic;
SIGNAL N842 : std_logic;
SIGNAL N841 : std_logic;
SIGNAL N840 : std_logic;
SIGNAL N839 : std_logic;
SIGNAL N834 : std_logic;
SIGNAL N833 : std_logic;
SIGNAL N832 : std_logic;
SIGNAL N823 : std_logic;
SIGNAL N822 : std_logic;
SIGNAL N821 : std_logic;
SIGNAL N820 : std_logic;
SIGNAL N819 : std_logic;
SIGNAL N818 : std_logic;
SIGNAL N817 : std_logic;
SIGNAL N815 : std_logic;
SIGNAL N814 : std_logic;
SIGNAL N813 : std_logic;
SIGNAL N812 : std_logic;
SIGNAL N811 : std_logic;
SIGNAL N810 : std_logic;
SIGNAL N809 : std_logic;
SIGNAL N808 : std_logic;
SIGNAL N807 : std_logic;
SIGNAL N806 : std_logic;
```

```
SIGNAL N798 : std_logic;
SIGNAL N790 : std_logic;
SIGNAL N789 : std_logic;
SIGNAL N788 : std_logic;
SIGNAL N787 : std_logic;
SIGNAL N786 : std_logic;
SIGNAL N785 : std_logic;
SIGNAL N784 : std_logic;
SIGNAL N783 : std_logic;
SIGNAL N782 : std_logic;
SIGNAL N781 : std_logic;
SIGNAL N780 : std_logic;
SIGNAL N770 : std_logic;
SIGNAL N769 : std_logic;
SIGNAL N768 : std_logic;
SIGNAL N767 : std_logic;
SIGNAL N766 : std_logic;
SIGNAL N764 : std_logic;
SIGNAL N762 : std_logic;
SIGNAL N761 : std_logic;
SIGNAL N760 : std_logic;
SIGNAL N759 : std_logic;
SIGNAL N758 : std_logic;
SIGNAL N757 : std_logic;
SIGNAL N756 : std_logic;
SIGNAL N755 : std_logic;
SIGNAL N754 : std_logic;
SIGNAL N742 : std_logic;
SIGNAL N741 : std_logic;
SIGNAL N740 : std_logic;
SIGNAL N739 : std_logic;
SIGNAL N738 : std_logic;
SIGNAL N737 : std_logic;
SIGNAL N736 : std_logic;
SIGNAL N735 : std_logic;
SIGNAL N734 : std_logic;
SIGNAL N733 : std_logic;
SIGNAL N731 : std_logic;
SIGNAL N730 : std_logic;
SIGNAL N729 : std_logic;
SIGNAL N728 : std_logic;
SIGNAL N718 : std_logic;
SIGNAL N717 : std_logic;
SIGNAL N716 : std_logic;
SIGNAL N714 : std_logic;
SIGNAL N710 : std_logic;
SIGNAL N709 : std_logic;
SIGNAL N708 : std_logic;
SIGNAL N707 : std_logic;
SIGNAL N706 : std_logic;
SIGNAL N705 : std_logic;
SIGNAL N704 : std_logic;
SIGNAL N703 : std_logic;
SIGNAL N695 : std_logic;
SIGNAL N684 : std_logic;
SIGNAL N683 : std_logic;
SIGNAL N682 : std_logic;
SIGNAL N681 : std_logic;
SIGNAL N678 : std_logic;
SIGNAL N669 : std_logic;
SIGNAL N668 : std_logic;
SIGNAL N667 : std_logic;
SIGNAL N666 : std_logic;
SIGNAL N665 : std_logic;
SIGNAL N661 : std_logic;
SIGNAL N660 : std_logic;
SIGNAL N659 : std_logic;
SIGNAL N658 : std_logic;
SIGNAL N657 : std_logic;
SIGNAL N656 : std_logic;
SIGNAL N655 : std_logic;
SIGNAL N654 : std_logic;
SIGNAL N653 : std_logic;
SIGNAL N652 : std_logic;
SIGNAL N648 : std_logic;
SIGNAL N644 : std_logic;
SIGNAL N616 : std_logic;
SIGNAL N615 : std_logic;
SIGNAL N614 : std_logic;
SIGNAL N613 : std_logic;
SIGNAL N612 : std_logic;
SIGNAL N611 : std_logic;
SIGNAL N608 : std_logic;
SIGNAL N607 : std_logic;
SIGNAL N606 : std_logic;
SIGNAL N605 : std_logic;
SIGNAL N604 : std_logic;
SIGNAL N603 : std_logic;
SIGNAL N602 : std_logic;
SIGNAL N601 : std_logic;
SIGNAL N600 : std_logic;
SIGNAL N595 : std_logic;
SIGNAL N593 : std_logic;
SIGNAL N592 : std_logic;
SIGNAL N591 : std_logic;
SIGNAL N590 : std_logic;
SIGNAL N589 : std_logic;
SIGNAL N573 : std_logic;
SIGNAL N572 : std_logic;
SIGNAL N571 : std_logic;
SIGNAL N570 : std_logic;
SIGNAL N569 : std_logic;
SIGNAL N568 : std_logic;
SIGNAL N567 : std_logic;
SIGNAL N566 : std_logic;
SIGNAL N565 : std_logic;
SIGNAL N561 : std_logic;
SIGNAL N559 : std_logic;
SIGNAL N556 : std_logic;
SIGNAL N555 : std_logic;
SIGNAL N554 : std_logic;
SIGNAL N553 : std_logic;
```

```
SIGNAL N552 : std_logic;
SIGNAL N551 : std_logic;
SIGNAL N550 : std_logic;
SIGNAL N549 : std_logic;
SIGNAL N543 : std_logic;
SIGNAL N542 : std_logic;
SIGNAL N541 : std_logic;
SIGNAL N530 : std_logic;
SIGNAL N529 : std_logic;
SIGNAL N528 : std_logic;
SIGNAL N526 : std_logic;
SIGNAL N523 : std_logic;
SIGNAL N522 : std_logic;
SIGNAL N521 : std_logic;
SIGNAL N520 : std_logic;
SIGNAL N519 : std_logic;
SIGNAL N518 : std_logic;
SIGNAL N517 : std_logic;
SIGNAL N516 : std_logic;
SIGNAL N508 : std_logic;
SIGNAL N497 : std_logic;
SIGNAL N496 : std_logic;
SIGNAL N495 : std_logic;
SIGNAL N494 : std_logic;
SIGNAL N490 : std_logic;
SIGNAL N472 : std_logic;
SIGNAL N471 : std_logic;
SIGNAL N470 : std_logic;
SIGNAL N469 : std_logic;
SIGNAL N468 : std_logic;
SIGNAL N467 : std_logic;
SIGNAL N466 : std_logic;
SIGNAL N465 : std_logic;
SIGNAL N464 : std_logic;
SIGNAL N463 : std_logic;
SIGNAL N457 : std_logic;
SIGNAL N456 : std_logic;
SIGNAL N455 : std_logic;
SIGNAL N445 : std_logic;
SIGNAL N444 : std_logic;
SIGNAL N443 : std_logic;
SIGNAL N442 : std_logic;
SIGNAL N441 : std_logic;
SIGNAL N439 : std_logic;
SIGNAL N438 : std_logic;
SIGNAL N437 : std_logic;
SIGNAL N436 : std_logic;
SIGNAL N435 : std_logic;
SIGNAL N419 : std_logic;
SIGNAL N418 : std_logic;
SIGNAL N417 : std_logic;
SIGNAL N416 : std_logic;
SIGNAL N415 : std_logic;
SIGNAL N414 : std_logic;
SIGNAL N413 : std_logic;
SIGNAL N412 : std_logic;
SIGNAL N411 : std_logic;
SIGNAL N408 : std_logic;
SIGNAL N405 : std_logic;
SIGNAL N403 : std_logic;
SIGNAL N394 : std_logic;
SIGNAL N393 : std_logic;
SIGNAL N392 : std_logic;
SIGNAL N391 : std_logic;
SIGNAL N390 : std_logic;
SIGNAL N389 : std_logic;
SIGNAL N388 : std_logic;
SIGNAL N386 : std_logic;
SIGNAL N385 : std_logic;
SIGNAL N384 : std_logic;
SIGNAL N383 : std_logic;
SIGNAL N382 : std_logic;
SIGNAL N381 : std_logic;
SIGNAL N380 : std_logic;
SIGNAL N379 : std_logic;
SIGNAL N378 : std_logic;
SIGNAL N377 : std_logic;
SIGNAL N372 : std_logic;
SIGNAL N369 : std_logic;
SIGNAL N368 : std_logic;
SIGNAL N367 : std_logic;
SIGNAL N366 : std_logic;
SIGNAL N365 : std_logic;
SIGNAL N364 : std_logic;
SIGNAL N363 : std_logic;
SIGNAL N362 : std_logic;
SIGNAL N354 : std_logic;
SIGNAL N343 : std_logic;
SIGNAL N342 : std_logic;
SIGNAL N341 : std_logic;
SIGNAL N340 : std_logic;
SIGNAL N336 : std_logic;
SIGNAL N318 : std_logic;
SIGNAL N317 : std_logic;
SIGNAL N316 : std_logic;
SIGNAL N315 : std_logic;
SIGNAL N314 : std_logic;
SIGNAL N313 : std_logic;
SIGNAL N312 : std_logic;
SIGNAL N311 : std_logic;
SIGNAL N310 : std_logic;
SIGNAL N309 : std_logic;
SIGNAL N303 : std_logic;
SIGNAL N302 : std_logic;
SIGNAL N301 : std_logic;
SIGNAL N291 : std_logic;
SIGNAL N290 : std_logic;
SIGNAL N289 : std_logic;
SIGNAL N288 : std_logic;
SIGNAL N287 : std_logic;
SIGNAL N285 : std_logic;
SIGNAL N284 : std_logic;
```

```
SIGNAL N283 : std_logic;
SIGNAL N282 : std_logic;
SIGNAL N281 : std_logic;
SIGNAL N265 : std_logic;
SIGNAL N264 : std_logic;
SIGNAL N263 : std_logic;
SIGNAL N262 : std_logic;
SIGNAL N261 : std_logic;
SIGNAL N260 : std_logic;
SIGNAL N259 : std_logic;
SIGNAL N258 : std_logic;
SIGNAL N257 : std_logic;
SIGNAL N254 : std_logic;
SIGNAL N251 : std_logic;
SIGNAL N249 : std_logic;
SIGNAL N240 : std_logic;
SIGNAL N239 : std_logic;
SIGNAL N238 : std_logic;
SIGNAL N237 : std_logic;
SIGNAL N236 : std_logic;
SIGNAL N235 : std_logic;
SIGNAL N234 : std_logic;
SIGNAL N232 : std_logic;
SIGNAL N231 : std_logic;
SIGNAL N230 : std_logic;
SIGNAL N229 : std_logic;
SIGNAL N228 : std_logic;
SIGNAL N227 : std_logic;
SIGNAL N226 : std_logic;
SIGNAL N225 : std_logic;
SIGNAL N224 : std_logic;
SIGNAL N223 : std_logic;
SIGNAL N218 : std_logic;
SIGNAL N215 : std_logic;
SIGNAL N214 : std_logic;
SIGNAL N213 : std_logic;
SIGNAL N212 : std_logic;
SIGNAL N211 : std_logic;
SIGNAL N210 : std_logic;
SIGNAL N209 : std_logic;
SIGNAL N208 : std_logic;
SIGNAL N200 : std_logic;
SIGNAL N189 : std_logic;
SIGNAL N188 : std_logic;
SIGNAL N187 : std_logic;
SIGNAL N186 : std_logic;
SIGNAL N182 : std_logic;
SIGNAL N174 : std_logic;
SIGNAL N173 : std_logic;
SIGNAL N172 : std_logic;
SIGNAL N171 : std_logic;
SIGNAL N170 : std_logic;
SIGNAL N169 : std_logic;
SIGNAL N168 : std_logic;
SIGNAL N167 : std_logic;
SIGNAL N166 : std_logic;
SIGNAL N165 : std_logic;
SIGNAL N164 : std_logic;
SIGNAL N163 : std_logic;
SIGNAL N162 : std_logic;
SIGNAL N161 : std_logic;
SIGNAL N160 : std_logic;
SIGNAL N159 : std_logic;
SIGNAL N158 : std_logic;
SIGNAL N157 : std_logic;
SIGNAL N156 : std_logic;
SIGNAL N155 : std_logic;
SIGNAL N149 : std_logic;
SIGNAL N148 : std_logic;
SIGNAL N147 : std_logic;
SIGNAL N137 : std_logic;
SIGNAL N136 : std_logic;
SIGNAL N135 : std_logic;
SIGNAL N134 : std_logic;
SIGNAL N133 : std_logic;
SIGNAL N100 : std_logic;
SIGNAL N95 : std_logic;
SIGNAL N86 : std_logic;
SIGNAL N85 : std_logic;
SIGNAL N84 : std_logic;
SIGNAL N83 : std_logic;
SIGNAL N82 : std_logic;
SIGNAL N81 : std_logic;
SIGNAL N80 : std_logic;
SIGNAL N78 : std_logic;
SIGNAL N77 : std_logic;
SIGNAL N76 : std_logic;
SIGNAL N75 : std_logic;
SIGNAL N74 : std_logic;
SIGNAL N73 : std_logic;
SIGNAL N72 : std_logic;
SIGNAL N71 : std_logic;
SIGNAL N70 : std_logic;
SIGNAL N69 : std_logic;
SIGNAL N64 : std_logic;
SIGNAL N28 : std_logic;
SIGNAL N25 : std_logic;
SIGNAL N22 : std_logic;
SIGNAL N19 : std_logic;
SIGNAL N16 : std_logic;
SIGNAL N13 : std_logic;
SIGNAL N10 : std_logic;
SIGNAL N7 : std_logic;
SIGNAL N4 : std_logic;
SIGNAL N1 : std_logic;

-- INSTANCES
BEGIN
DTM_SigFanout_21 : DTM_SigFanout
PORT MAP(
        SigIn => Gain3,
        SigOut1 => N1,
```

```
                SigOut2 => N1257
        );
        DTM_SigFanout_28 : DTM_SigFanout
PORT MAP(
                SigIn => Gain2,
                SigOut1 => N4,
                SigOut2 => N1258
        );
        DTM_SigFanout_29 : DTM_SigFanout
PORT MAP(
                SigIn => Gain1,
                SigOut1 => N7,
                SigOut2 => N1259
        );
        DTM_SigFanout_30 : DTM_SigFanout
PORT MAP(
                SigIn => Gain0,
                SigOut1 => N10,
                SigOut2 => N1256
        );
        DTM_SigFanout_22 : DTM_SigFanout
PORT MAP(
                SigIn => CLK,
                SigOut1 => N13,
                SigOut2 => N1254
        );
        DTM_SigFanout_23 : DTM_SigFanout
PORT MAP(
                SigIn => URB,
                SigOut1 => N16,
                SigOut2 => N1020
        );
        DTM_SigFanout_24 : DTM_SigFanout
PORT MAP(
                SigIn => PSV,
                SigOut1 => N19,
                SigOut2 => N954
        );
        DTM_SigFanout_25 : DTM_SigFanout
PORT MAP(
                SigIn => UNP,
                SigOut1 => N22,
                SigOut2 => N1101
        );
        DTM_SigFanout_26 : DTM_SigFanout
PORT MAP(
                SigIn => Oper,
                SigOut1 => N25,
                SigOut2 => N955
        );
        CG_RangeBinModulator_8          :
CG_RangeBinModulator   PORT MAP(
                CLK => N13,
                DRFM0 => N169,
                DRFM1 => N168,
                DRFM2 => N167,
                DRFM3 => N166,
                DRFM4 => N165,
                Gain0 => N10,
                Gain1 => N7,
                Gain2 => N4,
                Gain3 => N1,
                I0 => N148,
                I1 => N147,
                I2 => N257,
                I3 => N258,
                I4 => N259,
                I5 => N260,
                I6 => N261,
                I7 => N262,
                I8 => N263,
                I9 => N264,
                I10 => N265,
                I11 => N137,
                I12 => N136,
                I13 => N135,
                I14 => N134,
                I15 => N133,
                Inc0 => N173,
                Inc1 => N172,
                Inc2 => N171,
                Inc3 => N170,
                Inc4 => N174,
                IOV => N251,
                IS0 => OutPadIS0,
                IS1 => OutPadIS1,
                IS2 => OutPadIS2,
                IS3 => OutPadIS3,
                IS4 => OutPadIS4,
                IS5 => OutPadIS5,
                IS6 => OutPadIS6,
                IS7 => OutPadIS7,
                IS8 => OutPadIS8,
                IS9 => OutPadIS9,
                IS10 => OutPadIS10,
                IS11 => OutPadIS11,
                IS12 => OutPadIS12,
                IS13 => OutPadIS13,
                IS14 => OutPadIS14,
                IS15 => OutPadIS15,
                ISOV => OutPadISOV,
                ODVin => N28,
                ODVout => ODVout,
                Oper => N25,
                PRB => N1271,
                PSV => N19,
                Q0 => N164,
                Q1 => N163,
                Q2 => N162,
                Q3 => N161,
                Q4 => N160,
                Q5 => N159,
```

```
Q6 => N158,
Q7 => N157,
Q8 => N156,
Q9 => N155,
Q10 => N281,
Q11 => N282,
Q12 => N283,
Q13 => N284,
Q14 => N285,
Q15 => N149,
QOV => N100,
QS0 => OutPadQS0,
QS1 => OutPadQS1,
QS2 => OutPadQS2,
QS3 => OutPadQS3,
QS4 => OutPadQS4,
QS5 => OutPadQS5,
QS6 => OutPadQS6,
QS7 => OutPadQS7,
QS8 => OutPadQS8,
QS9 => OutPadQS9,
QS10 => OutPadQS10,
QS11 => OutPadQS11,
QS12 => OutPadQS12,
QS13 => OutPadQS13,
QS14 => OutPadQS14,
QS15 => OutPadQS15,
QSOV => OutPadQSOV,
UNP => N22,
URB => N16,
\~I0\ => N200,
\~I1\ => N78,
\~I2\ => N77,
\~I3\ => N76,
\~I4\ => N75,
\~I5\ => N74,
\~I6\ => N73,
\~I7\ => N72,
\~I8\ => N71,
\~I9\ => N70,
\~I10\ => N69,
\~I11\ => N189,
\~I12\ => N188,
\~I13\ => N187,
\~I14\ => N186,
\~I15\ => N64,
\~IS0\ => \OutPad~IS0\,
\~IS1\ => \OutPad~IS1\,
\~IS2\ => \OutPad~IS2\,
\~IS3\ => \OutPad~IS3\,
\~IS4\ => \OutPad~IS4\,
\~IS5\ => \OutPad~IS5\,
\~IS6\ => \OutPad~IS6\,
\~IS7\ => \OutPad~IS7\,
\~IS8\ => \OutPad~IS8\,
\~IS9\ => \OutPad~IS9\,
\~IS10\ => \OutPad~IS10\,
\~IS11\ => \OutPad~IS11\,
\~IS12\ => \OutPad~IS12\,
\~IS13\ => \OutPad~IS13\,
\~IS14\ => \OutPad~IS14\,
\~IS15\ => \OutPad~IS15\,
\~Q0\ => N95,
\~Q1\ => N215,
\~Q2\ => N214,
\~Q3\ => N213,
\~Q4\ => N212,
\~Q5\ => N211,
\~Q6\ => N210,
\~Q7\ => N209,
\~Q8\ => N208,
\~Q9\ => N86,
\~Q10\ => N85,
\~Q11\ => N84,
\~Q12\ => N83,
\~Q13\ => N82,
\~Q14\ => N81,
\~Q15\ => N80,
\~QS0\ => \OutPad~QS0\,
\~QS1\ => \OutPad~QS1\,
\~QS2\ => \OutPad~QS2\,
\~QS3\ => \OutPad~QS3\,
\~QS4\ => \OutPad~QS4\,
\~QS5\ => \OutPad~QS5\,
\~QS6\ => \OutPad~QS6\,
\~QS7\ => \OutPad~QS7\,
\~QS8\ => \OutPad~QS8\,
\~QS9\ => \OutPad~QS9\,
\~QS10\ => \OutPad~QS10\,
\~QS11\ => \OutPad~QS11\,
\~QS12\ => \OutPad~QS12\,
\~QS13\ => \OutPad~QS13\,
\~QS14\ => \OutPad~QS14\,
\~QS15\ => \OutPad~QS15\
    );
    CG_RangeBinModulator_7 :
CG_RangeBinModulator PORT MAP(
    CLK => N13,
    DRFM0 => N169,
    DRFM1 => N168,
    DRFM2 => N167,
    DRFM3 => N166,
    DRFM4 => N165,
    Gain0 => N10,
    Gain1 => N7,
    Gain2 => N4,
    Gain3 => N1,
    I0 => N302,
    I1 => N301,
    I2 => N411,
    I3 => N412,
    I4 => N413,
```

I5 => N414,
I6 => N415,
I7 => N416,
I8 => N417,
I9 => N418,
I10 => N419,
I11 => N291,
I12 => N290,
I13 => N289,
I14 => N288,
I15 => N287,
Inc0 => N173,
Inc1 => N172,
Inc2 => N171,
Inc3 => N170,
Inc4 => N174,
IOV => N405,
IS0 => N148,
IS1 => N147,
IS2 => N257,
IS3 => N258,
IS4 => N259,
IS5 => N260,
IS6 => N261,
IS7 => N262,
IS8 => N263,
IS9 => N264,
IS10 => N265,
IS11 => N137,
IS12 => N136,
IS13 => N135,
IS14 => N134,
IS15 => N133,
ISOV => N251,
ODVin => N182,
ODVout => N28,
Oper => N25,
PRB => N1270,
PSV => N19,
Q0 => N318,
Q1 => N317,
Q2 => N316,
Q3 => N315,
Q4 => N314,
Q5 => N313,
Q6 => N312,
Q7 => N311,
Q8 => N310,
Q9 => N309,
Q10 => N435,
Q11 => N436,
Q12 => N437,
Q13 => N438,
Q14 => N439,
Q15 => N303,
QOV => N254,
QS0 => N164,
QS1 => N163,
QS2 => N162,
QS3 => N161,
QS4 => N160,
QS5 => N159,
QS6 => N158,
QS7 => N157,
QS8 => N156,
QS9 => N155,
QS10 => N281,
QS11 => N282,
QS12 => N283,
QS13 => N284,
QS14 => N285,
QS15 => N149,
QSOV => N100,
UNP => N22,
URB => N16,
\~I0\ => N354,
\~I1\ => N232,
\~I2\ => N231,
\~I3\ => N230,
\~I4\ => N229,
\~I5\ => N228,
\~I6\ => N227,
\~I7\ => N226,
\~I8\ => N225,
\~I9\ => N224,
\~I10\ => N223,
\~I11\ => N343,
\~I12\ => N342,
\~I13\ => N341,
\~I14\ => N340,
\~I15\ => N218,
\~IS0\ => N200,
\~IS1\ => N78,
\~IS2\ => N77,
\~IS3\ => N76,
\~IS4\ => N75,
\~IS5\ => N74,
\~IS6\ => N73,
\~IS7\ => N72,
\~IS8\ => N71,
\~IS9\ => N70,
\~IS10\ => N69,
\~IS11\ => N189,
\~IS12\ => N188,
\~IS13\ => N187,
\~IS14\ => N186,
\~IS15\ => N64,
\~Q0\ => N249,
\~Q1\ => N369,
\~Q2\ => N368,
\~Q3\ => N367,
\~Q4\ => N366,

```
                \~Q5\ => N365,
                \~Q6\ => N364,
                \~Q7\ => N363,
                \~Q8\ => N362,
                \~Q9\ => N240,
                \~Q10\ => N239,
                \~Q11\ => N238,
                \~Q12\ => N237,
                \~Q13\ => N236,
                \~Q14\ => N235,
                \~Q15\ => N234,
                \~QS0\ => N95,
                \~QS1\ => N215,
                \~QS2\ => N214,
                \~QS3\ => N213,
                \~QS4\ => N212,
                \~QS5\ => N211,
                \~QS6\ => N210,
                \~QS7\ => N209,
                \~QS8\ => N208,
                \~QS9\ => N86,
                \~QS10\ => N85,
                \~QS11\ => N84,
                \~QS12\ => N83,
                \~QS13\ => N82,
                \~QS14\ => N81,
                \~QS15\ => N80
        );
        CG_RangeBinModulator_6          :
CG_RangeBinModulator   PORT MAP(
                CLK => N13,
                DRFM0 => N169,
                DRFM1 => N168,
                DRFM2 => N167,
                DRFM3 => N166,
                DRFM4 => N165,
                Gain0 => N10,
                Gain1 => N7,
                Gain2 => N4,
                Gain3 => N1,
                I0 => N456,
                I1 => N455,
                I2 => N565,
                I3 => N566,
                I4 => N567,
                I5 => N568,
                I6 => N569,
                I7 => N570,
                I8 => N571,
                I9 => N572,
                I10 => N573,
                I11 => N445,
                I12 => N444,
                I13 => N443,
                I14 => N442,
                I15 => N441,
                Inc0 => N173,
                Inc1 => N172,
                Inc2 => N171,
                Inc3 => N170,
                Inc4 => N174,
                IOV => N559,
                IS0 => N302,
                IS1 => N301,
                IS2 => N411,
                IS3 => N412,
                IS4 => N413,
                IS5 => N414,
                IS6 => N415,
                IS7 => N416,
                IS8 => N417,
                IS9 => N418,
                IS10 => N419,
                IS11 => N291,
                IS12 => N290,
                IS13 => N289,
                IS14 => N288,
                IS15 => N287,
                ISOV => N405,
                ODVin => N336,
                ODVout => N182,
                Oper => N25,
                PRB => N1269,
                PSV => N19,
                Q0 => N472,
                Q1 => N471,
                Q2 => N470,
                Q3 => N469,
                Q4 => N468,
                Q5 => N467,
                Q6 => N466,
                Q7 => N465,
                Q8 => N464,
                Q9 => N463,
                Q10 => N589,
                Q11 => N590,
                Q12 => N591,
                Q13 => N592,
                Q14 => N593,
                Q15 => N457,
                QOV => N408,
                QS0 => N318,
                QS1 => N317,
                QS2 => N316,
                QS3 => N315,
                QS4 => N314,
                QS5 => N313,
                QS6 => N312,
                QS7 => N311,
                QS8 => N310,
                QS9 => N309,
                QS10 => N435,
```

```
QS11 => N436,
QS12 => N437,
QS13 => N438,
QS14 => N439,
QS15 => N303,
QSOV => N254,
UNP => N22,
URB => N16,
\~I0\ => N508,
\~I1\ => N386,
\~I2\ => N385,
\~I3\ => N384,
\~I4\ => N383,
\~I5\ => N382,
\~I6\ => N381,
\~I7\ => N380,
\~I8\ => N379,
\~I9\ => N378,
\~I10\ => N377,
\~I11\ => N497,
\~I12\ => N496,
\~I13\ => N495,
\~I14\ => N494,
\~I15\ => N372,
\~IS0\ => N354,
\~IS1\ => N232,
\~IS2\ => N231,
\~IS3\ => N230,
\~IS4\ => N229,
\~IS5\ => N228,
\~IS6\ => N227,
\~IS7\ => N226,
\~IS8\ => N225,
\~IS9\ => N224,
\~IS10\ => N223,
\~IS11\ => N343,
\~IS12\ => N342,
\~IS13\ => N341,
\~IS14\ => N340,
\~IS15\ => N218,
\~Q0\ => N403,
\~Q1\ => N523,
\~Q2\ => N522,
\~Q3\ => N521,
\~Q4\ => N520,
\~Q5\ => N519,
\~Q6\ => N518,
\~Q7\ => N517,
\~Q8\ => N516,
\~Q9\ => N394,
\~Q10\ => N393,
\~Q11\ => N392,
\~Q12\ => N391,
\~Q13\ => N390,
\~Q14\ => N389,
\~Q15\ => N388,
\~QS0\ => N249,
\~QS1\ => N369,
\~QS2\ => N368,
\~QS3\ => N367,
\~QS4\ => N366,
\~QS5\ => N365,
\~QS6\ => N364,
\~QS7\ => N363,
\~QS8\ => N362,
\~QS9\ => N240,
\~QS10\ => N239,
\~QS11\ => N238,
\~QS12\ => N237,
\~QS13\ => N236,
\~QS14\ => N235,
\~QS15\ => N234
);
CG_RangeBinModulator_5 :
CG_RangeBinModulator PORT MAP(
CLK => N13,
DRFM0 => N169,
DRFM1 => N168,
DRFM2 => N167,
DRFM3 => N166,
DRFM4 => N165,
Gain0 => N10,
Gain1 => N7,
Gain2 => N4,
Gain3 => N1,
I0 => N717,
I1 => N718,
I2 => N608,
I3 => N607,
I4 => N606,
I5 => N605,
I6 => N604,
I7 => N603,
I8 => N602,
I9 => N601,
I10 => N600,
I11 => N728,
I12 => N729,
I13 => N730,
I14 => N731,
I15 => N595,
Inc0 => N173,
Inc1 => N172,
Inc2 => N171,
Inc3 => N170,
Inc4 => N174,
IOV => N561,
IS0 => N456,
IS1 => N455,
IS2 => N565,
IS3 => N566,
IS4 => N567,
```

IS5 => N568,
IS6 => N569,
IS7 => N570,
IS8 => N571,
IS9 => N572,
IS10 => N573,
IS11 => N445,
IS12 => N444,
IS13 => N443,
IS14 => N442,
IS15 => N441,
ISOV => N559,
ODVin => N490,
ODVout => N336,
Oper => N25,
PRB => N1268,
PSV => N19,
Q0 => N733,
Q1 => N734,
Q2 => N735,
Q3 => N736,
Q4 => N737,
Q5 => N738,
Q6 => N739,
Q7 => N740,
Q8 => N741,
Q9 => N742,
Q10 => N616,
Q11 => N615,
Q12 => N614,
Q13 => N613,
Q14 => N612,
Q15 => N611,
QOV => N714,
QS0 => N472,
QS1 => N471,
QS2 => N470,
QS3 => N469,
QS4 => N468,
QS5 => N467,
QS6 => N466,
QS7 => N465,
QS8 => N464,
QS9 => N463,
QS10 => N589,
QS11 => N590,
QS12 => N591,
QS13 => N592,
QS14 => N593,
QS15 => N457,
QSOV => N408,
UNP => N22,
URB => N16,
\~I0\ => N541,
\~I1\ => N661,
\~I2\ => N660,
\~I3\ => N659,
\~I4\ => N658,
\~I5\ => N657,
\~I6\ => N656,
\~I7\ => N655,
\~I8\ => N654,
\~I9\ => N653,
\~I10\ => N652,
\~I11\ => N530,
\~I12\ => N529,
\~I13\ => N528,
\~I14\ => N648,
\~I15\ => N526,
\~IS0\ => N508,
\~IS1\ => N386,
\~IS2\ => N385,
\~IS3\ => N384,
\~IS4\ => N383,
\~IS5\ => N382,
\~IS6\ => N381,
\~IS7\ => N380,
\~IS8\ => N379,
\~IS9\ => N378,
\~IS10\ => N377,
\~IS11\ => N497,
\~IS12\ => N496,
\~IS13\ => N495,
\~IS14\ => N494,
\~IS15\ => N372,
\~Q0\ => N678,
\~Q1\ => N556,
\~Q2\ => N555,
\~Q3\ => N554,
\~Q4\ => N553,
\~Q5\ => N552,
\~Q6\ => N551,
\~Q7\ => N550,
\~Q8\ => N549,
\~Q9\ => N669,
\~Q10\ => N668,
\~Q11\ => N667,
\~Q12\ => N666,
\~Q13\ => N665,
\~Q14\ => N543,
\~Q15\ => N542,
\~QS0\ => N403,
\~QS1\ => N523,
\~QS2\ => N522,
\~QS3\ => N521,
\~QS4\ => N520,
\~QS5\ => N519,
\~QS6\ => N518,
\~QS7\ => N517,
\~QS8\ => N516,
\~QS9\ => N394,
\~QS10\ => N393,

```
                \~QS11\ => N392,
                \~QS12\ => N391,
                \~QS13\ => N390,
                \~QS14\ => N389,
                \~QS15\ => N388
        );
        CG_RangeBinModulator_4          :
CG_RangeBinModulator   PORT MAP(
                CLK => N1254,
                DRFM0 => N785,
                DRFM1 => N784,
                DRFM2 => N783,
                DRFM3 => N782,
                DRFM4 => N781,
                Gain0 => N1256,
                Gain1 => N1259,
                Gain2 => N1258,
                Gain3 => N1257,
                I0 => N764,
                I1 => N872,
                I2 => N762,
                I3 => N761,
                I4 => N760,
                I5 => N759,
                I6 => N758,
                I7 => N757,
                I8 => N756,
                I9 => N755,
                I10 => N754,
                I11 => N882,
                I12 => N883,
                I13 => N884,
                I14 => N885,
                I15 => N886,
                Inc0 => N789,
                Inc1 => N788,
                Inc2 => N787,
                Inc3 => N786,
                Inc4 => N790,
                IOV => N867,
                IS0 => N717,
                IS1 => N718,
                IS2 => N608,
                IS3 => N607,
                IS4 => N606,
                IS5 => N605,
                IS6 => N604,
                IS7 => N603,
                IS8 => N602,
                IS9 => N601,
                IS10 => N600,
                IS11 => N728,
                IS12 => N729,
                IS13 => N730,
                IS14 => N731,
                IS15 => N595,
                ISOV => N561,
                ODVin => N644,
                ODVout => N490,
                Oper => N955,
                PRB => N1267,
                PSV => N954,
                Q0 => N780,
                Q1 => N888,
                Q2 => N889,
                Q3 => N890,
                Q4 => N891,
                Q5 => N892,
                Q6 => N893,
                Q7 => N894,
                Q8 => N895,
                Q9 => N896,
                Q10 => N770,
                Q11 => N769,
                Q12 => N768,
                Q13 => N767,
                Q14 => N766,
                Q15 => N902,
                QOV => N716,
                QS0 => N733,
                QS1 => N734,
                QS2 => N735,
                QS3 => N736,
                QS4 => N737,
                QS5 => N738,
                QS6 => N739,
                QS7 => N740,
                QS8 => N741,
                QS9 => N742,
                QS10 => N616,
                QS11 => N615,
                QS12 => N614,
                QS13 => N613,
                QS14 => N612,
                QS15 => N611,
                QSOV => N714,
                UNP => N1101,
                URB => N1020,
                \~I0\ => N695,
                \~I1\ => N815,
                \~I2\ => N814,
                \~I3\ => N813,
                \~I4\ => N812,
                \~I5\ => N811,
                \~I6\ => N810,
                \~I7\ => N809,
                \~I8\ => N808,
                \~I9\ => N807,
                \~I10\ => N806,
                \~I11\ => N684,
                \~I12\ => N683,
                \~I13\ => N682,
```

```
		\~I14\ => N681,
		\~I15\ => N817,
		\~IS0\ => N541,
		\~IS1\ => N661,
		\~IS2\ => N660,
		\~IS3\ => N659,
		\~IS4\ => N658,
		\~IS5\ => N657,
		\~IS6\ => N656,
		\~IS7\ => N655,
		\~IS8\ => N654,
		\~IS9\ => N653,
		\~IS10\ => N652,
		\~IS11\ => N530,
		\~IS12\ => N529,
		\~IS13\ => N528,
		\~IS14\ => N648,
		\~IS15\ => N526,
		\~Q0\ => N832,
		\~Q1\ => N710,
		\~Q2\ => N709,
		\~Q3\ => N708,
		\~Q4\ => N707,
		\~Q5\ => N706,
		\~Q6\ => N705,
		\~Q7\ => N704,
		\~Q8\ => N703,
		\~Q9\ => N823,
		\~Q10\ => N822,
		\~Q11\ => N821,
		\~Q12\ => N820,
		\~Q13\ => N819,
		\~Q14\ => N818,
		\~Q15\ => N833,
		\~QS0\ => N678,
		\~QS1\ => N556,
		\~QS2\ => N555,
		\~QS3\ => N554,
		\~QS4\ => N553,
		\~QS5\ => N552,
		\~QS6\ => N551,
		\~QS7\ => N550,
		\~QS8\ => N549,
		\~QS9\ => N669,
		\~QS10\ => N668,
		\~QS11\ => N667,
		\~QS12\ => N666,
		\~QS13\ => N665,
		\~QS14\ => N543,
		\~QS15\ => N542
	);
	CG_RangeBinModulator_3		:
CG_RangeBinModulator   PORT MAP(
		CLK => N1254,
		DRFM0 => N785,
		DRFM1 => N784,
		DRFM2 => N783,
		DRFM3 => N782,
		DRFM4 => N781,
		Gain0 => N1256,
		Gain1 => N1259,
		Gain2 => N1258,
		Gain3 => N1257,
		I0 => N918,
		I1 => N917,
		I2 => N1027,
		I3 => N1028,
		I4 => N1029,
		I5 => N1030,
		I6 => N1031,
		I7 => N1032,
		I8 => N1033,
		I9 => N1034,
		I10 => N1035,
		I11 => N907,
		I12 => N906,
		I13 => N905,
		I14 => N904,
		I15 => N903,
		Inc0 => N789,
		Inc1 => N788,
		Inc2 => N787,
		Inc3 => N786,
		Inc4 => N790,
		IOV => N1021,
		IS0 => N764,
		IS1 => N872,
		IS2 => N762,
		IS3 => N761,
		IS4 => N760,
		IS5 => N759,
		IS6 => N758,
		IS7 => N757,
		IS8 => N756,
		IS9 => N755,
		IS10 => N754,
		IS11 => N882,
		IS12 => N883,
		IS13 => N884,
		IS14 => N885,
		IS15 => N886,
		ISOV => N867,
		ODVin => N798,
		ODVout => N644,
		Oper => N955,
		PRB => N1266,
		PSV => N954,
		Q0 => N934,
		Q1 => N933,
		Q2 => N932,
		Q3 => N931,
		Q4 => N930,
```

```
Q5 => N929,
Q6 => N928,
Q7 => N927,
Q8 => N926,
Q9 => N925,
Q10 => N1051,
Q11 => N1052,
Q12 => N1053,
Q13 => N1054,
Q14 => N1055,
Q15 => N919,
QOV => N870,
QS0 => N780,
QS1 => N888,
QS2 => N889,
QS3 => N890,
QS4 => N891,
QS5 => N892,
QS6 => N893,
QS7 => N894,
QS8 => N895,
QS9 => N896,
QS10 => N770,
QS11 => N769,
QS12 => N768,
QS13 => N767,
QS14 => N766,
QS15 => N902,
QSOV => N716,
UNP => N1101,
URB => N1020,
\~I0\ => N970,
\~I1\ => N848,
\~I2\ => N847,
\~I3\ => N846,
\~I4\ => N845,
\~I5\ => N844,
\~I6\ => N843,
\~I7\ => N842,
\~I8\ => N841,
\~I9\ => N840,
\~I10\ => N839,
\~I11\ => N959,
\~I12\ => N958,
\~I13\ => N957,
\~I14\ => N956,
\~I15\ => N834,
\~IS0\ => N695,
\~IS1\ => N815,
\~IS2\ => N814,
\~IS3\ => N813,
\~IS4\ => N812,
\~IS5\ => N811,
\~IS6\ => N810,
\~IS7\ => N809,
\~IS8\ => N808,
\~IS9\ => N807,
\~IS10\ => N806,
\~IS11\ => N684,
\~IS12\ => N683,
\~IS13\ => N682,
\~IS14\ => N681,
\~IS15\ => N817,
\~Q0\ => N865,
\~Q1\ => N985,
\~Q2\ => N984,
\~Q3\ => N983,
\~Q4\ => N982,
\~Q5\ => N981,
\~Q6\ => N980,
\~Q7\ => N979,
\~Q8\ => N978,
\~Q9\ => N856,
\~Q10\ => N855,
\~Q11\ => N854,
\~Q12\ => N853,
\~Q13\ => N852,
\~Q14\ => N851,
\~Q15\ => N850,
\~QS0\ => N832,
\~QS1\ => N710,
\~QS2\ => N709,
\~QS3\ => N708,
\~QS4\ => N707,
\~QS5\ => N706,
\~QS6\ => N705,
\~QS7\ => N704,
\~QS8\ => N703,
\~QS9\ => N823,
\~QS10\ => N822,
\~QS11\ => N821,
\~QS12\ => N820,
\~QS13\ => N819,
\~QS14\ => N818,
\~QS15\ => N833
);
CG_RangeBinModulator_1 : CG_RangeBinModulator PORT MAP(
CLK => N1254,
DRFM0 => N785,
DRFM1 => N784,
DRFM2 => N783,
DRFM3 => N782,
DRFM4 => N781,
Gain0 => N1256,
Gain1 => N1259,
Gain2 => N1258,
Gain3 => N1257,
I0 => N1072,
I1 => N1071,
I2 => N1181,
I3 => N1182,
```

I4 => N1183,
I5 => N1184,
I6 => N1185,
I7 => N1186,
I8 => N1187,
I9 => N1188,
I10 => N1189,
I11 => N1061,
I12 => N1060,
I13 => N1059,
I14 => N1058,
I15 => N1057,
Inc0 => N789,
Inc1 => N788,
Inc2 => N787,
Inc3 => N786,
Inc4 => N790,
IOV => N1175,
IS0 => N918,
IS1 => N917,
IS2 => N1027,
IS3 => N1028,
IS4 => N1029,
IS5 => N1030,
IS6 => N1031,
IS7 => N1032,
IS8 => N1033,
IS9 => N1034,
IS10 => N1035,
IS11 => N907,
IS12 => N906,
IS13 => N905,
IS14 => N904,
IS15 => N903,
ISOV => N1021,
ODVin => N952,
ODVout => N798,
Oper => N955,
PRB => N1265,
PSV => N954,
Q0 => N1088,
Q1 => N1087,
Q2 => N1086,
Q3 => N1085,
Q4 => N1084,
Q5 => N1083,
Q6 => N1082,
Q7 => N1081,
Q8 => N1080,
Q9 => N1079,
Q10 => N1205,
Q11 => N1206,
Q12 => N1207,
Q13 => N1208,
Q14 => N1209,
Q15 => N1073,
QOV => N1024,
QS0 => N934,
QS1 => N933,
QS2 => N932,
QS3 => N931,
QS4 => N930,
QS5 => N929,
QS6 => N928,
QS7 => N927,
QS8 => N926,
QS9 => N925,
QS10 => N1051,
QS11 => N1052,
QS12 => N1053,
QS13 => N1054,
QS14 => N1055,
QS15 => N919,
QSOV => N870,
UNP => N1101,
URB => N1020,
\~I0\ => N1124,
\~I1\ => N1002,
\~I2\ => N1001,
\~I3\ => N1000,
\~I4\ => N999,
\~I5\ => N998,
\~I6\ => N997,
\~I7\ => N996,
\~I8\ => N995,
\~I9\ => N994,
\~I10\ => N993,
\~I11\ => N1113,
\~I12\ => N1112,
\~I13\ => N1111,
\~I14\ => N1110,
\~I15\ => N988,
\~IS0\ => N970,
\~IS1\ => N848,
\~IS2\ => N847,
\~IS3\ => N846,
\~IS4\ => N845,
\~IS5\ => N844,
\~IS6\ => N843,
\~IS7\ => N842,
\~IS8\ => N841,
\~IS9\ => N840,
\~IS10\ => N839,
\~IS11\ => N959,
\~IS12\ => N958,
\~IS13\ => N957,
\~IS14\ => N956,
\~IS15\ => N834,
\~Q0\ => N1019,
\~Q1\ => N1139,
\~Q2\ => N1138,
\~Q3\ => N1137,

```
                \~Q4\ => N1136,
                \~Q5\ => N1135,
                \~Q6\ => N1134,
                \~Q7\ => N1133,
                \~Q8\ => N1132,
                \~Q9\ => N1010,
                \~Q10\ => N1009,
                \~Q11\ => N1008,
                \~Q12\ => N1007,
                \~Q13\ => N1006,
                \~Q14\ => N1005,
                \~Q15\ => N1004,
                \~QS0\ => N865,
                \~QS1\ => N985,
                \~QS2\ => N984,
                \~QS3\ => N983,
                \~QS4\ => N982,
                \~QS5\ => N981,
                \~QS6\ => N980,
                \~QS7\ => N979,
                \~QS8\ => N978,
                \~QS9\ => N856,
                \~QS10\ => N855,
                \~QS11\ => N854,
                \~QS12\ => N853,
                \~QS13\ => N852,
                \~QS14\ => N851,
                \~QS15\ => N850
        );
        CG_RangeBinModulator_2          :
CG_RangeBinModulator   PORT MAP(
                CLK => N1254,
                DRFM0 => N785,
                DRFM1 => N784,
                DRFM2 => N783,
                DRFM3 => N782,
                DRFM4 => N781,
                Gain0 => N1256,
                Gain1 => N1259,
                Gain2 => N1258,
                Gain3 => N1257,
                I0 => InPadI0,
                I1 => InPadI1,
                I2 => InPadI2,
                I3 => InPadI3,
                I4 => InPadI4,
                I5 => InPadI5,
                I6 => InPadI6,
                I7 => InPadI7,
                I8 => InPadI8,
                I9 => InPadI9,
                I10 => InPadI10,
                I11 => InPadI11,
                I12 => InPadI12,
                I13 => InPadI13,
                I14 => InPadI14,
                I15 => InPadI15,
                Inc0 => N789,
                Inc1 => N788,
                Inc2 => N787,
                Inc3 => N786,
                Inc4 => N790,
                IOV => InPadIOV,
                IS0 => N1072,
                IS1 => N1071,
                IS2 => N1181,
                IS3 => N1182,
                IS4 => N1183,
                IS5 => N1184,
                IS6 => N1185,
                IS7 => N1186,
                IS8 => N1187,
                IS9 => N1188,
                IS10 => N1189,
                IS11 => N1061,
                IS12 => N1060,
                IS13 => N1059,
                IS14 => N1058,
                IS15 => N1057,
                ISOV => N1175,
                ODVin => ODVin,
                ODVout => N952,
                Oper => N955,
                PRB => N1264,
                PSV => N954,
                Q0 => InPadQ0,
                Q1 => InPadQ1,
                Q2 => InPadQ2,
                Q3 => InPadQ3,
                Q4 => InPadQ4,
                Q5 => InPadQ5,
                Q6 => InPadQ6,
                Q7 => InPadQ7,
                Q8 => InPadQ8,
                Q9 => InPadQ9,
                Q10 => InPadQ10,
                Q11 => InPadQ11,
                Q12 => InPadQ12,
                Q13 => InPadQ13,
                Q14 => InPadQ14,
                Q15 => InPadQ15,
                QOV => InPadQOV,
                QS0 => N1088,
                QS1 => N1087,
                QS2 => N1086,
                QS3 => N1085,
                QS4 => N1084,
                QS5 => N1083,
                QS6 => N1082,
                QS7 => N1081,
                QS8 => N1080,
                QS9 => N1079,
```

```
QS10 => N1205,
QS11 => N1206,
QS12 => N1207,
QS13 => N1208,
QS14 => N1209,
QS15 => N1073,
QSOV => N1024,
UNP => N1101,
URB => N1020,
\~I0\ => \InPad~I0\,
\~I1\ => \InPad~I1\,
\~I2\ => \InPad~I2\,
\~I3\ => \InPad~I3\,
\~I4\ => \InPad~I4\,
\~I5\ => \InPad~I5\,
\~I6\ => \InPad~I6\,
\~I7\ => \InPad~I7\,
\~I8\ => \InPad~I8\,
\~I9\ => \InPad~I9\,
\~I10\ => \InPad~I10\,
\~I11\ => \InPad~I11\,
\~I12\ => \InPad~I12\,
\~I13\ => \InPad~I13\,
\~I14\ => \InPad~I14\,
\~I15\ => \InPad~I15\,
\~IS0\ => N1124,
\~IS1\ => N1002,
\~IS2\ => N1001,
\~IS3\ => N1000,
\~IS4\ => N999,
\~IS5\ => N998,
\~IS6\ => N997,
\~IS7\ => N996,
\~IS8\ => N995,
\~IS9\ => N994,
\~IS10\ => N993,
\~IS11\ => N1113,
\~IS12\ => N1112,
\~IS13\ => N1111,
\~IS14\ => N1110,
\~IS15\ => N988,
\~Q0\ => \InPad~Q0\,
\~Q1\ => \InPad~Q1\,
\~Q2\ => \InPad~Q2\,
\~Q3\ => \InPad~Q3\,
\~Q4\ => \InPad~Q4\,
\~Q5\ => \InPad~Q5\,
\~Q6\ => \InPad~Q6\,
\~Q7\ => \InPad~Q7\,
\~Q8\ => \InPad~Q8\,
\~Q9\ => \InPad~Q9\,
\~Q10\ => \InPad~Q10\,
\~Q11\ => \InPad~Q11\,
\~Q12\ => \InPad~Q12\,
\~Q13\ => \InPad~Q13\,
\~Q14\ => \InPad~Q14\,
\~Q15\ => \InPad~Q15\,
\~QS0\ => N1019,
\~QS1\ => N1139,
\~QS2\ => N1138,
\~QS3\ => N1137,
\~QS4\ => N1136,
\~QS5\ => N1135,
\~QS6\ => N1134,
\~QS7\ => N1133,
\~QS8\ => N1132,
\~QS9\ => N1010,
\~QS10\ => N1009,
\~QS11\ => N1008,
\~QS12\ => N1007,
\~QS13\ => N1006,
\~QS14\ => N1005,
\~QS15\ => N1004
    );
    BO_3to8DECODER_1 : BO_3to8DECODER PORT MAP(
        D0 => N1264,
        D1 => N1265,
        D2 => N1266,
        D3 => N1267,
        D4 => N1268,
        D5 => N1269,
        D6 => N1270,
        D7 => N1271,
        Enable => ENABLE,
        Select0 => RBinSelect0,
        Select1 => RBinSelect1,
        Select2 => RBinSelect2
    );
    DTM_SigFanout_20 : DTM_SigFanout PORT MAP(
        SigIn => Inc0,
        SigOut1 => N173,
        SigOut2 => N789
    );
    DTM_SigFanout_19 : DTM_SigFanout PORT MAP(
        SigIn => Inc1,
        SigOut1 => N172,
        SigOut2 => N788
    );
    DTM_SigFanout_18 : DTM_SigFanout PORT MAP(
        SigIn => Inc2,
        SigOut1 => N171,
        SigOut2 => N787
    );
    DTM_SigFanout_16 : DTM_SigFanout PORT MAP(
        SigIn => Inc4,
        SigOut1 => N174,
        SigOut2 => N790
```

```
        );
        DTM_SigFanout_14 : DTM_SigFanout
PORT MAP(
                SigIn => DRFM1,
                SigOut1 => N168,
                SigOut2 => N784
        );
        DTM_SigFanout_13 : DTM_SigFanout
PORT MAP(
                SigIn => DRFM2,
                SigOut1 => N167,
                SigOut2 => N783
        );
        DTM_SigFanout_12 : DTM_SigFanout
PORT MAP(
                SigIn => DRFM3,
                SigOut1 => N166,
                SigOut2 => N782
        );
        DTM_SigFanout_11 : DTM_SigFanout
PORT MAP(
                SigIn => DRFM4,
                SigOut1 => N165,
                SigOut2 => N781
        );
        DTM_SigFanout_17 : DTM_SigFanout
PORT MAP(
                SigIn => Inc3,
                SigOut1 => N170,
                SigOut2 => N786
        );
        DTM_SigFanout_1  :  DTM_SigFanout
PORT MAP(
                SigIn => DRFM0,
                SigOut1 => N169,
                SigOut2 => N785
        );
        END structural;
```

B. TEST BENCH FOR THE 8 RANGE BIN

```
--------------------------------------------------
--
-- Title      : Test Bench for dtm_8rbps
-- Design     : HB_8_RB
-- Author     : Hakan Bergon
-- Company    : NPS
--
--------------------------------------------------
--
-- File       : $DSN\src\TestBench\dtm_8rbps_TB.vhd
-- Generated  : 8/19/2002, 5:09 PM
-- From       : $DSN\src\dtm_8rbps.vhd
-- By         : Active-HDL Built-in Test Bench Generator ver. 1.2s
--
--------------------------------------------------
--
-- Description : Automatically generated Test Bench for dtm_8rbps_tb
--
--------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;

        -- Add your library and packages declaration here ...

entity dtm_8rbps_tb is
end dtm_8rbps_tb;

architecture TB_ARCHITECTURE of dtm_8rbps_tb is
        -- Component declaration of the tested unit
        component dtm_8rbps
        port(
        CLK : in std_logic;
        DRFM0 : in std_logic;
        DRFM1 : in std_logic;
        DRFM2 : in std_logic;
        DRFM3 : in std_logic;
        DRFM4 : in std_logic;
        ENABLE : in std_logic;
        Gain0 : in std_logic;
        Gain1 : in std_logic;
        Gain2 : in std_logic;
        Gain3 : in std_logic;
        Inc0 : in std_logic;
        Inc1 : in std_logic;
        Inc2 : in std_logic;
        Inc3 : in std_logic;
        Inc4 : in std_logic;
        InPadI0 : in std_logic;
        InPadI1 : in std_logic;
        InPadI2 : in std_logic;
        InPadI3 : in std_logic;
        InPadI4 : in std_logic;
        InPadI5 : in std_logic;
        InPadI6 : in std_logic;
        InPadI7 : in std_logic;
        InPadI8 : in std_logic;
        InPadI9 : in std_logic;
        InPadI10 : in std_logic;
        InPadI11 : in std_logic;
        InPadI12 : in std_logic;
        InPadI13 : in std_logic;
        InPadI14 : in std_logic;
        InPadI15 : in std_logic;
        InPadIOV : in std_logic;
        InPadQ0 : in std_logic;
        InPadQ1 : in std_logic;
        InPadQ2 : in std_logic;
        InPadQ3 : in std_logic;
        InPadQ4 : in std_logic;
        InPadQ5 : in std_logic;
        InPadQ6 : in std_logic;
        InPadQ7 : in std_logic;
        InPadQ8 : in std_logic;
        InPadQ9 : in std_logic;
        InPadQ10 : in std_logic;
        InPadQ11 : in std_logic;
        InPadQ12 : in std_logic;
        InPadQ13 : in std_logic;
        InPadQ14 : in std_logic;
        InPadQ15 : in std_logic;
        InPadQOV : in std_logic;
        \InPad~I0\ : in std_logic;
        \InPad~I1\ : in std_logic;
        \InPad~I2\ : in std_logic;
        \InPad~I3\ : in std_logic;
        \InPad~I4\ : in std_logic;
        \InPad~I5\ : in std_logic;
        \InPad~I6\ : in std_logic;
        \InPad~I7\ : in std_logic;
        \InPad~I8\ : in std_logic;
        \InPad~I9\ : in std_logic;
        \InPad~I10\ : in std_logic;
        \InPad~I11\ : in std_logic;
        \InPad~I12\ : in std_logic;
        \InPad~I13\ : in std_logic;
        \InPad~I14\ : in std_logic;
        \InPad~I15\ : in std_logic;
        \InPad~Q0\ : in std_logic;
        \InPad~Q1\ : in std_logic;
        \InPad~Q2\ : in std_logic;
        InPad~Q3\ : in std_logic;
```

```
\InPad~Q4\ : in std_logic;
\InPad~Q5\ : in std_logic;
\InPad~Q6\ : in std_logic;
\InPad~Q7\ : in std_logic;
\InPad~Q8\ : in std_logic;
\InPad~Q9\ : in std_logic;
\InPad~Q10\ : in std_logic;
\InPad~Q11\ : in std_logic;
\InPad~Q12\ : in std_logic;
\InPad~Q13\ : in std_logic;
\InPad~Q14\ : in std_logic;
\InPad~Q15\ : in std_logic;
ODVin : in std_logic;
ODVout : out std_logic;
Oper : in std_logic;
OutPadIS0 : out std_logic;
OutPadIS1 : out std_logic;
OutPadIS2 : out std_logic;
OutPadIS3 : out std_logic;
OutPadIS4 : out std_logic;
OutPadIS5 : out std_logic;
OutPadIS6 : out std_logic;
OutPadIS7 : out std_logic;
OutPadIS8 : out std_logic;
OutPadIS9 : out std_logic;
OutPadIS10 : out std_logic;
OutPadIS11 : out std_logic;
OutPadIS12 : out std_logic;
OutPadIS13 : out std_logic;
OutPadIS14 : out std_logic;
OutPadIS15 : out std_logic;
OutPadISOV : out std_logic;
OutPadQS0 : out std_logic;
OutPadQS1 : out std_logic;
OutPadQS2 : out std_logic;
OutPadQS3 : out std_logic;
OutPadQS4 : out std_logic;
OutPadQS5 : out std_logic;
OutPadQS6 : out std_logic;
OutPadQS7 : out std_logic;
OutPadQS8 : out std_logic;
OutPadQS9 : out std_logic;
OutPadQS10 : out std_logic;
OutPadQS11 : out std_logic;
OutPadQS12 : out std_logic;
OutPadQS13 : out std_logic;
OutPadQS14 : out std_logic;
OutPadQS15 : out std_logic;
OutPadQSOV : out std_logic;
\OutPad~IS0\ : out std_logic;
\OutPad~IS1\ : out std_logic;
\OutPad~IS2\ : out std_logic;
\OutPad~IS3\ : out std_logic;
\OutPad~IS4\ : out std_logic;
\OutPad~IS5\ : out std_logic;
OutPad~IS6\ : out std_logic;
\OutPad~IS7\ : out std_logic;
\OutPad~IS8\ : out std_logic;
\OutPad~IS9\ : out std_logic;
\OutPad~IS10\ : out std_logic;
\OutPad~IS11\ : out std_logic;
\OutPad~IS12\ : out std_logic;
\OutPad~IS13\ : out std_logic;
\OutPad~IS14\ : out std_logic;
\OutPad~IS15\ : out std_logic;
\OutPad~QS0\ : out std_logic;
\OutPad~QS1\ : out std_logic;
\OutPad~QS2\ : out std_logic;
\OutPad~QS3\ : out std_logic;
\OutPad~QS4\ : out std_logic;
\OutPad~QS5\ : out std_logic;
\OutPad~QS6\ : out std_logic;
\OutPad~QS7\ : out std_logic;
\OutPad~QS8\ : out std_logic;
\OutPad~QS9\ : out std_logic;
\OutPad~QS10\: out std_logic;
\OutPad~QS11\: out std_logic;
\OutPad~QS12\: out std_logic;
\OutPad~QS13\: out std_logic;
\OutPad~QS14\: out std_logic;
\OutPad~QS15\: out std_logic;
PSV : in std_logic;
RBinSelect0 : in std_logic;
RBinSelect1 : in std_logic;
RBinSelect2 : in std_logic;
UNP : in std_logic;
URB : in std_logic );
end component;

-- Stimulus signals - signals mapped to
the input and inout ports of tested entity
signal CLK : std_logic;
signal DRFM0 : std_logic;
signal DRFM1 : std_logic;
signal DRFM2 : std_logic;
signal DRFM3 : std_logic;
signal DRFM4 : std_logic;
signal ENABLE : std_logic;
signal Gain0 : std_logic;
signal Gain1 : std_logic;
signal Gain2 : std_logic;
signal Gain3 : std_logic;
signal Inc0 : std_logic;
signal Inc1 : std_logic;
signal Inc2 : std_logic;
signal Inc3 : std_logic;
signal Inc4 : std_logic;
signal InPadI0 : std_logic;
signal InPadI1 : std_logic;
signal InPadI2 : std_logic;
signal InPadI3 : std_logic;
signal InPadI4 : std_logic;
```

```
signal InPadI5 : std_logic;
signal InPadI6 : std_logic;
signal InPadI7 : std_logic;
signal InPadI8 : std_logic;
signal InPadI9 : std_logic;
signal InPadI10 : std_logic;
signal InPadI11 : std_logic;
signal InPadI12 : std_logic;
signal InPadI13 : std_logic;
signal InPadI14 : std_logic;
signal InPadI15 : std_logic;
signal InPadIOV : std_logic;
signal InPadQ0 : std_logic;
signal InPadQ1 : std_logic;
signal InPadQ2 : std_logic;
signal InPadQ3 : std_logic;
signal InPadQ4 : std_logic;
signal InPadQ5 : std_logic;
signal InPadQ6 : std_logic;
signal InPadQ7 : std_logic;
signal InPadQ8 : std_logic;
signal InPadQ9 : std_logic;
signal InPadQ10 : std_logic;
signal InPadQ11 : std_logic;
signal InPadQ12 : std_logic;
signal InPadQ13 : std_logic;
signal InPadQ14 : std_logic;
signal InPadQ15 : std_logic;
signal InPadQOV : std_logic;
signal \InPad~I0\ : std_logic;
signal \InPad~I1\ : std_logic;
signal \InPad~I2\ : std_logic;
signal \InPad~I3\ : std_logic;
signal \InPad~I4\ : std_logic;
signal \InPad~I5\ : std_logic;
signal \InPad~I6\ : std_logic;
signal \InPad~I7\ : std_logic;
signal \InPad~I8\ : std_logic;
signal \InPad~I9\ : std_logic;
signal \InPad~I10\ : std_logic;
signal \InPad~I11\ : std_logic;
signal \InPad~I12\ : std_logic;
signal \InPad~I13\ : std_logic;
signal \InPad~I14\ : std_logic;
signal \InPad~I15\ : std_logic;
signal \InPad~Q0\ : std_logic;
signal \InPad~Q1\ : std_logic;
signal \InPad~Q2\ : std_logic;
signal \InPad~Q3\ : std_logic;
signal \InPad~Q4\ : std_logic;
signal \InPad~Q5\ : std_logic;
signal \InPad~Q6\ : std_logic;
signal \InPad~Q7\ : std_logic;
signal \InPad~Q8\ : std_logic;
signal \InPad~Q9\ : std_logic;
signal \InPad~Q10\ : std_logic;
signal \InPad~Q11\ : std_logic;
signal \InPad~Q12\ : std_logic;
signal \InPad~Q13\ : std_logic;
signal \InPad~Q14\ : std_logic;
signal \InPad~Q15\ : std_logic;
signal ODVin : std_logic;
signal Oper : std_logic;
signal PSV : std_logic;
signal RBinSelect0 : std_logic;
signal RBinSelect1 : std_logic;
signal RBinSelect2 : std_logic;
signal UNP : std_logic;
signal URB : std_logic;
-- Observed signals - signals mapped to the output ports of tested entity
signal ODVout : std_logic;
signal OutPadIS0 : std_logic;
signal OutPadIS1 : std_logic;
signal OutPadIS2 : std_logic;
signal OutPadIS3 : std_logic;
signal OutPadIS4 : std_logic;
signal OutPadIS5 : std_logic;
signal OutPadIS6 : std_logic;
signal OutPadIS7 : std_logic;
signal OutPadIS8 : std_logic;
signal OutPadIS9 : std_logic;
signal OutPadIS10 : std_logic;
signal OutPadIS11 : std_logic;
signal OutPadIS12 : std_logic;
signal OutPadIS13 : std_logic;
signal OutPadIS14 : std_logic;
signal OutPadIS15 : std_logic;
signal OutPadISOV : std_logic;
signal OutPadQS0 : std_logic;
signal OutPadQS1 : std_logic;
signal OutPadQS2 : std_logic;
signal OutPadQS3 : std_logic;
signal OutPadQS4 : std_logic;
signal OutPadQS5 : std_logic;
signal OutPadQS6 : std_logic;
signal OutPadQS7 : std_logic;
signal OutPadQS8 : std_logic;
signal OutPadQS9 : std_logic;
signal OutPadQS10: std_logic;
signal OutPadQS11: std_logic;
signal OutPadQS12: std_logic;
signal OutPadQS13: std_logic;
signal OutPadQS14: std_logic;
signal OutPadQS15: std_logic;
signal OutPadQSOV:std_logic;
signal \OutPad~IS0\: std_logic;
signal \OutPad~IS1\: std_logic;
signal \OutPad~IS2\: std_logic;
signal \OutPad~IS3\: std_logic;
signal \OutPad~IS4\: std_logic;
```

```
        signal \OutPad~IS5\: std_logic;
        signal \OutPad~IS6\: std_logic;
        signal \OutPad~IS7\: std_logic;
        signal \OutPad~IS8\: std_logic;
        signal \OutPad~IS9\: std_logic;
        signal\OutPad~IS10\:std_logic;
        signal \OutPad~IS11\:std_logic;
        signal \OutPad~IS12\:std_logic;
        signal \OutPad~IS13\:std_logic;
        signal \OutPad~IS14\ : std_logic;
        signal \OutPad~IS15\ : std_logic;
        signal \OutPad~QS0\ : std_logic;
        signal \OutPad~QS1\ : std_logic;
        signal \OutPad~QS2\ : std_logic;
        signal \OutPad~QS3\ : std_logic;
        signal \OutPad~QS4\ : std_logic;
        signal \OutPad~QS5\ : std_logic;
        signal \OutPad~QS6\ : std_logic;
        signal \OutPad~QS7\ : std_logic;
        signal \OutPad~QS8\ : std_logic;
        signal \OutPad~QS9\ : std_logic;
        signal \OutPad~QS10\ : std_logic;
        signal \OutPad~QS11\ : std_logic;
        signal \OutPad~QS12\ : std_logic;
        signal \OutPad~QS13\ : std_logic;
        signal \OutPad~QS14\ : std_logic;
        signal \OutPad~QS15\ : std_logic;
        --Signal is used to stop clock signal
geerators
        signal                  END_SIM:
BOOLEAN:=FALSE;

                -- Add your code here ...

        begin

                -- Unit Under Test port map
                UUT : dtm_8rbps
                        port map (
                        CLK => CLK,
                        DRFM0 => DRFM0,
                        DRFM1 => DRFM1,
                        DRFM2 => DRFM2,
                        DRFM3 => DRFM3,
                        DRFM4 => DRFM4,
                ENABLE => ENABLE,
                        Gain0 => Gain0,
                        Gain1 => Gain1,
                        Gain2 => Gain2,
                        Gain3 => Gain3,
                        Inc0 => Inc0,
                        Inc1 => Inc1,
                        Inc2 => Inc2,
                        Inc3 => Inc3,
                        Inc4 => Inc4,
                        InPadI0 => InPadI0,
                        InPadI1 => InPadI1,
                        InPadI2 => InPadI2,
                        InPadI3 => InPadI3,
                        InPadI4 => InPadI4,
                        InPadI5 => InPadI5,
                        InPadI6 => InPadI6,
                        InPadI7 => InPadI7,
                        InPadI8 => InPadI8,
                        InPadI9 => InPadI9,
                InPadI10 => InPadI10,
                InPadI11 => InPadI11,
                InPadI12 => InPadI12,
                InPadI13 => InPadI13,
                InPadI14 => InPadI14,
                InPadI15 => InPadI15,
                InPadIOV => InPadIOV,
                InPadQ0 => InPadQ0,
                InPadQ1 => InPadQ1,
                InPadQ2 => InPadQ2,
                InPadQ3 => InPadQ3,
                InPadQ4 => InPadQ4,
                InPadQ5 => InPadQ5,
                InPadQ6 => InPadQ6,
                InPadQ7 => InPadQ7,
                InPadQ8 => InPadQ8,
                InPadQ9 => InPadQ9,
                InPadQ10 => InPadQ10,
                InPadQ11 => InPadQ11,
                InPadQ12 => InPadQ12,
                InPadQ13 => InPadQ13,
                InPadQ14 => InPadQ14,
                InPadQ15 => InPadQ15,
                InPadQOV => InPadQOV,
                \InPad~I0\ => \InPad~I0\,
                \InPad~I1\ => \InPad~I1\,
                \InPad~I2\ => \InPad~I2\,
                \InPad~I3\ => \InPad~I3\,
                \InPad~I4\ => \InPad~I4\,
                \InPad~I5\ => \InPad~I5\,
                \InPad~I6\ => \InPad~I6\,
                \InPad~I7\ => \InPad~I7\,
                \InPad~I8\ => \InPad~I8\,
                \InPad~I9\ => \InPad~I9\,
                \InPad~I10\ => \InPad~I10\,
                \InPad~I11\ => \InPad~I11\,
                \InPad~I12\ => \InPad~I12\,
                \InPad~I13\ => \InPad~I13\,
                \InPad~I14\ => \InPad~I14\,
                \InPad~I15\ => \InPad~I15\,
                \InPad~Q0\ => \InPad~Q0\,
                \InPad~Q1\ => \InPad~Q1\,
                \InPad~Q2\ => \InPad~Q2\,
                \InPad~Q3\ => \InPad~Q3\,
                \InPad~Q4\ => \InPad~Q4\,
                \InPad~Q5\ => \InPad~Q5\,
                \InPad~Q6\ => \InPad~Q6\,
```

```
        \InPad~Q7\ => \InPad~Q7\,
        \InPad~Q8\ => \InPad~Q8\,
        \InPad~Q9\ => \InPad~Q9\,
        \InPad~Q10\ => \InPad~Q10\,
        \InPad~Q11\ => \InPad~Q11\,
        \InPad~Q12\ => \InPad~Q12\,
        \InPad~Q13\ => \InPad~Q13\,
        \InPad~Q14\ => \InPad~Q14\,
        \InPad~Q15\ => \InPad~Q15\,
        ODVin => ODVin,
        ODVout => ODVout,
        Oper => Oper,
        OutPadIS0 => OutPadIS0,
        OutPadIS1 => OutPadIS1,
        OutPadIS2 => OutPadIS2,
        OutPadIS3 => OutPadIS3,
        OutPadIS4 => OutPadIS4,
        OutPadIS5 => OutPadIS5,
        OutPadIS6 => OutPadIS6,
        OutPadIS7 => OutPadIS7,
        OutPadIS8 => OutPadIS8,
        OutPadIS9 => OutPadIS9,
        OutPadIS10  =>  OutPadIS10,
        OutPadIS11 => OutPadIS11,
        OutPadIS12 => OutPadIS12,
        OutPadIS13 => OutPadIS13,
        OutPadIS14 => OutPadIS14,
        OutPadIS15 => OutPadIS15,
        OutPadISOV => OutPadISOV,
        OutPadQS0 => OutPadQS0,
        OutPadQS1 => OutPadQS1,
        OutPadQS2 => OutPadQS2,
        OutPadQS3 => OutPadQS3,
        OutPadQS4 => OutPadQS4,
        OutPadQS5 => OutPadQS5,
        OutPadQS6 => OutPadQS6,
        OutPadQS7 => OutPadQS7,
        OutPadQS8 => OutPadQS8,
        OutPadQS9 => OutPadQS9,
        OutPadQS10 => OutPadQS10,
        OutPadQS11 => OutPadQS11,
        OutPadQS12 => OutPadQS12,
        OutPadQS13 => OutPadQS13,
        OutPadQS14 => OutPadQS14,
        OutPadQS15 => OutPadQS15,
OutPadQSOV => OutPadQSOV,
\OutPad~IS0\ => \OutPad~IS0\,
\OutPad~IS1\ => \OutPad~IS1\,
\OutPad~IS2\ => \OutPad~IS2\,
\OutPad~IS3\ => \OutPad~IS3\,
\OutPad~IS4\ => \OutPad~IS4\,
\OutPad~IS5\ => \OutPad~IS5\,
\OutPad~IS6\ => \OutPad~IS6\,
\OutPad~IS7\ => \OutPad~IS7\,
\OutPad~IS8\ => \OutPad~IS8\,
\OutPad~IS9\ => \OutPad~IS9\,
        \OutPad~IS10\ => \OutPad~IS10\,
        \OutPad~IS11\ => \OutPad~IS11\,
        \OutPad~IS12\ => \OutPad~IS12\,
        \OutPad~IS13\ => \OutPad~IS13\,
        \OutPad~IS14\ => \OutPad~IS14\,
        \OutPad~IS15\ => \OutPad~IS15\,
        \OutPad~QS0\ => \OutPad~QS0\,
        \OutPad~QS1\ => \OutPad~QS1\,
        \OutPad~QS2\ => \OutPad~QS2\,
        \OutPad~QS3\ => \OutPad~QS3\,
        \OutPad~QS4\ => \OutPad~QS4\,
        \OutPad~QS5\ => \OutPad~QS5\,
        \OutPad~QS6\ => \OutPad~QS6\,
        \OutPad~QS7\ => \OutPad~QS7\,
        \OutPad~QS8\ => \OutPad~QS8\,
        \OutPad~QS9\ => \OutPad~QS9\,
        \OutPad~QS10\ => \OutPad~QS10\,
        \OutPad~QS11\ => \OutPad~QS11\,
        \OutPad~QS12\ => \OutPad~QS12\,
        \OutPad~QS13\ => \OutPad~QS13\,
        \OutPad~QS14\ => \OutPad~QS14\,
        \OutPad~QS15\ => \OutPad~QS15\,
        PSV => PSV,
        RBinSelect0 => RBinSelect0,
        RBinSelect1 => RBinSelect1,
        RBinSelect2 => RBinSelect2,
        UNP => UNP,
        URB => URB
        );
        --Below VHDL code is an inserted
.\compile\Waveform Editor 1.vhs
                --User can modify it ....

        STIMULUS: process
        begin  -- of stimulus process
        --wait for <time to next event>; --
<current time>

                InPadI10 <= '0';
                InPadI11 <= '0';
                InPadI12 <= '0';
                InPadI13 <= '0';
                InPadI14 <= '0';
                InPadI15 <= '0';
                InPadI7 <= '0';
                InPadI8 <= '0';
                InPadI9 <= '0';
                InPadI1 <= '0';
                \InPad~I6\ <= '1';
                \InPad~I7\ <= '1';
                \InPad~I8\ <= '1';
                \InPad~I9\ <= '1';
                \InPad~I10\ <= '1';
                \InPad~I11\ <= '1';
                InPadQOV <= '0';
                \InPad~I13\ <= '1';
```

```
\InPad~I12\ <= '1';
\InPad~Q15\ <= '1';
InPadI6 <= '0';
InPadI5 <= '0';
InPadI4 <= '0';
InPadI3 <= '0';
InPadI2 <= '0';
\InPad~I14\ <= '1';
\InPad~Q14\ <= '1';
\InPad~Q13\ <= '1';
\InPad~Q12\ <= '1';
\InPad~Q11\ <= '1';
\InPad~I0\ <= '1';
\InPad~I1\ <= '1';
\InPad~I2\ <= '1';
\InPad~I3\ <= '1';
\InPad~I4\ <= '1';
\InPad~I5\ <= '1';
\InPad~Q10\ <= '1';
\InPad~Q9\ <= '1';
\InPad~Q8\ <= '1';
\InPad~Q7\ <= '1';
\InPad~Q6\ <= '1';
\InPad~Q5\ <= '1';
\InPad~Q4\ <= '1';
\InPad~Q3\ <= '1';
\InPad~Q2\ <= '1';
\InPad~Q1\ <= '1';
\InPad~Q0\ <= '1';
\InPad~I15\ <= '1';
InPadIOV <= '0';
InPadQ15 <= '0';
InPadQ14 <= '0';
InPadQ13 <= '0';
InPadQ12 <= '0';
InPadQ11 <= '0';
InPadQ10 <= '0';
InPadQ9 <= '0';
InPadQ8 <= '0';
InPadQ7 <= '0';
InPadQ6 <= '0';
InPadQ5 <= '0';
InPadQ4 <= '0';
InPadQ3 <= '0';
InPadQ2 <= '0';
InPadQ1 <= '0';
InPadQ0 <= '0';
Gain2 <= '1';
InPadI0 <= '0';
RBinSelect2 <= '1';
RBinSelect1 <= '1';
RBinSelect0 <= '1';
PSV <= '0';
Oper <= '1';
ODVin <= '0';
URB <= '1';
        UNP <= '0';
        Inc4 <= '1';
        Inc3 <= '1';
        Inc2 <= '1';
        Inc1 <= '0';
        Inc0 <= '0';
        Gain3 <= '1';
        Gain1 <= '0';
        Gain0 <= '1';
        DRFM4 <= 'U';
        DRFM3 <= 'U';
        DRFM2 <= 'U';
        DRFM1 <= 'U';
        DRFM0 <= 'U';
        ENABLE <= '1';
    wait for 2 ns; --0 fs
        RBinSelect0 <= '0';
        Inc2 <= '0';
        Inc1 <= '1';
    wait for 2 ns; --2 ns
        RBinSelect1 <= '0';
        RBinSelect0 <= '1';
        Inc3 <= '0';
        Inc2 <= '1';
    wait for 2 ns; --4 ns
        RBinSelect0 <= '0';
        Inc2 <= '0';
        Inc1 <= '0';
    wait for 2 ns; --6 ns
        RBinSelect2 <= '0';
        RBinSelect1 <= '1';
        RBinSelect0 <= '1';
        Gain1 <= '1';
        Gain0 <= '0';
    wait for 2 ns; --8 ns
        RBinSelect0 <= '0';
        Inc4 <= '0';
        Inc3 <= '1';
        Inc2 <= '1';
        Inc0 <= '1';
        Gain1 <= '0';
        Gain0 <= '1';
    wait for 2 ns; --10 ns
        RBinSelect1 <= '0';
        RBinSelect0 <= '1';
        Inc4 <= '1';
        Inc3 <= '0';
        Inc2 <= '0';
        Inc0 <= '0';
    wait for 2 ns; --12 ns
        RBinSelect0 <= '0';
        Inc4 <= '0';
        Inc3 <= '1';
        Inc2 <= '1';
        Inc1 <= '1';
        Inc0 <= '1';
```

```
        wait for 2 ns; --14 ns
                UNP <= '1';
                ENABLE <= '0';
        wait for 2 ns; --16 ns
                PSV <= '1';
                UNP <= '0';
                DRFM4 <= '0';
                DRFM3 <= '0';
                DRFM2 <= '0';
                DRFM1 <= '0';
                DRFM0 <= '0';
        wait for 2 ns; --18 ns
                DRFM4 <= '1';
                DRFM3 <= '1';
                DRFM2 <= '1';
                DRFM1 <= '1';
                DRFM0 <= '1';
        wait for 12 ns; --20 ns
                DRFM0 <= '0';
        wait for 6 ns; --32 ns
                PSV <= '0';
        wait for 62 ns; --38 ns
                END_SIM <= TRUE;
        --      end of stimulus events
                wait;
    end process; -- end of stimulus process

    CLOCK_CLK : process
    begin
                --this process was generated
based on formula: 0 0 ns, 1 1 ns -r 2 ns
                --wait for <time to next
event>; -- <current time>
                if END_SIM = FALSE then
                        CLK <= '0';
                        wait for 1 ns; --0 fs
                else
                        wait;
                end if;
                if END_SIM = FALSE then
                        CLK <= '1';
                        wait for 1 ns; --1 ns
                else
                        wait;
                end if;
        end process;

                -- Add your stimulus here ...

        end TB_ARCHITECTURE;

        configuration
TESTBENCH_FOR_dtm_8rbps                 of
dtm_8rbps_tb is
                for TB_ARCHITECTURE
                        for UUT : dtm_8rbps
                                use     entity
work.dtm_8rbps(structural);
                        end for;
                end for;
        end TESTBENCH_FOR_dtm_8rbps;
```

C. EXECUTING MACRO FOR THE 8 RANGE-BIN TEST BENCH

```
SetActiveLib -work
comp                    -include
"$DSN\src\dtm_8rbps.vhd"
comp                    -include
"$DSN\src\TestBench\dtm_8rbps_TB.vhd"
asim TESTBENCH_FOR_dtm_8rbps
wave
wave -noreg CLK
wave -noreg DRFM0
wave -noreg DRFM1
wave -noreg DRFM2
wave -noreg DRFM3
wave -noreg DRFM4
wave -noreg ENABLE
wave -noreg Gain0
wave -noreg Gain1
wave -noreg Gain2
wave -noreg Gain3
wave -noreg Inc0
wave -noreg Inc1
wave -noreg Inc2
wave -noreg Inc3
wave -noreg Inc4
wave -noreg InPadI0
wave -noreg InPadI1
wave -noreg InPadI2
wave -noreg InPadI3
wave -noreg InPadI4
wave -noreg InPadI5
wave -noreg InPadI6
wave -noreg InPadI7
wave -noreg InPadI8
wave -noreg InPadI9
wave -noreg InPadI10
wave -noreg InPadI11
wave -noreg InPadI12
wave -noreg InPadI13
wave -noreg InPadI14
wave -noreg InPadI15
wave -noreg InPadIOV
wave -noreg InPadQ0
wave -noreg InPadQ1
wave -noreg InPadQ2
wave -noreg InPadQ3
wave -noreg InPadQ4
wave -noreg InPadQ5
wave -noreg InPadQ6
wave -noreg InPadQ7
wave -noreg InPadQ8
wave -noreg InPadQ9
wave -noreg InPadQ10
wave -noreg InPadQ11
wave -noreg InPadQ12
wave -noreg InPadQ13
wave -noreg InPadQ14
wave -noreg InPadQ15
wave -noreg InPadQOV
wave -noreg {\InPad~I0\}
wave -noreg {\InPad~I1\}
wave -noreg {\InPad~I2\}
wave -noreg {\InPad~I3\}
wave -noreg {\InPad~I4\}
wave -noreg {\InPad~I5\}
wave -noreg {\InPad~I6\}
wave -noreg {\InPad~I7\}
wave -noreg {\InPad~I8\}
wave -noreg {\InPad~I9\}
wave -noreg {\InPad~I10\}
wave -noreg {\InPad~I11\}
wave -noreg {\InPad~I12\}
wave -noreg {\InPad~I13\}
wave -noreg {\InPad~I14\}
wave -noreg {\InPad~I15\}
wave -noreg {\InPad~Q0\}
wave -noreg {\InPad~Q1\}
wave -noreg {\InPad~Q2\}
wave -noreg {\InPad~Q3\}
wave -noreg {\InPad~Q4\}
wave -noreg {\InPad~Q5\}
wave -noreg {\InPad~Q6\}
wave -noreg {\InPad~Q7\}
wave -noreg {\InPad~Q8\}
wave -noreg {\InPad~Q9\}
wave -noreg {\InPad~Q10\}
wave -noreg {\InPad~Q11\}
wave -noreg {\InPad~Q12\}
wave -noreg {\InPad~Q13\}
wave -noreg {\InPad~Q14\}
wave -noreg {\InPad~Q15\}
wave -noreg ODVin
wave -noreg ODVout
wave -noreg Oper
wave -noreg OutPadIS0
wave -noreg OutPadIS1
wave -noreg OutPadIS2
wave -noreg OutPadIS3
wave -noreg OutPadIS4
wave -noreg OutPadIS5
wave -noreg OutPadIS6
wave -noreg OutPadIS7
wave -noreg OutPadIS8
wave -noreg OutPadIS9
wave -noreg OutPadIS10
wave -noreg OutPadIS11
wave -noreg OutPadIS12
wave -noreg OutPadIS13
wave -noreg OutPadIS14
wave -noreg OutPadIS15
```

```
wave -noreg OutPadISOV
wave -noreg OutPadQS0
wave -noreg OutPadQS1
wave -noreg OutPadQS2
wave -noreg OutPadQS3
wave -noreg OutPadQS4
wave -noreg OutPadQS5
wave -noreg OutPadQS6
wave -noreg OutPadQS7
wave -noreg OutPadQS8
wave -noreg OutPadQS9
wave -noreg OutPadQS10
wave -noreg OutPadQS11
wave -noreg OutPadQS12
wave -noreg OutPadQS13
wave -noreg OutPadQS14
wave -noreg OutPadQS15
wave -noreg OutPadQSOV
wave -noreg {\OutPad~IS0\}
wave -noreg {\OutPad~IS1\}
wave -noreg {\OutPad~IS2\}
wave -noreg {\OutPad~IS3\}
wave -noreg {\OutPad~IS4\}
wave -noreg {\OutPad~IS5\}
wave -noreg {\OutPad~IS6\}
wave -noreg {\OutPad~IS7\}
wave -noreg {\OutPad~IS8\}
wave -noreg {\OutPad~IS9\}
wave -noreg {\OutPad~IS10\}
wave -noreg {\OutPad~IS11\}
wave -noreg {\OutPad~IS12\}
wave -noreg {\OutPad~IS13\}
wave -noreg {\OutPad~IS14\}
wave -noreg {\OutPad~IS15\}
wave -noreg {\OutPad~QS0\}
wave -noreg {\OutPad~QS1\}
wave -noreg {\OutPad~QS2\}
wave -noreg {\OutPad~QS3\}
wave -noreg {\OutPad~QS4\}
wave -noreg {\OutPad~QS5\}
wave -noreg {\OutPad~QS6\}
wave -noreg {\OutPad~QS7\}
wave -noreg {\OutPad~QS8\}
wave -noreg {\OutPad~QS9\}
wave -noreg {\OutPad~QS10\}
wave -noreg {\OutPad~QS11\}
wave -noreg {\OutPad~QS12\}
wave -noreg {\OutPad~QS13\}
wave -noreg {\OutPad~QS14\}
wave -noreg {\OutPad~QS15\}
wave -noreg PSV
wave -noreg RBinSelect0
wave -noreg RBinSelect1
wave -noreg RBinSelect2
wave -noreg UNP
wave -noreg URB
run 100.00 ns
# The following lines can be used for timing simulation
#                                    acom <backannotated_vhdl_file_name>
#                comp                -include "$DSN\src\TestBench\dtm_8rbps_TB_tim_cfg.vhd"
# asim TIMING_FOR_dtm_8rbps
```

APPENDIX F. VHDL CODE FOR THE 32 RANGE-BIN MODULATOR

A. TOP LEVEL VHDL CODE

```
-------------------------------------------------
-- Title      :
-- Design     : HB_32_RB_2
-- Author     : Hakan Bergon
-- Company    : NPS
-------------------------------------------------
--
--      File                    :
c:\My_Designs\HB_32_RB_2\compile\HB_32R
BPs.vhd
-- Generated    : Wed Aug 21 12:04:11
2002
--      From                    :
c:\My_Designs\HB_32_RB_2\src\HB_32RBPs.b
de
-- By         : Bde2Vhdl ver. 2.01
--
-- Description :
--
-- Design unit header --
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

entity HB_32RBPs is
  port(
      CLK : in std_logic;
      DRFM0 : in std_logic;
      DRFM1 : in std_logic;
      DRFM2 : in std_logic;
      DRFM3 : in std_logic;
      DRFM4 : in std_logic;
      ENABLE_1 : in std_logic;
      ENABLE_2 : in std_logic;
      ENABLE_3 : in std_logic;
      ENABLE_4 : in std_logic;
      Gain0 : in std_logic;
      Gain1 : in std_logic;
      Gain2 : in std_logic;
      Gain3 : in std_logic;
      InPadI0 : in std_logic;
      InPadI1 : in std_logic;
      InPadI10 : in std_logic;
      InPadI11 : in std_logic;
      InPadI12 : in std_logic;
      InPadI13 : in std_logic;
      InPadI14 : in std_logic;
      InPadI15 : in std_logic;
      InPadI2 : in std_logic;
      InPadI3 : in std_logic;
      InPadI4 : in std_logic;
      InPadI5 : in std_logic;
      InPadI6 : in std_logic;
      InPadI7 : in std_logic;
      InPadI8 : in std_logic;
      InPadI9 : in std_logic;
      InPadIOV : in std_logic;
      InPadQ0 : in std_logic;
      InPadQ1 : in std_logic;
      InPadQ10 : in std_logic;
      InPadQ11 : in std_logic;
      InPadQ12 : in std_logic;
      InPadQ13 : in std_logic;
      InPadQ14 : in std_logic;
      InPadQ15 : in std_logic;
      InPadQ2 : in std_logic;
      InPadQ3 : in std_logic;
      InPadQ4 : in std_logic;
      InPadQ5 : in std_logic;
      InPadQ6 : in std_logic;
      InPadQ7 : in std_logic;
      InPadQ8 : in std_logic;
      InPadQ9 : in std_logic;
      InPadQOV : in std_logic;
      Inc0 : in std_logic;
      Inc1 : in std_logic;
      Inc2 : in std_logic;
      Inc3 : in std_logic;
      Inc4 : in std_logic;
      ODVin : in std_logic;
      Oper : in std_logic;
      PSV : in std_logic;
      RB_81_inSelect0 : in std_logic;
      RB_81_inSelect1 : in std_logic;
      RB_81_inSelect2 : in std_logic;
      RB_82_inSelect0 : in std_logic;
      RB_82_inSelect1 : in std_logic;
      RB_82_inSelect2 : in std_logic;
      RB_83_inSelect0 : in std_logic;
      RB_83_inSelect1 : in std_logic;
      RB_83_inSelect2 : in std_logic;
      RB_84_inSelect0 : in std_logic;
      RB_84_inSelect1 : in std_logic;
      RB_84_inSelect2 : in std_logic;
      UNP : in std_logic;
      URB : in std_logic;
      \InPad~I0\ : in std_logic;
      \InPad~I10\ : in std_logic;
      \InPad~I11\ : in std_logic;
```

```
\InPad~I12\ : in std_logic;
\InPad~I13\ : in std_logic;
\InPad~I14\ : in std_logic;
\InPad~I15\ : in std_logic;
\InPad~I1\ : in std_logic;
\InPad~I2\ : in std_logic;
\InPad~I3\ : in std_logic;
\InPad~I4\ : in std_logic;
\InPad~I5\ : in std_logic;
\InPad~I6\ : in std_logic;
\InPad~I7\ : in std_logic;
\InPad~I8\ : in std_logic;
\InPad~I9\ : in std_logic;
\InPad~Q0\ : in std_logic;
\InPad~Q10\ : in std_logic;
\InPad~Q11\ : in std_logic;
\InPad~Q12\ : in std_logic;
\InPad~Q13\ : in std_logic;
\InPad~Q14\ : in std_logic;
\InPad~Q15\ : in std_logic;
\InPad~Q1\ : in std_logic;
\InPad~Q2\ : in std_logic;
\InPad~Q3\ : in std_logic;
\InPad~Q4\ : in std_logic;
\InPad~Q5\ : in std_logic;
\InPad~Q6\ : in std_logic;
\InPad~Q7\ : in std_logic;
\InPad~Q8\ : in std_logic;
\InPad~Q9\ : in std_logic;
ODVout : out std_logic;
OutPadIS0 : out std_logic;
OutPadIS1 : out std_logic;
OutPadIS10 : out std_logic;
OutPadIS11 : out std_logic;
OutPadIS12 : out std_logic;
OutPadIS13 : out std_logic;
OutPadIS14 : out std_logic;
OutPadIS15 : out std_logic;
OutPadIS2 : out std_logic;
OutPadIS3 : out std_logic;
OutPadIS4 : out std_logic;
OutPadIS5 : out std_logic;
OutPadIS6 : out std_logic;
OutPadIS7 : out std_logic;
OutPadIS8 : out std_logic;
OutPadIS9 : out std_logic;
OutPadISOV : out std_logic;
OutPadQS0 : out std_logic;
OutPadQS1 : out std_logic;
OutPadQS10 : out std_logic;
OutPadQS11 : out std_logic;
OutPadQS12 : out std_logic;
OutPadQS13 : out std_logic;
OutPadQS14 : out std_logic;
OutPadQS15 : out std_logic;
OutPadQS2 : out std_logic;
OutPadQS3 : out std_logic;
OutPadQS4 : out std_logic;
OutPadQS5 : out std_logic;
OutPadQS6 : out std_logic;
OutPadQS7 : out std_logic;
OutPadQS8 : out std_logic;
OutPadQS9 : out std_logic;
OutPadQSOV : out std_logic;
\OutPad~IS0\ : out std_logic;
\OutPad~IS10\ : out std_logic;
\OutPad~IS11\ : out std_logic;
\OutPad~IS12\ : out std_logic;
\OutPad~IS13\ : out std_logic;
\OutPad~IS14\ : out std_logic;
\OutPad~IS15\ : out std_logic;
\OutPad~IS1\ : out std_logic;
\OutPad~IS2\ : out std_logic;
\OutPad~IS3\ : out std_logic;
\OutPad~IS4\ : out std_logic;
\OutPad~IS5\ : out std_logic;
\OutPad~IS6\ : out std_logic;
\OutPad~IS7\ : out std_logic;
\OutPad~IS8\ : out std_logic;
\OutPad~IS9\ : out std_logic;
\OutPad~QS0\ : out std_logic;
\OutPad~QS10\ : out std_logic;
\OutPad~QS11\ : out std_logic;
\OutPad~QS12\ : out std_logic;
\OutPad~QS13\ : out std_logic;
\OutPad~QS14\ : out std_logic;
\OutPad~QS15\ : out std_logic;
\OutPad~QS1\ : out std_logic;
\OutPad~QS2\ : out std_logic;
\OutPad~QS3\ : out std_logic;
\OutPad~QS4\ : out std_logic;
\OutPad~QS5\ : out std_logic;
\OutPad~QS6\ : out std_logic;
\OutPad~QS7\ : out std_logic;
\OutPad~QS8\ : out std_logic;
\OutPad~QS9\ : out std_logic
  );
end HB_32RBPs;

architecture structural of HB_32RBPs is

---- Component declarations -----

component DTM_8RBPs
  port (
      CLK : in STD_LOGIC;
      DRFM0 : in STD_LOGIC;
      DRFM1 : in STD_LOGIC;
      DRFM2 : in STD_LOGIC;
      DRFM3 : in STD_LOGIC;
      DRFM4 : in STD_LOGIC;
      ENABLE : in STD_LOGIC;
```

```
Gain0 : in STD_LOGIC;
Gain1 : in STD_LOGIC;
Gain2 : in STD_LOGIC;
Gain3 : in STD_LOGIC;
InPadI0 : in STD_LOGIC;
InPadI1 : in STD_LOGIC;
InPadI10 : in STD_LOGIC;
InPadI11 : in STD_LOGIC;
InPadI12 : in STD_LOGIC;
InPadI13 : in STD_LOGIC;
InPadI14 : in STD_LOGIC;
InPadI15 : in STD_LOGIC;
InPadI2 : in STD_LOGIC;
InPadI3 : in STD_LOGIC;
InPadI4 : in STD_LOGIC;
InPadI5 : in STD_LOGIC;
InPadI6 : in STD_LOGIC;
InPadI7 : in STD_LOGIC;
InPadI8 : in STD_LOGIC;
InPadI9 : in STD_LOGIC;
InPadIOV : in STD_LOGIC;
InPadQ0 : in STD_LOGIC;
InPadQ1 : in STD_LOGIC;
InPadQ10 : in STD_LOGIC;
InPadQ11 : in STD_LOGIC;
InPadQ12 : in STD_LOGIC;
InPadQ13 : in STD_LOGIC;
InPadQ14 : in STD_LOGIC;
InPadQ15 : in STD_LOGIC;
InPadQ2 : in STD_LOGIC;
InPadQ3 : in STD_LOGIC;
InPadQ4 : in STD_LOGIC;
InPadQ5 : in STD_LOGIC;
InPadQ6 : in STD_LOGIC;
InPadQ7 : in STD_LOGIC;
InPadQ8 : in STD_LOGIC;
InPadQ9 : in STD_LOGIC;
InPadQOV : in STD_LOGIC;
Inc0 : in STD_LOGIC;
Inc1 : in STD_LOGIC;
Inc2 : in STD_LOGIC;
Inc3 : in STD_LOGIC;
Inc4 : in STD_LOGIC;
ODVin : in STD_LOGIC;
Oper : in STD_LOGIC;
PSV : in STD_LOGIC;
RBinSelect0 : in STD_LOGIC;
RBinSelect1 : in STD_LOGIC;
RBinSelect2 : in STD_LOGIC;
UNP : in STD_LOGIC;
URB : in STD_LOGIC;
\InPad~I0\ : in STD_LOGIC;
\InPad~I10\ : in STD_LOGIC;
\InPad~I11\ : in STD_LOGIC;
\InPad~I12\ : in STD_LOGIC;
\InPad~I13\ : in STD_LOGIC;
\InPad~I14\ : in STD_LOGIC;
\InPad~I15\ : in STD_LOGIC;
\InPad~I1\ : in STD_LOGIC;
\InPad~I2\ : in STD_LOGIC;
\InPad~I3\ : in STD_LOGIC;
\InPad~I4\ : in STD_LOGIC;
\InPad~I5\ : in STD_LOGIC;
\InPad~I6\ : in STD_LOGIC;
\InPad~I7\ : in STD_LOGIC;
\InPad~I8\ : in STD_LOGIC;
\InPad~I9\ : in STD_LOGIC;
\InPad~Q0\ : in STD_LOGIC;
\InPad~Q10\ : in STD_LOGIC;
\InPad~Q11\ : in STD_LOGIC;
\InPad~Q12\ : in STD_LOGIC;
\InPad~Q13\ : in STD_LOGIC;
\InPad~Q14\ : in STD_LOGIC;
\InPad~Q15\ : in STD_LOGIC;
\InPad~Q1\ : in STD_LOGIC;
\InPad~Q2\ : in STD_LOGIC;
\InPad~Q3\ : in STD_LOGIC;
\InPad~Q4\ : in STD_LOGIC;
\InPad~Q5\ : in STD_LOGIC;
\InPad~Q6\ : in STD_LOGIC;
\InPad~Q7\ : in STD_LOGIC;
\InPad~Q8\ : in STD_LOGIC;
\InPad~Q9\ : in STD_LOGIC;
ODVout : out STD_LOGIC;
OutPadIS0 : out STD_LOGIC;
OutPadIS1 : out STD_LOGIC;
OutPadIS10 : out STD_LOGIC;
OutPadIS11 : out STD_LOGIC;
OutPadIS12 : out STD_LOGIC;
OutPadIS13 : out STD_LOGIC;
OutPadIS14 : out STD_LOGIC;
OutPadIS15 : out STD_LOGIC;
OutPadIS2 : out STD_LOGIC;
OutPadIS3 : out STD_LOGIC;
OutPadIS4 : out STD_LOGIC;
OutPadIS5 : out STD_LOGIC;
OutPadIS6 : out STD_LOGIC;
OutPadIS7 : out STD_LOGIC;
OutPadIS8 : out STD_LOGIC;
OutPadIS9 : out STD_LOGIC;
OutPadISOV : out STD_LOGIC;
OutPadQS0 : out STD_LOGIC;
OutPadQS1 : out STD_LOGIC;
OutPadQS10 : out STD_LOGIC;
OutPadQS11 : out STD_LOGIC;
OutPadQS12 : out STD_LOGIC;
OutPadQS13 : out STD_LOGIC;
OutPadQS14 : out STD_LOGIC;
OutPadQS15 : out STD_LOGIC;
OutPadQS2 : out STD_LOGIC;
OutPadQS3 : out STD_LOGIC;
OutPadQS4 : out STD_LOGIC;
```

```
        OutPadQS5 : out STD_LOGIC;
        OutPadQS6 : out STD_LOGIC;
        OutPadQS7 : out STD_LOGIC;
        OutPadQS8 : out STD_LOGIC;
        OutPadQS9 : out STD_LOGIC;
        OutPadQSOV : out STD_LOGIC;
        \OutPad~IS0\ : out STD_LOGIC;
        \OutPad~IS10\ : out STD_LOGIC;
        \OutPad~IS11\ : out STD_LOGIC;
        \OutPad~IS12\ : out STD_LOGIC;
        \OutPad~IS13\ : out STD_LOGIC;
        \OutPad~IS14\ : out STD_LOGIC;
        \OutPad~IS15\ : out STD_LOGIC;
        \OutPad~IS1\ : out STD_LOGIC;
        \OutPad~IS2\ : out STD_LOGIC;
        \OutPad~IS3\ : out STD_LOGIC;
        \OutPad~IS4\ : out STD_LOGIC;
        \OutPad~IS5\ : out STD_LOGIC;
        \OutPad~IS6\ : out STD_LOGIC;
        \OutPad~IS7\ : out STD_LOGIC;
        \OutPad~IS8\ : out STD_LOGIC;
        \OutPad~IS9\ : out STD_LOGIC;
        \OutPad~QS0\ : out STD_LOGIC;
        \OutPad~QS10\ : out STD_LOGIC;
        \OutPad~QS11\ : out STD_LOGIC;
        \OutPad~QS12\ : out STD_LOGIC;
        \OutPad~QS13\ : out STD_LOGIC;
        \OutPad~QS14\ : out STD_LOGIC;
        \OutPad~QS15\ : out STD_LOGIC;
        \OutPad~QS1\ : out STD_LOGIC;
        \OutPad~QS2\ : out STD_LOGIC;
        \OutPad~QS3\ : out STD_LOGIC;
        \OutPad~QS4\ : out STD_LOGIC;
        \OutPad~QS5\ : out STD_LOGIC;
        \OutPad~QS6\ : out STD_LOGIC;
        \OutPad~QS7\ : out STD_LOGIC;
        \OutPad~QS8\ : out STD_LOGIC;
        \OutPad~QS9\ : out STD_LOGIC
    );
    end component;

    ---- Signal declarations used on the
diagram ----

    signal P8101 : STD_LOGIC;
    signal P8102 : STD_LOGIC;
    signal P8103 : STD_LOGIC;
    signal P8104 : STD_LOGIC;
    signal P8105 : STD_LOGIC;
    signal P8106 : STD_LOGIC;
    signal P8107 : STD_LOGIC;
    signal P8108 : STD_LOGIC;
    signal P8109 : STD_LOGIC;
    signal P8110 : STD_LOGIC;
    signal P8111 : STD_LOGIC;
    signal P8112 : STD_LOGIC;
    signal P8113 : STD_LOGIC;
    signal P8114 : STD_LOGIC;
    signal P8115 : STD_LOGIC;
    signal P8116 : STD_LOGIC;
    signal P8117 : STD_LOGIC;
    signal P8118 : STD_LOGIC;
    signal P8119 : STD_LOGIC;
    signal P8120 : STD_LOGIC;
    signal P8121 : STD_LOGIC;
    signal P8122 : STD_LOGIC;
    signal P8123 : STD_LOGIC;
    signal P8124 : STD_LOGIC;
    signal P8125 : STD_LOGIC;
    signal P8126 : STD_LOGIC;
    signal P8127 : STD_LOGIC;
    signal P8128 : STD_LOGIC;
    signal P8129 : STD_LOGIC;
    signal P8130 : STD_LOGIC;
    signal P8131 : STD_LOGIC;
    signal P8132 : STD_LOGIC;
    signal P8133 : STD_LOGIC;
    signal P8134 : STD_LOGIC;
    signal P8135 : STD_LOGIC;
    signal P8136 : STD_LOGIC;
    signal P8137 : STD_LOGIC;
    signal P8138 : STD_LOGIC;
    signal P8139 : STD_LOGIC;
    signal P8140 : STD_LOGIC;
    signal P8141 : STD_LOGIC;
    signal P8142 : STD_LOGIC;
    signal P8143 : STD_LOGIC;
    signal P8144 : STD_LOGIC;
    signal P8145 : STD_LOGIC;
    signal P8146 : STD_LOGIC;
    signal P8147 : STD_LOGIC;
    signal P8148 : STD_LOGIC;
    signal P8149 : STD_LOGIC;
    signal P8150 : STD_LOGIC;
    signal P8151 : STD_LOGIC;
    signal P8152 : STD_LOGIC;
    signal P8153 : STD_LOGIC;
    signal P8154 : STD_LOGIC;
    signal P8155 : STD_LOGIC;
    signal P8156 : STD_LOGIC;
    signal P8157 : STD_LOGIC;
    signal P8158 : STD_LOGIC;
    signal P8159 : STD_LOGIC;
    signal P8160 : STD_LOGIC;
    signal P8161 : STD_LOGIC;
    signal P8162 : STD_LOGIC;
    signal P8163 : STD_LOGIC;
    signal P8164 : STD_LOGIC;
    signal P8165 : STD_LOGIC;
    signal P8166 : STD_LOGIC;
    signal P8201 : STD_LOGIC;
    signal P8202 : STD_LOGIC;
```

signal P8203 : STD_LOGIC;
signal P8204 : STD_LOGIC;
signal P8205 : STD_LOGIC;
signal P8206 : STD_LOGIC;
signal P8207 : STD_LOGIC;
signal P8208 : STD_LOGIC;
signal P8209 : STD_LOGIC;
signal P8210 : STD_LOGIC;
signal P8211 : STD_LOGIC;
signal P8212 : STD_LOGIC;
signal P8213 : STD_LOGIC;
signal P8214 : STD_LOGIC;
signal P8215 : STD_LOGIC;
signal P8216 : STD_LOGIC;
signal P8217 : STD_LOGIC;
signal P8218 : STD_LOGIC;
signal P8219 : STD_LOGIC;
signal P8220 : STD_LOGIC;
signal P8221 : STD_LOGIC;
signal P8222 : STD_LOGIC;
signal P8223 : STD_LOGIC;
signal P8224 : STD_LOGIC;
signal P8225 : STD_LOGIC;
signal P8226 : STD_LOGIC;
signal P8227 : STD_LOGIC;
signal P8228 : STD_LOGIC;
signal P8229 : STD_LOGIC;
signal P8230 : STD_LOGIC;
signal P8231 : STD_LOGIC;
signal P8232 : STD_LOGIC;
signal P8233 : STD_LOGIC;
signal P8234 : STD_LOGIC;
signal P8235 : STD_LOGIC;
signal P8236 : STD_LOGIC;
signal P8237 : STD_LOGIC;
signal P8238 : STD_LOGIC;
signal P8239 : STD_LOGIC;
signal P8240 : STD_LOGIC;
signal P8241 : STD_LOGIC;
signal P8242 : STD_LOGIC;
signal P8243 : STD_LOGIC;
signal P8244 : STD_LOGIC;
signal P8245 : STD_LOGIC;
signal P8246 : STD_LOGIC;
signal P8247 : STD_LOGIC;
signal P8248 : STD_LOGIC;
signal P8249 : STD_LOGIC;
signal P8250 : STD_LOGIC;
signal P8251 : STD_LOGIC;
signal P8252 : STD_LOGIC;
signal P8253 : STD_LOGIC;
signal P8254 : STD_LOGIC;
signal P8255 : STD_LOGIC;
signal P8256 : STD_LOGIC;
signal P8257 : STD_LOGIC;
signal P8258 : STD_LOGIC;
signal P8259 : STD_LOGIC;
signal P8260 : STD_LOGIC;
signal P8261 : STD_LOGIC;
signal P8262 : STD_LOGIC;
signal P8263 : STD_LOGIC;
signal P8264 : STD_LOGIC;
signal P8265 : STD_LOGIC;
signal P8266 : STD_LOGIC;
signal P8301 : STD_LOGIC;
signal P8302 : STD_LOGIC;
signal P8303 : STD_LOGIC;
signal P8304 : STD_LOGIC;
signal P8305 : STD_LOGIC;
signal P8306 : STD_LOGIC;
signal P8307 : STD_LOGIC;
signal P8308 : STD_LOGIC;
signal P8309 : STD_LOGIC;
signal P8310 : STD_LOGIC;
signal P8311 : STD_LOGIC;
signal P8312 : STD_LOGIC;
signal P8313 : STD_LOGIC;
signal P8314 : STD_LOGIC;
signal P8315 : STD_LOGIC;
signal P8316 : STD_LOGIC;
signal P8317 : STD_LOGIC;
signal P8318 : STD_LOGIC;
signal P8319 : STD_LOGIC;
signal P8320 : STD_LOGIC;
signal P8321 : STD_LOGIC;
signal P8322 : STD_LOGIC;
signal P8323 : STD_LOGIC;
signal P8324 : STD_LOGIC;
signal P8325 : STD_LOGIC;
signal P8326 : STD_LOGIC;
signal P8327 : STD_LOGIC;
signal P8328 : STD_LOGIC;
signal P8329 : STD_LOGIC;
signal P8330 : STD_LOGIC;
signal P8331 : STD_LOGIC;
signal P8332 : STD_LOGIC;
signal P8333 : STD_LOGIC;
signal P8334 : STD_LOGIC;
signal P8335 : STD_LOGIC;
signal P8336 : STD_LOGIC;
signal P8337 : STD_LOGIC;
signal P8338 : STD_LOGIC;
signal P8339 : STD_LOGIC;
signal P8340 : STD_LOGIC;
signal P8341 : STD_LOGIC;
signal P8342 : STD_LOGIC;
signal P8343 : STD_LOGIC;
signal P8344 : STD_LOGIC;
signal P8345 : STD_LOGIC;
signal P8346 : STD_LOGIC;
signal P8347 : STD_LOGIC;
signal P8348 : STD_LOGIC;

```
        signal P8349 : STD_LOGIC;
        signal P8350 : STD_LOGIC;
        signal P8351 : STD_LOGIC;
        signal P8352 : STD_LOGIC;
        signal P8353 : STD_LOGIC;
        signal P8354 : STD_LOGIC;
        signal P8355 : STD_LOGIC;
        signal P8356 : STD_LOGIC;
        signal P8357 : STD_LOGIC;
        signal P8358 : STD_LOGIC;
        signal P8359 : STD_LOGIC;
        signal P8360 : STD_LOGIC;
        signal P8361 : STD_LOGIC;
        signal P8362 : STD_LOGIC;
        signal P8363 : STD_LOGIC;
        signal P8364 : STD_LOGIC;
        signal P8365 : STD_LOGIC;
        signal P8366 : STD_LOGIC;

        begin

        ---- Component instantiations  ----

        \HB_RangeBinModulator_32_1\         :
DTM_8RBPs
            port map(
                CLK => CLK,
                DRFM0 => DRFM0,
                DRFM1 => DRFM1,
                DRFM2 => DRFM2,
                DRFM3 => DRFM3,
                DRFM4 => DRFM4,
                ENABLE => ENABLE_1,
                Gain0 => Gain0,
                Gain1 => Gain1,
                Gain2 => Gain2,
                Gain3 => Gain3,
                InPadI0 => InPadI0,
                InPadI1 => InPadI1,
                InPadI10 => InPadI10,
                InPadI11 => InPadI11,
                InPadI12 => InPadI12,
                InPadI13 => InPadI13,
                InPadI14 => InPadI14,
                InPadI15 => InPadI15,
                InPadI2 => InPadI2,
                InPadI3 => InPadI3,
                InPadI4 => InPadI4,
                InPadI5 => InPadI5,
                InPadI6 => InPadI6,
                InPadI7 => InPadI7,
                InPadI8 => InPadI8,
                InPadI9 => InPadI9,
                InPadIOV => InPadIOV,
                InPadQ0 => InPadQ0,
                InPadQ1 => InPadQ1,
                InPadQ10 => InPadQ10,
                InPadQ11 => InPadQ11,
                InPadQ12 => InPadQ12,
                InPadQ13 => InPadQ13,
                InPadQ14 => InPadQ14,
                InPadQ15 => InPadQ15,
                InPadQ2 => InPadQ2,
                InPadQ3 => InPadQ3,
                InPadQ4 => InPadQ4,
                InPadQ5 => InPadQ5,
                InPadQ6 => InPadQ6,
                InPadQ7 => InPadQ7,
                InPadQ8 => InPadQ8,
                InPadQ9 => InPadQ9,
                InPadQOV => InPadQOV,
                Inc0 => Inc0,
                Inc1 => Inc1,
                Inc2 => Inc2,
                Inc3 => Inc3,
                Inc4 => Inc4,
                ODVin => ODVin,
                ODVout => ODVout,
                Oper => Oper,
                OutPadIS0 => P8101,
                OutPadIS1 => P8102,
                OutPadIS10 => P8111,
                OutPadIS11 => P8112,
                OutPadIS12 => P8113,
                OutPadIS13 => P8114,
                OutPadIS14 => P8115,
                OutPadIS15 => P8116,
                OutPadIS2 => P8103,
                OutPadIS3 => P8104,
                OutPadIS4 => P8105,
                OutPadIS5 => P8106,
                OutPadIS6 => P8107,
                OutPadIS7 => P8108,
                OutPadIS8 => P8109,
                OutPadIS9 => P8110,
                OutPadISOV => P8117,
                OutPadQS0 => P8118,
                OutPadQS1 => P8119,
                OutPadQS10 => P8127,
                OutPadQS11 => P8128,
                OutPadQS12 => P8129,
                OutPadQS13 => P8130,
                OutPadQS14 => P8131,
                OutPadQS15 => P8132,
                OutPadQS2 => P8120,
                OutPadQS3 => P8121,
                OutPadQS4 => P8122,
                OutPadQS5 => P8166,
                OutPadQS6 => P8123,
                OutPadQS7 => P8124,
                OutPadQS8 => P8125,
                OutPadQS9 => P8126,
```

```
OutPadQSOV => P8133,
PSV => PSV,
RBinSelect0 => RB_81_inSelect0,
RBinSelect1 => RB_81_inSelect1,
RBinSelect2 => RB_81_inSelect2,
UNP => UNP,
URB => URB,
\InPad~I0\ => \InPad~I0\,
\InPad~I10\ => \InPad~I10\,
\InPad~I11\ => \InPad~I11\,
\InPad~I12\ => \InPad~I12\,
\InPad~I13\ => \InPad~I13\,
\InPad~I14\ => \InPad~I14\,
\InPad~I15\ => \InPad~I15\,
\InPad~I1\ => \InPad~I1\,
\InPad~I2\ => \InPad~I2\,
\InPad~I3\ => \InPad~I3\,
\InPad~I4\ => \InPad~I4\,
\InPad~I5\ => \InPad~I5\,
\InPad~I6\ => \InPad~I6\,
\InPad~I7\ => \InPad~I7\,
\InPad~I8\ => \InPad~I8\,
\InPad~I9\ => \InPad~I9\,
\InPad~Q0\ => \InPad~Q0\,
\InPad~Q10\ => \InPad~Q10\,
\InPad~Q11\ => \InPad~Q11\,
\InPad~Q12\ => \InPad~Q12\,
\InPad~Q13\ => \InPad~Q13\,
\InPad~Q14\ => \InPad~Q14\,
\InPad~Q15\ => \InPad~Q15\,
\InPad~Q1\ => \InPad~Q1\,
\InPad~Q2\ => \InPad~Q2\,
\InPad~Q3\ => \InPad~Q3\,
\InPad~Q4\ => \InPad~Q4\,
\InPad~Q5\ => \InPad~Q5\,
\InPad~Q6\ => \InPad~Q6\,
\InPad~Q7\ => \InPad~Q7\,
\InPad~Q8\ => \InPad~Q8\,
\InPad~Q9\ => \InPad~Q9\,
\OutPad~IS0\ => P8134,
\OutPad~IS10\ => P8144,
\OutPad~IS11\ => P8145,
\OutPad~IS12\ => P8146,
\OutPad~IS13\ => P8147,
\OutPad~IS14\ => P8148,
\OutPad~IS15\ => P8149,
\OutPad~IS1\ => P8135,
\OutPad~IS2\ => P8136,
\OutPad~IS3\ => P8137,
\OutPad~IS4\ => P8138,
\OutPad~IS5\ => P8139,
\OutPad~IS6\ => P8140,
\OutPad~IS7\ => P8141,
\OutPad~IS8\ => P8142,
\OutPad~IS9\ => P8143,
\OutPad~QS0\ => P8150,
\OutPad~QS10\ => P8160,
\OutPad~QS11\ => P8161,
\OutPad~QS12\ => P8162,
\OutPad~QS13\ => P8163,
\OutPad~QS14\ => P8164,
\OutPad~QS15\ => P8165,
\OutPad~QS1\ => P8151,
\OutPad~QS2\ => P8152,
\OutPad~QS3\ => P8153,
\OutPad~QS4\ => P8154,
\OutPad~QS5\ => P8155,
\OutPad~QS6\ => P8156,
\OutPad~QS7\ => P8157,
\OutPad~QS8\ => P8158,
\OutPad~QS9\ => P8159
);

\HB_RangeBinModulator_32_2\ : DTM_8RBPs
port map(
CLK => CLK,
DRFM0 => DRFM0,
DRFM1 => DRFM1,
DRFM2 => DRFM2,
DRFM3 => DRFM3,
DRFM4 => DRFM4,
ENABLE => ENABLE_2,
Gain0 => Gain0,
Gain1 => Gain1,
Gain2 => Gain2,
Gain3 => Gain3,
InPadI0 => P8101,
InPadI1 => P8102,
InPadI10 => P8111,
InPadI11 => P8112,
InPadI12 => P8113,
InPadI13 => P8114,
InPadI14 => P8115,
InPadI15 => P8116,
InPadI2 => P8103,
InPadI3 => P8104,
InPadI4 => P8105,
InPadI5 => P8106,
InPadI6 => P8107,
InPadI7 => P8108,
InPadI8 => P8109,
InPadI9 => P8110,
InPadIOV => P8117,
InPadQ0 => P8118,
InPadQ1 => P8119,
InPadQ10 => P8127,
InPadQ11 => P8128,
InPadQ12 => P8129,
InPadQ13 => P8130,
InPadQ14 => P8131,
InPadQ15 => P8132,
```

InPadQ2 => P8120,
InPadQ3 => P8121,
InPadQ4 => P8122,
InPadQ5 => P8166,
InPadQ6 => P8123,
InPadQ7 => P8124,
InPadQ8 => P8125,
InPadQ9 => P8126,
InPadQOV => P8133,
Inc0 => Inc0,
Inc1 => Inc1,
Inc2 => Inc2,
Inc3 => Inc3,
Inc4 => Inc4,
ODVin => ODVin,
ODVout => ODVout,
Oper => Oper,
OutPadIS0 => P8201,
OutPadIS1 => P8202,
OutPadIS10 => P8211,
OutPadIS11 => P8212,
OutPadIS12 => P8213,
OutPadIS13 => P8214,
OutPadIS14 => P8215,
OutPadIS15 => P8216,
OutPadIS2 => P8203,
OutPadIS3 => P8204,
OutPadIS4 => P8205,
OutPadIS5 => P8206,
OutPadIS6 => P8207,
OutPadIS7 => P8208,
OutPadIS8 => P8209,
OutPadIS9 => P8210,
OutPadISOV => P8217,
OutPadQS0 => P8218,
OutPadQS1 => P8219,
OutPadQS10 => P8227,
OutPadQS11 => P8228,
OutPadQS12 => P8229,
OutPadQS13 => P8230,
OutPadQS14 => P8231,
OutPadQS15 => P8232,
OutPadQS2 => P8220,
OutPadQS3 => P8221,
OutPadQS4 => P8222,
OutPadQS5 => P8266,
OutPadQS6 => P8223,
OutPadQS7 => P8224,
OutPadQS8 => P8225,
OutPadQS9 => P8226,
OutPadQSOV => P8233,
PSV => PSV,
RBinSelect0 => RB_82_inSelect0,
RBinSelect1 => RB_82_inSelect1,
RBinSelect2 => RB_82_inSelect2,
UNP => UNP,
URB => URB,
\InPad~I0\ => P8134,
\InPad~I10\ => P8144,
\InPad~I11\ => P8145,
\InPad~I12\ => P8146,
\InPad~I13\ => P8147,
\InPad~I14\ => P8148,
\InPad~I15\ => P8149,
\InPad~I1\ => P8135,
\InPad~I2\ => P8136,
\InPad~I3\ => P8137,
\InPad~I4\ => P8138,
\InPad~I5\ => P8139,
\InPad~I6\ => P8140,
\InPad~I7\ => P8141,
\InPad~I8\ => P8142,
\InPad~I9\ => P8143,
\InPad~Q0\ => P8150,
\InPad~Q10\ => P8160,
\InPad~Q11\ => P8161,
\InPad~Q12\ => P8162,
\InPad~Q13\ => P8163,
\InPad~Q14\ => P8164,
\InPad~Q15\ => P8165,
\InPad~Q1\ => P8151,
\InPad~Q2\ => P8152,
\InPad~Q3\ => P8153,
\InPad~Q4\ => P8154,
\InPad~Q5\ => P8155,
\InPad~Q6\ => P8156,
\InPad~Q7\ => P8157,
\InPad~Q8\ => P8158,
\InPad~Q9\ => P8159,
\OutPad~IS0\ => P8234,
\OutPad~IS10\ => P8244,
\OutPad~IS11\ => P8245,
\OutPad~IS12\ => P8246,
\OutPad~IS13\ => P8247,
\OutPad~IS14\ => P8248,
\OutPad~IS15\ => P8249,
\OutPad~IS1\ => P8235,
\OutPad~IS2\ => P8236,
\OutPad~IS3\ => P8237,
\OutPad~IS4\ => P8238,
\OutPad~IS5\ => P8239,
\OutPad~IS6\ => P8240,
\OutPad~IS7\ => P8241,
\OutPad~IS8\ => P8242,
\OutPad~IS9\ => P8243,
\OutPad~QS0\ => P8250,
\OutPad~QS10\ => P8260,
\OutPad~QS11\ => P8261,
\OutPad~QS12\ => P8262,
\OutPad~QS13\ => P8263,
\OutPad~QS14\ => P8264,
\OutPad~QS15\ => P8265,

```
            \OutPad~QS1\ => P8251,
            \OutPad~QS2\ => P8252,
            \OutPad~QS3\ => P8253,
            \OutPad~QS4\ => P8254,
            \OutPad~QS5\ => P8255,
            \OutPad~QS6\ => P8256,
            \OutPad~QS7\ => P8257,
            \OutPad~QS8\ => P8258,
            \OutPad~QS9\ => P8259
        );

    \HB_RangeBinModulator_32_3\ : DTM_8RBPs
        port map(
            CLK => CLK,
            DRFM0 => DRFM0,
            DRFM1 => DRFM1,
            DRFM2 => DRFM2,
            DRFM3 => DRFM3,
            DRFM4 => DRFM4,
            ENABLE => ENABLE_3,
            Gain0 => Gain0,
            Gain1 => Gain1,
            Gain2 => Gain2,
            Gain3 => Gain3,
            InPadI0 => P8201,
            InPadI1 => P8202,
            InPadI10 => P8211,
            InPadI11 => P8212,
            InPadI12 => P8213,
            InPadI13 => P8214,
            InPadI14 => P8215,
            InPadI15 => P8216,
            InPadI2 => P8203,
            InPadI3 => P8204,
            InPadI4 => P8205,
            InPadI5 => P8206,
            InPadI6 => P8207,
            InPadI7 => P8208,
            InPadI8 => P8209,
            InPadI9 => P8210,
            InPadIOV => P8217,
            InPadQ0 => P8218,
            InPadQ1 => P8219,
            InPadQ10 => P8227,
            InPadQ11 => P8228,
            InPadQ12 => P8229,
            InPadQ13 => P8230,
            InPadQ14 => P8231,
            InPadQ15 => P8232,
            InPadQ2 => P8220,
            InPadQ3 => P8221,
            InPadQ4 => P8222,
            InPadQ5 => P8266,
            InPadQ6 => P8223,
            InPadQ7 => P8224,
            InPadQ8 => P8225,
            InPadQ9 => P8226,
            InPadQOV => P8233,
            Inc0 => Inc0,
            Inc1 => Inc1,
            Inc2 => Inc2,
            Inc3 => Inc3,
            Inc4 => Inc4,
            ODVin => ODVin,
            ODVout => ODVout,
            Oper => Oper,
            OutPadIS0 => P8301,
            OutPadIS1 => P8302,
            OutPadIS10 => P8311,
            OutPadIS11 => P8312,
            OutPadIS12 => P8313,
            OutPadIS13 => P8314,
            OutPadIS14 => P8315,
            OutPadIS15 => P8316,
            OutPadIS2 => P8303,
            OutPadIS3 => P8304,
            OutPadIS4 => P8305,
            OutPadIS5 => P8306,
            OutPadIS6 => P8307,
            OutPadIS7 => P8308,
            OutPadIS8 => P8309,
            OutPadIS9 => P8310,
            OutPadISOV => P8317,
            OutPadQS0 => P8318,
            OutPadQS1 => P8319,
            OutPadQS10 => P8327,
            OutPadQS11 => P8328,
            OutPadQS12 => P8329,
            OutPadQS13 => P8330,
            OutPadQS14 => P8331,
            OutPadQS15 => P8332,
            OutPadQS2 => P8320,
            OutPadQS3 => P8321,
            OutPadQS4 => P8322,
            OutPadQS5 => P8366,
            OutPadQS6 => P8323,
            OutPadQS7 => P8324,
            OutPadQS8 => P8325,
            OutPadQS9 => P8326,
            OutPadQSOV => P8333,
            PSV => PSV,
            RBinSelect0 => RB_83_inSelect0,
            RBinSelect1 => RB_83_inSelect1,
            RBinSelect2 => RB_83_inSelect2,
            UNP => UNP,
            URB => URB,
            \InPad~I0\ => P8234,
            \InPad~I10\ => P8244,
            \InPad~I11\ => P8245,
            \InPad~I12\ => P8246,
            \InPad~I13\ => P8247,
```

```
\InPad~I14\ => P8248,
\InPad~I15\ => P8249,
\InPad~I1\ => P8235,
\InPad~I2\ => P8236,
\InPad~I3\ => P8237,
\InPad~I4\ => P8238,
\InPad~I5\ => P8239,
\InPad~I6\ => P8240,
\InPad~I7\ => P8241,
\InPad~I8\ => P8242,
\InPad~I9\ => P8243,
\InPad~Q0\ => P8250,
\InPad~Q10\ => P8260,
\InPad~Q11\ => P8261,
\InPad~Q12\ => P8262,
\InPad~Q13\ => P8263,
\InPad~Q14\ => P8264,
\InPad~Q15\ => P8265,
\InPad~Q1\ => P8251,
\InPad~Q2\ => P8252,
\InPad~Q3\ => P8253,
\InPad~Q4\ => P8254,
\InPad~Q5\ => P8255,
\InPad~Q6\ => P8256,
\InPad~Q7\ => P8257,
\InPad~Q8\ => P8258,
\InPad~Q9\ => P8259,
\OutPad~IS0\ => P8334,
\OutPad~IS10\ => P8344,
\OutPad~IS11\ => P8345,
\OutPad~IS12\ => P8346,
\OutPad~IS13\ => P8347,
\OutPad~IS14\ => P8348,
\OutPad~IS15\ => P8349,
\OutPad~IS1\ => P8335,
\OutPad~IS2\ => P8336,
\OutPad~IS3\ => P8337,
\OutPad~IS4\ => P8338,
\OutPad~IS5\ => P8339,
\OutPad~IS6\ => P8340,
\OutPad~IS7\ => P8341,
\OutPad~IS8\ => P8342,
\OutPad~IS9\ => P8343,
\OutPad~QS0\ => P8350,
\OutPad~QS10\ => P8360,
\OutPad~QS11\ => P8361,
\OutPad~QS12\ => P8362,
\OutPad~QS13\ => P8363,
\OutPad~QS14\ => P8364,
\OutPad~QS15\ => P8365,
\OutPad~QS1\ => P8351,
\OutPad~QS2\ => P8352,
\OutPad~QS3\ => P8353,
\OutPad~QS4\ => P8354,
\OutPad~QS5\ => P8355,
\OutPad~QS6\ => P8356,
\OutPad~QS7\ => P8357,
\OutPad~QS8\ => P8358,
\OutPad~QS9\ => P8359
);

\HB_RangeBinModulator_32_4\ : DTM_8RBPs
port map(
CLK => CLK,
DRFM0 => DRFM0,
DRFM1 => DRFM1,
DRFM2 => DRFM2,
DRFM3 => DRFM3,
DRFM4 => DRFM4,
ENABLE => ENABLE_4,
Gain0 => Gain0,
Gain1 => Gain1,
Gain2 => Gain2,
Gain3 => Gain3,
InPadI0 => P8301,
InPadI1 => P8302,
InPadI10 => P8311,
InPadI11 => P8312,
InPadI12 => P8313,
InPadI13 => P8314,
InPadI14 => P8315,
InPadI15 => P8316,
InPadI2 => P8303,
InPadI3 => P8304,
InPadI4 => P8305,
InPadI5 => P8306,
InPadI6 => P8307,
InPadI7 => P8308,
InPadI8 => P8309,
InPadI9 => P8310,
InPadIOV => P8317,
InPadQ0 => P8318,
InPadQ1 => P8319,
InPadQ10 => P8327,
InPadQ11 => P8328,
InPadQ12 => P8329,
InPadQ13 => P8330,
InPadQ14 => P8331,
InPadQ15 => P8332,
InPadQ2 => P8320,
InPadQ3 => P8321,
InPadQ4 => P8322,
InPadQ5 => P8366,
InPadQ6 => P8323,
InPadQ7 => P8324,
InPadQ8 => P8325,
InPadQ9 => P8326,
InPadQOV => P8333,
Inc0 => Inc0,
Inc1 => Inc1,
Inc2 => Inc2,
```

```
Inc3 => Inc3,
Inc4 => Inc4,
ODVin => ODVin,
ODVout => ODVout,
Oper => Oper,
OutPadIS0 => OutPadIS0,
OutPadIS1 => OutPadIS1,
OutPadIS10 => OutPadIS10,
OutPadIS11 => OutPadIS11,
OutPadIS12 => OutPadIS12,
OutPadIS13 => OutPadIS13,
OutPadIS14 => OutPadIS14,
OutPadIS15 => OutPadIS15,
OutPadIS2 => OutPadIS2,
OutPadIS3 => OutPadIS3,
OutPadIS4 => OutPadIS4,
OutPadIS5 => OutPadIS5,
OutPadIS6 => OutPadIS6,
OutPadIS7 => OutPadIS7,
OutPadIS8 => OutPadIS8,
OutPadIS9 => OutPadIS9,
OutPadISOV => OutPadISOV,
OutPadQS0 => OutPadQS0,
OutPadQS1 => OutPadQS1,
OutPadQS10 => OutPadQS10,
OutPadQS11 => OutPadQS11,
OutPadQS12 => OutPadQS12,
OutPadQS13 => OutPadQS13,
OutPadQS14 => OutPadQS14,
OutPadQS15 => OutPadQS15,
OutPadQS2 => OutPadQS2,
OutPadQS3 => OutPadQS3,
OutPadQS4 => OutPadQS4,
OutPadQS5 => OutPadQS5,
OutPadQS6 => OutPadQS6,
OutPadQS7 => OutPadQS7,
OutPadQS8 => OutPadQS8,
OutPadQS9 => OutPadQS9,
OutPadQSOV => OutPadQSOV,
PSV => PSV,
RBinSelect0 => RB_84_inSelect0,
RBinSelect1 => RB_84_inSelect1,
RBinSelect2 => RB_84_inSelect2,
UNP => UNP,
URB => URB,
\InPad~I0\ => P8334,
\InPad~I10\ => P8344,
\InPad~I11\ => P8345,
\InPad~I12\ => P8346,
\InPad~I13\ => P8347,
\InPad~I14\ => P8348,
\InPad~I15\ => P8349,
\InPad~I1\ => P8335,
\InPad~I2\ => P8336,
\InPad~I3\ => P8337,
\InPad~I4\ => P8338,
\InPad~I5\ => P8339,
\InPad~I6\ => P8340,
\InPad~I7\ => P8341,
\InPad~I8\ => P8342,
\InPad~I9\ => P8343,
\InPad~Q0\ => P8350,
\InPad~Q10\ => P8360,
\InPad~Q11\ => P8361,
\InPad~Q12\ => P8362,
\InPad~Q13\ => P8363,
\InPad~Q14\ => P8364,
\InPad~Q15\ => P8365,
\InPad~Q1\ => P8351,
\InPad~Q2\ => P8352,
\InPad~Q3\ => P8353,
\InPad~Q4\ => P8354,
\InPad~Q5\ => P8355,
\InPad~Q6\ => P8356,
\InPad~Q7\ => P8357,
\InPad~Q8\ => P8358,
\InPad~Q9\ => P8359,
\OutPad~IS0\ => \OutPad~IS0\,
\OutPad~IS10\ => \OutPad~IS10\,
\OutPad~IS11\ => \OutPad~IS11\,
\OutPad~IS12\ => \OutPad~IS12\,
\OutPad~IS13\ => \OutPad~IS13\,
\OutPad~IS14\ => \OutPad~IS14\,
\OutPad~IS15\ => \OutPad~IS15\,
\OutPad~IS1\ => \OutPad~IS1\,
\OutPad~IS2\ => \OutPad~IS2\,
\OutPad~IS3\ => \OutPad~IS3\,
\OutPad~IS4\ => \OutPad~IS4\,
\OutPad~IS5\ => \OutPad~IS5\,
\OutPad~IS6\ => \OutPad~IS6\,
\OutPad~IS7\ => \OutPad~IS7\,
\OutPad~IS8\ => \OutPad~IS8\,
\OutPad~IS9\ => \OutPad~IS9\,
\OutPad~QS0\ => \OutPad~QS0\,
\OutPad~QS10\=> \OutPad~QS10\,
\OutPad~QS11\=> \OutPad~QS11\,
\OutPad~QS12\=> \OutPad~QS12\,
\OutPad~QS13\=> \OutPad~QS13\,
\OutPad~QS14\=> \OutPad~QS14\,
\OutPad~QS15\=> \OutPad~QS15\,
\OutPad~QS1\ => \OutPad~QS1\,
\OutPad~QS2\ => \OutPad~QS2\,
\OutPad~QS3\ => \OutPad~QS3\,
\OutPad~QS4\ => \OutPad~QS4\,
\OutPad~QS5\ => \OutPad~QS5\,
\OutPad~QS6\ => \OutPad~QS6\,
\OutPad~QS7\ => \OutPad~QS7\,
\OutPad~QS8\ => \OutPad~QS8\,
\OutPad~QS9\ => \OutPad~QS9\
);

end structural;
```

B. TEST BENCH FOR THE 32 RANGE BIN MODULATOR

```
-----------------------------------------------
--
-- Title      : Test Bench for hb_32rbps
-- Design     : HB_32_RB_2
-- Author     : Hakan Bergon
-- Company    : NPS
--
-----------------------------------------------
--
-- File       : $DSN\src\TestBench\hb_32rbps_TB.vhd
-- Generated  : 7/29/2002, 9:02 AM
-- From       : $DSN\src\hb_32rbps.vhd
-- By         : Active-HDL Built-in Test Bench Generator ver. 1.2s
--
-- Description : Automatically generated Test Bench for hb_32rbps_tb
--

library ieee;
use ieee.std_logic_1164.all;

        -- Add your library and packages declaration here ...

entity hb_32rbps_tb is
end hb_32rbps_tb;

architecture TB_ARCHITECTURE of hb_32rbps_tb is
        -- Component declaration of the tested unit
        component hb_32rbps
        port(
        CLK : in std_logic;
        DRFM0 : in std_logic;
        DRFM1 : in std_logic;
        DRFM2 : in std_logic;
        DRFM3 : in std_logic;
        DRFM4 : in std_logic;
        ENABLE_1 : in std_logic;
        ENABLE_2 : in std_logic;
        ENABLE_3 : in std_logic;
        ENABLE_4 : in std_logic;
        Gain0 : in std_logic;
        Gain1 : in std_logic;
        Gain2 : in std_logic;
        Gain3 : in std_logic;
        Inc0 : in std_logic;
        Inc1 : in std_logic;
        Inc2 : in std_logic;
        Inc3 : in std_logic;
        Inc4 : in std_logic;
        InPadI0 : in std_logic;
        InPadI1 : in std_logic;
        InPadI2 : in std_logic;
        InPadI3 : in std_logic;
        InPadI4 : in std_logic;
        InPadI5 : in std_logic;
        InPadI6 : in std_logic;
        InPadI7 : in std_logic;
        InPadI8 : in std_logic;
        InPadI9 : in std_logic;
        InPadI10 : in std_logic;
        InPadI11 : in std_logic;
        InPadI12 : in std_logic;
        InPadI13 : in std_logic;
        InPadI14 : in std_logic;
        InPadI15 : in std_logic;
        InPadIOV : in std_logic;
        InPadQ0 : in std_logic;
        InPadQ1 : in std_logic;
        InPadQ2 : in std_logic;
        InPadQ3 : in std_logic;
        InPadQ4 : in std_logic;
        InPadQ5 : in std_logic;
        InPadQ6 : in std_logic;
        InPadQ7 : in std_logic;
        InPadQ8 : in std_logic;
        InPadQ9 : in std_logic;
        InPadQ10 : in std_logic;
        InPadQ11 : in std_logic;
        InPadQ12 : in std_logic;
        InPadQ13 : in std_logic;
        InPadQ14 : in std_logic;
        InPadQ15 : in std_logic;
        InPadQOV : in std_logic;
        \InPad~I0\ : in std_logic;
        \InPad~I1\ : in std_logic;
        \InPad~I2\ : in std_logic;
        \InPad~I3\ : in std_logic;
        \InPad~I4\ : in std_logic;
        \InPad~I5\ : in std_logic;
        \InPad~I6\ : in std_logic;
        \InPad~I7\ : in std_logic;
        \InPad~I8\ : in std_logic;
        \InPad~I9\ : in std_logic;
        \InPad~I10\ : in std_logic;
        \InPad~I11\ : in std_logic;
        \InPad~I12\ : in std_logic;
        \InPad~I13\ : in std_logic;
        \InPad~I14\ : in std_logic;
        \InPad~I15\ : in std_logic;
        \InPad~Q0\ : in std_logic;
        \InPad~Q1\ : in std_logic;
        \InPad~Q2\ : in std_logic;
        \InPad~Q3\ : in std_logic;
```

```
\InPad~Q4\ : in std_logic;
\InPad~Q5\ : in std_logic;
\InPad~Q6\ : in std_logic;
\InPad~Q7\ : in std_logic;
\InPad~Q8\ : in std_logic;
\InPad~Q9\ : in std_logic;
\InPad~Q10\ : in std_logic;
\InPad~Q11\ : in std_logic;
\InPad~Q12\ : in std_logic;
\InPad~Q13\ : in std_logic;
\InPad~Q14\ : in std_logic;
\InPad~Q15\: in std_logic;
ODVin : in std_logic;
ODVout : out std_logic;
Oper : in std_logic;
OutPadIS0 : out std_logic;
OutPadIS1 : out std_logic;
OutPadIS2 : out std_logic;
OutPadIS3 : out std_logic;
OutPadIS4 : out std_logic;
OutPadIS5 : out std_logic;
OutPadIS6 : out std_logic;
OutPadIS7 : out std_logic;
OutPadIS8 : out std_logic;
OutPadIS9 : out std_logic;
OutPadIS10 : out std_logic;
OutPadIS11 : out std_logic;
OutPadIS12 : out std_logic;
OutPadIS13 : out std_logic;
OutPadIS14 : out std_logic;
OutPadIS15 : out std_logic;
OutPadISOV : out std_logic;
OutPadQS0 : out std_logic;
OutPadQS1 : out std_logic;
OutPadQS2 : out std_logic;
OutPadQS3 : out std_logic;
OutPadQS4 : out std_logic;
OutPadQS5 : out std_logic;
OutPadQS6 : out std_logic;
OutPadQS7 : out std_logic;
OutPadQS8 : out std_logic;
OutPadQS9 : out std_logic;
OutPadQS10 : out std_logic;
OutPadQS11 : out std_logic;
OutPadQS12 : out std_logic;
OutPadQS13 : out std_logic;
OutPadQS14 : out std_logic;
OutPadQS15 : out std_logic;
OutPadQSOV : out std_logic;
\OutPad~IS0\ : out std_logic;
\OutPad~IS1\ : out std_logic;
\OutPad~IS2\ : out std_logic;
\OutPad~IS3\ : out std_logic;
\OutPad~IS4\ : out std_logic;
\OutPad~IS5\ : out std_logic;
OutPad~IS6\ : out std_logic;
\OutPad~IS7\ : out std_logic;
\OutPad~IS8\ : out std_logic;
\OutPad~IS9\ : out std_logic;
\OutPad~IS10\ : out std_logic;
\OutPad~IS11\ : out std_logic;
\OutPad~IS12\ : out std_logic;
\OutPad~IS13\ : out std_logic;
\OutPad~IS14\ : out std_logic;
\OutPad~IS15\ : out std_logic;
\OutPad~QS0\ : out std_logic;
\OutPad~QS1\ : out std_logic;
\OutPad~QS2\ : out std_logic;
\OutPad~QS3\ : out std_logic;
\OutPad~QS4\ : out std_logic;
\OutPad~QS5\ : out std_logic;
\OutPad~QS6\ : out std_logic;
\OutPad~QS7\ : out std_logic;
\OutPad~QS8\ : out std_logic;
\OutPad~QS9\ : out std_logic;
\OutPad~QS10\: out std_logic;
\OutPad~QS11\: out std_logic;
\OutPad~QS12\: out std_logic;
\OutPad~QS13\: out std_logic;
\OutPad~QS14\: out std_logic;
\OutPad~QS15\: out std_logic;
PSV : in std_logic;
RB_81_inSelect0: in std_logic;
RB_81_inSelect1: in std_logic;
RB_81_inSelect2: in std_logic;
RB_82_inSelect0: in std_logic;
RB_82_inSelect1: in std_logic;
RB_82_inSelect2: in std_logic;
RB_83_inSelect0: in std_logic;
RB_83_inSelect1: in std_logic;
RB_83_inSelect2: in std_logic;
RB_84_inSelect0: in std_logic;
RB_84_inSelect1: in std_logic;
RB_84_inSelect2: in std_logic;
UNP : in std_logic;
URB : in std_logic );
end component;

        -- Stimulus signals - signals mapped to
the input and inout ports of tested entity
signal CLK : std_logic;
signal DRFM0 : std_logic;
signal DRFM1 : std_logic;
signal DRFM2 : std_logic;
signal DRFM3 : std_logic;
signal DRFM4 : std_logic;
signal ENABLE_1 : std_logic;
signal ENABLE_2 : std_logic;
signal ENABLE_3 : std_logic;
signal ENABLE_4 : std_logic;
signal Gain0 : std_logic;
signal Gain1 : std_logic;
```

```
signal Gain2 : std_logic;
signal Gain3 : std_logic;
signal Inc0 : std_logic;
signal Inc1 : std_logic;
signal Inc2 : std_logic;
signal Inc3 : std_logic;
signal Inc4 : std_logic;
signal InPadI0 : std_logic;
signal InPadI1 : std_logic;
signal InPadI2 : std_logic;
signal InPadI3 : std_logic;
signal InPadI4 : std_logic;
signal InPadI5 : std_logic;
signal InPadI6 : std_logic;
signal InPadI7 : std_logic;
signal InPadI8 : std_logic;
signal InPadI9 : std_logic;
signal InPadI10 : std_logic;
signal InPadI11 : std_logic;
signal InPadI12 : std_logic;
signal InPadI13 : std_logic;
signal InPadI14 : std_logic;
signal InPadI15 : std_logic;
signal InPadIOV : std_logic;
signal InPadQ0 : std_logic;
signal InPadQ1 : std_logic;
signal InPadQ2 : std_logic;
signal InPadQ3 : std_logic;
signal InPadQ4 : std_logic;
signal InPadQ5 : std_logic;
signal InPadQ6 : std_logic;
signal InPadQ7 : std_logic;
signal InPadQ8 : std_logic;
signal InPadQ9 : std_logic;
signal InPadQ10 : std_logic;
signal InPadQ11 : std_logic;
signal InPadQ12 : std_logic;
signal InPadQ13 : std_logic;
signal InPadQ14 : std_logic;
signal InPadQ15 : std_logic;
signal InPadQOV : std_logic;
signal \InPad~I0\ : std_logic;
signal \InPad~I1\ : std_logic;
signal \InPad~I2\ : std_logic;
signal \InPad~I3\ : std_logic;
signal \InPad~I4\ : std_logic;
signal \InPad~I5\ : std_logic;
signal \InPad~I6\ : std_logic;
signal \InPad~I7\ : std_logic;
signal \InPad~I8\ : std_logic;
signal \InPad~I9\ : std_logic;
signal \InPad~I10\ : std_logic;
signal \InPad~I11\ : std_logic;
signal \InPad~I12\ : std_logic;
signal \InPad~I13\ : std_logic;
signal \InPad~I14\ : std_logic;
signal \InPad~I15\ : std_logic;
signal \InPad~Q0\ : std_logic;
signal \InPad~Q1\ : std_logic;
signal \InPad~Q2\ : std_logic;
signal \InPad~Q3\ : std_logic;
signal \InPad~Q4\ : std_logic;
signal \InPad~Q5\ : std_logic;
signal \InPad~Q6\ : std_logic;
signal \InPad~Q7\ : std_logic;
signal \InPad~Q8\ : std_logic;
signal \InPad~Q9\ : std_logic;
signal \InPad~Q10\ : std_logic;
signal \InPad~Q11\ : std_logic;
signal \InPad~Q12\ : std_logic;
signal \InPad~Q13\ : std_logic;
signal \InPad~Q14\ : std_logic;
signal \InPad~Q15\ : std_logic;
signal ODVin : std_logic;
signal Oper : std_logic;
signal PSV : std_logic;
signalRB_81_inSelect0: std_logic;
signal RB_81_inSelect1 : std_logic;
signal RB_81_inSelect2 : std_logic;
signal RB_82_inSelect0 : std_logic;
signal RB_82_inSelect1 : std_logic;
signal RB_82_inSelect2 : std_logic;
signal RB_83_inSelect0 : std_logic;
signal RB_83_inSelect1 : std_logic;
signal RB_83_inSelect2 : std_logic;
signal RB_84_inSelect0 : std_logic;
signal RB_84_inSelect1 : std_logic;
signal RB_84_inSelect2 : std_logic;
signal UNP : std_logic;
signal URB : std_logic;
        -- Observed signals - signals
mapped to the output ports of tested entity
signal ODVout : std_logic;
signal OutPadIS0 : std_logic;
signal OutPadIS1 : std_logic;
signal OutPadIS2 : std_logic;
signal OutPadIS3 : std_logic;
signal OutPadIS4 : std_logic;
signal OutPadIS5 : std_logic;
signal OutPadIS6 : std_logic;
signal OutPadIS7 : std_logic;
signal OutPadIS8 : std_logic;
signal OutPadIS9 : std_logic;
signal OutPadIS10 : std_logic;
signal OutPadIS11 : std_logic;
signal OutPadIS12 : std_logic;
signal OutPadIS13 : std_logic;
signal OutPadIS14 : std_logic;
signal OutPadIS15 : std_logic;
signal OutPadISOV : std_logic;
signal OutPadQS0 : std_logic;
signal OutPadQS1 : std_logic;
```

```
        signal OutPadQS2 : std_logic;
        signal OutPadQS3 : std_logic;
        signal OutPadQS4 : std_logic;
        signal OutPadQS5 : std_logic;
        signal OutPadQS6 : std_logic;
        signal OutPadQS7 : std_logic;
        signal OutPadQS8 : std_logic;
        signal OutPadQS9 : std_logic;
    signal OutPadQS10 : std_logic;
    signal OutPadQS11 : std_logic;
    signal OutPadQS12 : std_logic;
    signal OutPadQS13 : std_logic;
    signal OutPadQS14 : std_logic;
    signal OutPadQS15 : std_logic;
    signal OutPadQSOV : std_logic;
    signal \OutPad~IS0\ : std_logic;
    signal \OutPad~IS1\ : std_logic;
    signal \OutPad~IS2\ : std_logic;
    signal \OutPad~IS3\ : std_logic;
    signal \OutPad~IS4\ : std_logic;
    signal \OutPad~IS5\ : std_logic;
    signal \OutPad~IS6\ : std_logic;
    signal \OutPad~IS7\ : std_logic;
    signal \OutPad~IS8\ : std_logic;
    signal \OutPad~IS9\ : std_logic;
    signal \OutPad~IS10\ : std_logic;
    signal \OutPad~IS11\ : std_logic;
    signal \OutPad~IS12\ : std_logic;
    signal \OutPad~IS13\ : std_logic;
    signal \OutPad~IS14\ : std_logic;
    signal \OutPad~IS15\ : std_logic;
    signal \OutPad~QS0\ : std_logic;
    signal \OutPad~QS1\ : std_logic;
    signal \OutPad~QS2\ : std_logic;
    signal \OutPad~QS3\ : std_logic;
    signal \OutPad~QS4\ : std_logic;
    signal \OutPad~QS5\ : std_logic;
    signal \OutPad~QS6\ : std_logic;
    signal \OutPad~QS7\ : std_logic;
    signal \OutPad~QS8\ : std_logic;
    signal \OutPad~QS9\ : std_logic;
    signal \OutPad~QS10\ : std_logic;
    signal \OutPad~QS11\ : std_logic;
    signal \OutPad~QS12\ : std_logic;
    signal \OutPad~QS13\ : std_logic;
    signal \OutPad~QS14\ : std_logic;
    signal \OutPad~QS15\ : std_logic;
    --Signal is used to stop clock signal
generators
    signalEND_SIM:BOOLEAN:=FALSE;

        -- Add your code here ...

    begin

        -- Unit Under Test port map
UUT : hb_32rbps
        port map (
        CLK => CLK,
        DRFM0 => DRFM0,
        DRFM1 => DRFM1,
        DRFM2 => DRFM2,
        DRFM3 => DRFM3,
        DRFM4 => DRFM4,
        ENABLE_1 => ENABLE_1,
        ENABLE_2 => ENABLE_2,
        ENABLE_3 => ENABLE_3,
        ENABLE_4 => ENABLE_4,
        Gain0 => Gain0,
        Gain1 => Gain1,
        Gain2 => Gain2,
        Gain3 => Gain3,
        Inc0 => Inc0,
        Inc1 => Inc1,
        Inc2 => Inc2,
        Inc3 => Inc3,
        Inc4 => Inc4,
        InPadI0 => InPadI0,
        InPadI1 => InPadI1,
        InPadI2 => InPadI2,
        InPadI3 => InPadI3,
        InPadI4 => InPadI4,
        InPadI5 => InPadI5,
        InPadI6 => InPadI6,
        InPadI7 => InPadI7,
        InPadI8 => InPadI8,
        InPadI9 => InPadI9,
        InPadI10 => InPadI10,
        InPadI11 => InPadI11,
        InPadI12 => InPadI12,
        InPadI13 => InPadI13,
        InPadI14 => InPadI14,
        InPadI15 => InPadI15,
        InPadIOV => InPadIOV,
        InPadQ0 => InPadQ0,
        InPadQ1 => InPadQ1,
        InPadQ2 => InPadQ2,
        InPadQ3 => InPadQ3,
        InPadQ4 => InPadQ4,
        InPadQ5 => InPadQ5,
        InPadQ6 => InPadQ6,
        InPadQ7 => InPadQ7,
        InPadQ8 => InPadQ8,
        InPadQ9 => InPadQ9,
        InPadQ10 => InPadQ10,
        InPadQ11 => InPadQ11,
        InPadQ12 => InPadQ12,
        InPadQ13 => InPadQ13,
        InPadQ14 => InPadQ14,
        InPadQ15 => InPadQ15,
        InPadQOV => InPadQOV,
        \InPad~I0\ => \InPad~I0\,
```

```
\InPad~I1\ => \InPad~I1\,
\InPad~I2\ => \InPad~I2\,
\InPad~I3\ => \InPad~I3\,
\InPad~I4\ => \InPad~I4\,
\InPad~I5\ => \InPad~I5\,
\InPad~I6\ => \InPad~I6\,
\InPad~I7\ => \InPad~I7\,
\InPad~I8\ => \InPad~I8\,
\InPad~I9\ => \InPad~I9\,
\InPad~I10\ => \InPad~I10\,
\InPad~I11\ => \InPad~I11\,
\InPad~I12\ => \InPad~I12\,
\InPad~I13\ => \InPad~I13\,
\InPad~I14\ => \InPad~I14\,
\InPad~I15\ => \InPad~I15\,
\InPad~Q0\ => \InPad~Q0\,
\InPad~Q1\ => \InPad~Q1\,
\InPad~Q2\ => \InPad~Q2\,
\InPad~Q3\ => \InPad~Q3\,
\InPad~Q4\ => \InPad~Q4\,
\InPad~Q5\ => \InPad~Q5\,
\InPad~Q6\ => \InPad~Q6\,
\InPad~Q7\ => \InPad~Q7\,
\InPad~Q8\ => \InPad~Q8\,
\InPad~Q9\ => \InPad~Q9\,
\InPad~Q10\ => \InPad~Q10\,
\InPad~Q11\ => \InPad~Q11\,
\InPad~Q12\ => \InPad~Q12\,
\InPad~Q13\ => \InPad~Q13\,
\InPad~Q14\ => \InPad~Q14\,
\InPad~Q15\ => \InPad~Q15\,
ODVin => ODVin,
ODVout => ODVout,
Oper => Oper,
OutPadIS0 => OutPadIS0,
OutPadIS1 => OutPadIS1,
OutPadIS2 => OutPadIS2,
OutPadIS3 => OutPadIS3,
OutPadIS4 => OutPadIS4,
OutPadIS5 => OutPadIS5,
OutPadIS6 => OutPadIS6,
OutPadIS7 => OutPadIS7,
OutPadIS8 => OutPadIS8,
OutPadIS9 => OutPadIS9,
OutPadIS10 => OutPadIS10,
OutPadIS11 => OutPadIS11,
OutPadIS12 => OutPadIS12,
OutPadIS13 => OutPadIS13,
OutPadIS14 => OutPadIS14,
OutPadIS15 => OutPadIS15,
OutPadISOV => OutPadISOV,
OutPadQS0 => OutPadQS0,
OutPadQS1 => OutPadQS1,
OutPadQS2 => OutPadQS2,
OutPadQS3 => OutPadQS3,
OutPadQS4 => OutPadQS4,
OutPadQS5 => OutPadQS5,
OutPadQS6 => OutPadQS6,
OutPadQS7 => OutPadQS7,
OutPadQS8 => OutPadQS8,
OutPadQS9 => OutPadQS9,
OutPadQS10 => OutPadQS10,
OutPadQS11 => OutPadQS11,
OutPadQS12 => OutPadQS12,
OutPadQS13 => OutPadQS13,
OutPadQS14 => OutPadQS14,
OutPadQS15 => OutPadQS15,
OutPadQSOV => OutPadQSOV,
\OutPad~IS0\ => \OutPad~IS0\,
\OutPad~IS1\ => \OutPad~IS1\,
\OutPad~IS2\ => \OutPad~IS2\,
\OutPad~IS3\ => \OutPad~IS3\,
\OutPad~IS4\ => \OutPad~IS4\,
\OutPad~IS5\ => \OutPad~IS5\,
\OutPad~IS6\ => \OutPad~IS6\,
\OutPad~IS7\ => \OutPad~IS7\,
\OutPad~IS8\ => \OutPad~IS8\,
\OutPad~IS9\ => \OutPad~IS9\,
\OutPad~IS10\ => \OutPad~IS10\,
\OutPad~IS11\ => \OutPad~IS11\,
\OutPad~IS12\ => \OutPad~IS12\,
\OutPad~IS13\ => \OutPad~IS13\,
\OutPad~IS14\ => \OutPad~IS14\,
\OutPad~IS15\ => \OutPad~IS15\,
\OutPad~QS0\ => \OutPad~QS0\,
\OutPad~QS1\ => \OutPad~QS1\,
\OutPad~QS2\ => \OutPad~QS2\,
\OutPad~QS3\ => \OutPad~QS3\,
\OutPad~QS4\ => \OutPad~QS4\,
\OutPad~QS5\ => \OutPad~QS5\,
\OutPad~QS6\ => \OutPad~QS6\,
\OutPad~QS7\ => \OutPad~QS7\,
\OutPad~QS8\ => \OutPad~QS8\,
\OutPad~QS9\ => \OutPad~QS9\,
\OutPad~QS10\ => \OutPad~QS10\,
\OutPad~QS11\ => \OutPad~QS11\,
\OutPad~QS12\ => \OutPad~QS12\,
\OutPad~QS13\ => \OutPad~QS13\,
\OutPad~QS14\ => \OutPad~QS14\,
\OutPad~QS15\ => \OutPad~QS15\,
PSV => PSV,
RB_81_inSelect0 => RB_81_inSelect0,
RB_81_inSelect1 => RB_81_inSelect1,
RB_81_inSelect2 => RB_81_inSelect2,
RB_82_inSelect0 => RB_82_inSelect0,
RB_82_inSelect1 => RB_82_inSelect1,
RB_82_inSelect2 => RB_82_inSelect2,
RB_83_inSelect0 => RB_83_inSelect0,
RB_83_inSelect1 => RB_83_inSelect1,
RB_83_inSelect2 => RB_83_inSelect2,
RB_84_inSelect0 => RB_84_inSelect0,
RB_84_inSelect1 => RB_84_inSelect1,
```

```
        RB_84_inSelect2 => RB_84_inSelect2,
        UNP => UNP,
        URB => URB
        );

                --Below VHDL code is an
inserted .\compile\Pulse 1 first 10 samples.vhs
                --User can modify it ....

        STIMULUS: process
        begin  -- of stimulus process
        --wait for <time to next event>; --
<current time>

                Gain0 <= '1';
                Gain1 <= '1';
                Gain2 <= '0';
                Gain3 <= '1';
                Inc0 <= '0';
                Inc1 <= '0';
                Inc2 <= '0';
                Inc3 <= '0';
                Inc4 <= '0';
                DRFM0 <= '0';
                DRFM1 <= '0';
                DRFM2 <= '0';
                DRFM3 <= '0';
                DRFM4 <= '0';
                ENABLE_1 <= '0';
                ENABLE_2 <= '0';
                ENABLE_3 <= '0';
                ENABLE_4 <= '1';
                UNP <= '0';
                ODVin <= '0';
                PSV <= '0';
                URB <= '1';
                Oper <= '1';
                InPadI0 <= '0';
                InPadI1 <= '0';
                InPadI2 <= '0';
                InPadI3 <= '0';
                InPadI4 <= '0';
                InPadI5 <= '0';
                InPadI6 <= '0';
                InPadI7 <= '0';
                InPadI8 <= '0';
                InPadI9 <= '0';
                InPadI10 <= '0';
                InPadI11 <= '0';
                InPadI12 <= '0';
                InPadI13 <= '0';
                InPadI14 <= '0';
                InPadI15 <= '0';
                InPadIOV <= '0';
                InPadQ0 <= '0';
                InPadQ1 <= '0';
                InPadQ2 <= '0';
                InPadQ3 <= '0';
                InPadQ4 <= '0';
                InPadQ5 <= '0';
                InPadQ6 <= '0';
                InPadQ7 <= '0';
                InPadQ8 <= '0';
                InPadQ9 <= '0';
                InPadQ10 <= '0';
                InPadQ11 <= '0';
                InPadQ12 <= '0';
                InPadQ13 <= '0';
                InPadQ14 <= '0';
                InPadQ15 <= '0';
                InPadQOV <= '0';

                \InPad~I0\ <= '1';
                \InPad~I1\ <= '1';
                \InPad~I2\ <= '1';
                \InPad~I3\ <= '1';
                \InPad~I4\ <= '1';
                \InPad~I5\ <= '1';
                \InPad~I6\ <= '1';
                \InPad~I7\ <= '1';
                \InPad~I8\ <= '1';
                \InPad~I9\ <= '1';
                \InPad~I10\ <= '1';
                \InPad~I11\ <= '1';
                \InPad~I12\ <= '1';
                \InPad~I13\ <= '1';
                \InPad~I14\ <= '1';
                \InPad~I15\ <= '1';
                \InPad~Q0\ <= '1';
                \InPad~Q1\ <= '1';
                \InPad~Q2\ <= '1';
                \InPad~Q3\ <= '1';
                \InPad~Q4\ <= '1';
                \InPad~Q5\ <= '1';
                \InPad~Q6\ <= '1';
                \InPad~Q7\ <= '1';
                \InPad~Q8\ <= '1';
                \InPad~Q9\ <= '1';
                \InPad~Q10\ <= '1';
                \InPad~Q11\ <= '1';
                \InPad~Q12\ <= '1';
                \InPad~Q13\ <= '1';
                \InPad~Q14\ <= '1';
                \InPad~Q15\ <= '1';
        wait for 2 ns; --2 ns
                Gain0 <= '1';
                Gain1 <= '1';
                Gain2 <= '0';
                Gain3 <= '1';
                Inc0 <= '0';
                Inc1 <= '0';
                Inc2 <= '0';
```

```
        Inc3 <= '0';
        Inc4 <= '0';
wait for 2 ns; --4 ns
        Gain0 <= '1';
        Gain1 <= '1';
        Gain2 <= '0';
        Gain3 <= '1';
        Inc0 <= '1';
        Inc1 <= '0';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --6 ns
        Gain0 <= '1';
        Gain1 <= '0';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '1';
        Inc1 <= '0';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --8 ns
        Gain0 <= '1';
        Gain1 <= '0';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '0';
        Inc1 <= '1';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --10 ns
        Gain0 <= '1';
        Gain1 <= '0';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '1';
        Inc1 <= '0';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --12 ns
        Gain0 <= '1';
        Gain1 <= '1';
        Gain2 <= '0';
        Gain3 <= '1';
        Inc0 <= '1';
        Inc1 <= '0';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --14 ns
        Gain0 <= '1';
        Gain1 <= '1';
        Gain2 <= '0';
        Gain3 <= '1';
        Inc0 <= '0';
        Inc1 <= '1';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --16 ns
        Gain0 <= '1';
        Gain1 <= '1';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '1';
        Inc1 <= '0';
        Inc2 <= '1';
        Inc3 <= '0';
        Inc4 <= '0';
        ENABLE_4 <= '0';
        ENABLE_3 <= '1';
        wait for 2 ns; --18 ns
        Gain0 <= '1';
        Gain1 <= '0';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '0';
        Inc1 <= '1';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --20 ns
        Gain0 <= '1';
        Gain1 <= '0';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '1';
        Inc1 <= '1';
        Inc2 <= '0';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --22 ns
        Gain0 <= '0';
        Gain1 <= '1';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '1';
        Inc1 <= '0';
        Inc2 <= '1';
        Inc3 <= '0';
        Inc4 <= '0';
        wait for 2 ns; --24 ns
        Gain0 <= '0';
        Gain1 <= '1';
        Gain2 <= '1';
        Gain3 <= '1';
        Inc0 <= '0';
        Inc1 <= '0';
        Inc2 <= '1';
```

```
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --26 ns
Gain0 <= '0';
Gain1 <= '1';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '1';
Inc1 <= '0';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --28 ns
Gain0 <= '0';
Gain1 <= '1';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '1';
Inc1 <= '0';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --30 ns
Gain0 <= '0';
Gain1 <= '1';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --32 ns
Gain0 <= '0';
Gain1 <= '1';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
ENABLE_2 <= '1';
ENABLE_3 <= '0';
wait for 2 ns; --34 ns
Gain0 <= '0';
Gain1 <= '1';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --36 ns
Gain0 <= '0';
Gain1 <= '1';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --38 ns
Gain0 <= '1';
Gain1 <= '0';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --40 ns
Gain0 <= '1';
Gain1 <= '0';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --42 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '0';
Gain3 <= '1';
Inc0 <= '1';
Inc1 <= '0';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --44 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '0';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --46 ns
Gain0 <= '1';
Gain1 <= '0';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
```

```
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --48 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '0';
Inc3 <= '1';
Inc4 <= '0';
ENABLE_2 <= '0';
ENABLE_1 <= '1';
wait for 2 ns; --50 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '0';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --52 ns
Gain0 <= '1';
Gain1 <= '0';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '0';
Inc2 <= '0';
Inc3 <= '1';
Inc4 <= '0';
wait for 2 ns; --54 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '0';
Gain3 <= '1';
Inc0 <= '1';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --56 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '0';
Gain3 <= '1';
Inc0 <= '1';
Inc1 <= '1';
Inc2 <= '1';
Inc3 <= '0';
Inc4 <= '0';
wait for 2 ns; --58 ns
Gain0 <= '1';
Gain1 <= '0';
Gain2 <= '1';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '0';
Inc2 <= '0';
Inc3 <= '1';
Inc4 <= '0';
wait for 2 ns; --60 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '0';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '0';
Inc2 <= '0';
Inc3 <= '1';
Inc4 <= '0';
wait for 2 ns; --64 ns
Gain0 <= '1';
Gain1 <= '1';
Gain2 <= '0';
Gain3 <= '1';
Inc0 <= '0';
Inc1 <= '0';
Inc2 <= '0';
Inc3 <= '1';
Inc4 <= '0';
wait for 2 ns; --64 ns
UNP <= '1';
ENABLE_1 <= '0';
wait for 2 ns; --66 ns
UNP <= '0';
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '0';
PSV <= '1';
wait for 14 ns; --80 ns
DRFM0 <= '1';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '0';
wait for 8 ns; --88 ns
DRFM0 <= '0';
DRFM1 <= '1';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '0';
wait for 4 ns; --92 ns
DRFM0 <= '1';
DRFM1 <= '1';
DRFM2 <= '0';
DRFM3 <= '0';
```

```
        DRFM4 <= '0';
        wait for 4 ns; --96 ns
        DRFM0 <= '0';
        DRFM1 <= '0';
        DRFM2 <= '1';
        DRFM3 <= '0';
        DRFM4 <= '0';
wait for 4 ns; --100 ns
        DRFM0 <= '1';
        DRFM1 <= '0';
        DRFM2 <= '1';
        DRFM3 <= '0';
        DRFM4 <= '0';
        wait for 4 ns; --104 ns
        DRFM0 <= '0';
        DRFM1 <= '1';
        DRFM2 <= '1';
        DRFM3 <= '0';
        DRFM4 <= '0';
        wait for 2 ns; --106 ns
        DRFM0 <= '1';
        DRFM1 <= '1';
        DRFM2 <= '1';
        DRFM3 <= '0';
        DRFM4 <= '0';
        wait for 4 ns; --110 ns
        DRFM0 <= '0';
        DRFM1 <= '0';
        DRFM2 <= '0';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 2 ns; --112 ns
        DRFM0 <= '1';
        DRFM1 <= '0';
        DRFM2 <= '0';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 2 ns; --114 ns
        DRFM0 <= '0';
        DRFM1 <= '1';
        DRFM2 <= '0';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 4 ns; --118 ns
        DRFM0 <= '1';
        DRFM1 <= '1';
        DRFM2 <= '0';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 2 ns; --120 ns
        DRFM0 <= '0';
        DRFM1 <= '0';
        DRFM2 <= '1';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 2 ns; --122 ns
        DRFM0 <= '1';
        DRFM1 <= '0';
        DRFM2 <= '1';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 2 ns; --124 ns
        DRFM0 <= '0';
        DRFM1 <= '1';
        DRFM2 <= '1';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 2 ns; --126 ns
        DRFM0 <= '1';
        DRFM1 <= '1';
        DRFM2 <= '1';
        DRFM3 <= '1';
        DRFM4 <= '0';
        wait for 2 ns; --128 ns
        DRFM0 <= '0';
        DRFM1 <= '0';
        DRFM2 <= '0';
        DRFM3 <= '0';
        DRFM4 <= '1';
        wait for 2 ns; --130 ns
        DRFM0 <= '1';
        DRFM1 <= '0';
        DRFM2 <= '0';
        DRFM3 <= '0';
        DRFM4 <= '1';
        wait for 2 ns; --132 ns
        DRFM0 <= '0';
        DRFM1 <= '1';
        DRFM2 <= '0';
        DRFM3 <= '0';
        DRFM4 <= '1';
        wait for 2 ns; --134 ns
        DRFM0 <= '1';
        DRFM1 <= '1';
        DRFM2 <= '0';
        DRFM3 <= '0';
        DRFM4 <= '1';
        wait for 2 ns; --136 ns
        DRFM0 <= '0';
        DRFM1 <= '0';
        DRFM2 <= '1';
        DRFM3 <= '0';
        DRFM4 <= '1';
        wait for 2 ns; --138 ns
        DRFM0 <= '1';
        DRFM1 <= '0';
        DRFM2 <= '1';
        DRFM3 <= '0';
        DRFM4 <= '1';
        wait for 2 ns; --140 ns
        DRFM0 <= '1';
        DRFM1 <= '1';
```

```
DRFM2 <= '1';
DRFM3 <= '0';
DRFM4 <= '1';
wait for 2 ns; --142 ns
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '1';
DRFM4 <= '1';
wait for 2 ns; --144 ns
DRFM0 <= '1';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '1';
DRFM4 <= '1';
wait for 2 ns; --146 ns
DRFM0 <= '0';
DRFM1 <= '1';
DRFM2 <= '0';
DRFM3 <= '1';
DRFM4 <= '1';
wait for 2 ns; --148 ns
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '1';
DRFM3 <= '1';
DRFM4 <= '1';
wait for 2 ns; --150 ns
DRFM0 <= '1';
DRFM1 <= '0';
DRFM2 <= '1';
DRFM3 <= '1';
DRFM4 <= '1';
wait for 2 ns; --152 ns
DRFM0 <= '0';
DRFM1 <= '1';
DRFM2 <= '1';
DRFM3 <= '1';
DRFM4 <= '1';
wait for 2 ns; --154 ns
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '0';
wait for 2 ns; --156 ns
DRFM0 <= '1';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '0';
wait for 2 ns; --158 ns
DRFM0 <= '1';
DRFM1 <= '1';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '0';
wait for 2 ns; --160 ns
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '1';
DRFM3 <= '0';
DRFM4 <= '0';
wait for 2 ns; --162 ns
DRFM0 <= '0';
DRFM1 <= '1';
DRFM2 <= '1';
DRFM3 <= '0';
DRFM4 <= '0';
wait for 2 ns; --164 ns
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '1';
DRFM4 <= '0';
wait for 2 ns; --166 ns
DRFM0 <= '1';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '1';
DRFM4 <= '0';
wait for 2 ns; --168 ns
DRFM0 <= '1';
DRFM1 <= '1';
DRFM2 <= '0';
DRFM3 <= '1';
DRFM4 <= '0';
wait for 2 ns; --170 ns
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '1';
DRFM3 <= '1';
DRFM4 <= '0';
wait for 2 ns; --172 ns
DRFM0 <= '0';
DRFM1 <= '1';
DRFM2 <= '1';
DRFM3 <= '1';
DRFM4 <= '0';
wait for 2 ns; --174 ns
DRFM0 <= '0';
DRFM1 <= '0';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '1';
wait for 2 ns; --176 ns
DRFM0 <= '0';
DRFM1 <= '1';
DRFM2 <= '0';
DRFM3 <= '0';
DRFM4 <= '1';
wait for 2 ns; --178 ns
```

```
		DRFM0 <= '1';
		DRFM1 <= '1';
		DRFM2 <= '0';
		DRFM3 <= '0';
		DRFM4 <= '1';
		wait for 2 ns; --180 ns
		PSV <= '0';
	   wait for 66 ns; --244 ns
		END_SIM <= TRUE;
	--	end of stimulus events
		wait;
	end process; -- end of stimulus process

	CLOCK_CLK : process
	begin
		--this process was generated based on formula: 0 0 ns, 1 1 ns -r 2 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
			CLK <= '0';
			wait for 1 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
			CLK <= '1';
			wait for 1 ns; --1 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_84_inSelect0 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 2 ns -r 4 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_84_inSelect0 <= '1';
			wait for 2 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_84_inSelect0 <= '0';
			wait for 2 ns; --2 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_84_inSelect1 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 4 ns -r 8 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_84_inSelect1 <= '1';
			wait for 4 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_84_inSelect1 <= '0';
			wait for 4 ns; --4 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_84_inSelect2 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 8 ns -r 16 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_84_inSelect2 <= '1';
			wait for 8 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_84_inSelect2 <= '0';
			wait for 8 ns; --8 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_83_inSelect0 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 2 ns -r 4 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_83_inSelect0 <= '1';
			wait for 2 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_83_inSelect0 <= '0';
			wait for 2 ns; --2 ns
		else
			wait;
		end if;
```

```
	end process;

	CLOCK_RB_83_inSelect1 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 4 ns -r 8 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_83_inSelect1 <= '1';
			wait for 4 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_83_inSelect1 <= '0';
			wait for 4 ns; --4 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_83_inSelect2 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 8 ns -r 16 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_83_inSelect2 <= '1';
			wait for 8 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_83_inSelect2 <= '0';
			wait for 8 ns; --8 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_82_inSelect0 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 2 ns -r 4 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_82_inSelect0 <= '1';
			wait for 2 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_82_inSelect0 <= '0';
			wait for 2 ns; --2 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_82_inSelect1 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 4 ns -r 8 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_82_inSelect1 <= '1';
			wait for 4 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_82_inSelect1 <= '0';
			wait for 4 ns; --4 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_82_inSelect2 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 8 ns -r 16 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_82_inSelect2 <= '1';
			wait for 8 ns; --0 fs
		else
			wait;
		end if;
		if END_SIM = FALSE then
		RB_82_inSelect2 <= '0';
			wait for 8 ns; --8 ns
		else
			wait;
		end if;
	end process;

	CLOCK_RB_81_inSelect0 : process
	begin
		--this process was generated based on formula: 1 0 ns, 0 2 ns -r 4 ns
		--wait for <time to next event>; -- <current time>
		if END_SIM = FALSE then
		RB_81_inSelect0 <= '1';
			wait for 2 ns; --0 fs
		else
```

```
                wait;
            end if;
            if END_SIM = FALSE then
            RB_81_inSelect0 <= '0';
                wait for 2 ns; --2 ns
            else
                wait;
            end if;
        end process;

        CLOCK_RB_81_inSelect1 : process
        begin
            --this process was generated based on formula: 1 0 ns, 0 4 ns -r 8 ns
            --wait for <time to next event>; -- <current time>
            if END_SIM = FALSE then
            RB_81_inSelect1 <= '1';
                wait for 4 ns; --0 fs
            else
                wait;
            end if;
            if END_SIM = FALSE then
            RB_81_inSelect1 <= '0';
                wait for 4 ns; --4 ns
            else
                wait;
            end if;
        end process;

        CLOCK_RB_81_inSelect2 : process
        begin
            --this process was generated based on formula: 1 0 ns, 0 8 ns -r 16 ns
            --wait for <time to next event>; -- <current time>
            if END_SIM = FALSE then
            RB_81_inSelect2 <= '1';
                wait for 8 ns; --0 fs
            else
                wait;
            end if;
            if END_SIM = FALSE then
            RB_81_inSelect2 <= '0';
                wait for 8 ns; --8 ns
            else
                wait;
            end if;
        end process;




            -- Add your stimulus here ...

        end TB_ARCHITECTURE;

        configuration TESTBENCH_FOR_hb_32rbps of hb_32rbps_tb is
            for TB_ARCHITECTURE
                for UUT : hb_32rbps
                    use entity work.hb_32rbps(structural);
                end for;
            end for;
        end TESTBENCH_FOR_hb_32rbps;
```

C. EXECUTING MACRO FOR THE 32 RANGE BIN TEST BENCH

```
SetActiveLib -work
comp -include "$DSN\src\hb_32rbps.vhd"
comp -include "$DSN\src\TestBench\hb_32rbps_TB.vhd"
asim TESTBENCH_FOR_hb_32rbps
wave
wave -noreg CLK
wave -noreg DRFM0
wave -noreg DRFM1
wave -noreg DRFM2
wave -noreg DRFM3
wave -noreg DRFM4
wave -noreg ENABLE_1
wave -noreg ENABLE_2
wave -noreg ENABLE_3
wave -noreg ENABLE_4
wave -noreg Gain0
wave -noreg Gain1
wave -noreg Gain2
wave -noreg Gain3
wave -noreg Inc0
wave -noreg Inc1
wave -noreg Inc2
wave -noreg Inc3
wave -noreg Inc4
wave -noreg InPadI0
wave -noreg InPadI1
wave -noreg InPadI2
wave -noreg InPadI3
wave -noreg InPadI4
wave -noreg InPadI5
wave -noreg InPadI6
wave -noreg InPadI7
wave -noreg InPadI8
wave -noreg InPadI9
wave -noreg InPadI10
wave -noreg InPadI11
wave -noreg InPadI12
wave -noreg InPadI13
wave -noreg InPadI14
wave -noreg InPadI15
wave -noreg InPadIOV
wave -noreg InPadQ0
wave -noreg InPadQ1
wave -noreg InPadQ2
wave -noreg InPadQ3
wave -noreg InPadQ4
wave -noreg InPadQ5
wave -noreg InPadQ6
wave -noreg InPadQ7
wave -noreg InPadQ8
wave -noreg InPadQ9
wave -noreg InPadQ10
wave -noreg InPadQ11
wave -noreg InPadQ12
wave -noreg InPadQ13
wave -noreg InPadQ14
wave -noreg InPadQ15
wave -noreg InPadQOV
wave -noreg {\InPad~I0\}
wave -noreg {\InPad~I1\}
wave -noreg {\InPad~I2\}
wave -noreg {\InPad~I3\}
wave -noreg {\InPad~I4\}
wave -noreg {\InPad~I5\}
wave -noreg {\InPad~I6\}
wave -noreg {\InPad~I7\}
wave -noreg {\InPad~I8\}
wave -noreg {\InPad~I9\}
wave -noreg {\InPad~I10\}
wave -noreg {\InPad~I11\}
wave -noreg {\InPad~I12\}
wave -noreg {\InPad~I13\}
wave -noreg {\InPad~I14\}
wave -noreg {\InPad~I15\}
wave -noreg {\InPad~Q0\}
wave -noreg {\InPad~Q1\}
wave -noreg {\InPad~Q2\}
wave -noreg {\InPad~Q3\}
wave -noreg {\InPad~Q4\}
wave -noreg {\InPad~Q5\}
wave -noreg {\InPad~Q6\}
wave -noreg {\InPad~Q7\}
wave -noreg {\InPad~Q8\}
wave -noreg {\InPad~Q9\}
wave -noreg {\InPad~Q10\}
wave -noreg {\InPad~Q11\}
wave -noreg {\InPad~Q12\}
wave -noreg {\InPad~Q13\}
wave -noreg {\InPad~Q14\}
wave -noreg {\InPad~Q15\}
wave -noreg ODVin
wave -noreg ODVout
wave -noreg Oper
wave -noreg OutPadIS0
wave -noreg OutPadIS1
wave -noreg OutPadIS2
wave -noreg OutPadIS3
wave -noreg OutPadIS4
wave -noreg OutPadIS5
wave -noreg OutPadIS6
wave -noreg OutPadIS7
wave -noreg OutPadIS8
wave -noreg OutPadIS9
wave -noreg OutPadIS10
wave -noreg OutPadIS11
wave -noreg OutPadIS12
```

```
wave -noreg OutPadIS13
wave -noreg OutPadIS14
wave -noreg OutPadIS15
wave -noreg OutPadISOV
wave -noreg OutPadQS0
wave -noreg OutPadQS1
wave -noreg OutPadQS2
wave -noreg OutPadQS3
wave -noreg OutPadQS4
wave -noreg OutPadQS5
wave -noreg OutPadQS6
wave -noreg OutPadQS7
wave -noreg OutPadQS8
wave -noreg OutPadQS9
wave -noreg OutPadQS10
wave -noreg OutPadQS11
wave -noreg OutPadQS12
wave -noreg OutPadQS13
wave -noreg OutPadQS14
wave -noreg OutPadQS15
wave -noreg OutPadQSOV
wave -noreg {\OutPad~IS0\}
wave -noreg {\OutPad~IS1\}
wave -noreg {\OutPad~IS2\}
wave -noreg {\OutPad~IS3\}
wave -noreg {\OutPad~IS4\}
wave -noreg {\OutPad~IS5\}
wave -noreg {\OutPad~IS6\}
wave -noreg {\OutPad~IS7\}
wave -noreg {\OutPad~IS8\}
wave -noreg {\OutPad~IS9\}
wave -noreg {\OutPad~IS10\}
wave -noreg {\OutPad~IS11\}
wave -noreg {\OutPad~IS12\}
wave -noreg {\OutPad~IS13\}
wave -noreg {\OutPad~IS14\}
wave -noreg {\OutPad~IS15\}
wave -noreg {\OutPad~QS0\}
wave -noreg {\OutPad~QS1\}
wave -noreg {\OutPad~QS2\}
wave -noreg {\OutPad~QS3\}
wave -noreg {\OutPad~QS4\}
wave -noreg {\OutPad~QS5\}
wave -noreg {\OutPad~QS6\}
wave -noreg {\OutPad~QS7\}
wave -noreg {\OutPad~QS8\}
wave -noreg {\OutPad~QS9\}
wave -noreg {\OutPad~QS10\}
wave -noreg {\OutPad~QS11\}
wave -noreg {\OutPad~QS12\}
wave -noreg {\OutPad~QS13\}
wave -noreg {\OutPad~QS14\}
wave -noreg {\OutPad~QS15\}
wave -noreg PSV
wave -noreg RB_81_inSelect0
wave -noreg RB_81_inSelect1
wave -noreg RB_81_inSelect2
wave -noreg RB_82_inSelect0
wave -noreg RB_82_inSelect1
wave -noreg RB_82_inSelect2
wave -noreg RB_83_inSelect0
wave -noreg RB_83_inSelect1
wave -noreg RB_83_inSelect2
wave -noreg RB_84_inSelect0
wave -noreg RB_84_inSelect1
wave -noreg RB_84_inSelect2
wave -noreg UNP
wave -noreg URB
run 200.00 ns
# The following lines can be used for timing simulation
# acom <backannotated_vhdl_file_name>
# comp -include "$DSN\src\TestBench\hb_32rbps_TB_tim_cfg.vhd"
# asim TIMING_FOR_hb_32rbps
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Ekestorm, S. R. T. and Karow, C., *An All-Digital Image Synthesizer For Countering High-Resolution Imaging Radars,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2000.

[2] Guillaume, C. H., *Circuit Design and Simulation For A Digital Image Synthesizer Range Bin Modulator,* Master's Thesis, Naval Postgraduate School, Monterey, California, March 2002.

[3] Borrione, D. D., Pierre, L. V., Salem, A. M., *Formal Verification of VHDL Descriptions in Prevail Environment*, IEEE Design and Test of Computers , Vol. 9, Issue 2, pp. 42–56, June 1992.

[4] Hua, G. X.; Zhang, H., *Formal Semantics of VHDL for Verification of Circuit Designs*, Computer Design: VLSI in Computers and Processors, 1993. ICCD '93. Proceedings, 1993 IEEE International Conference, pp. 446 –449, 1993.

[5] Bergeron, J., *Writing Testbenches: Functional Verification of HDL Model*, Kluver Academic Publishers, Norwell Massachusetts USA, Sixth Printing, 2002.

[6] York, G., Mueller-Thuns, R., Patel, J., Beatty, D., *An Integrated Environment for HDL Verification*, Verilog HDL Conference, 1995. Proceedings, 1995 IEEE International, pp. 9–18, 1995.

[7] Borman, J., Lohse, J., Payer, M., Venzl, G., *Model Checking in Industrial Hardware Design*, ACM/IEEE Design Automation Conference, 1995.

[8] Deharbe, D., Shankar, S., Clarke, E. M., *Formal Verification of VHDL: The Model Checker CV*, Integrated Circuit Design, 1998. Proceedings. XI Brazilian Symposium, pp. 95 –98, 1998.

[9] Beer, I., Ben-David, S., Eisner, C., Landver, A., *RuleBase: An Industry-Oriented Formal Verification Tool*, Proceedings, Design Automation Conference, 1996.

[10] Borrione, D. D., Pierre, L. V., Salem, A. M., *Formal Verification of VHDL Descriptions in Prevail Environment*, IEEE Design & Test of Computers, Vol. 9, Issue 2, pp. 42 –56, June 1992.

[11] Binder, R. V., *Testing Object-Oriented Systems,* Addison-Wesley, Third Printing June 2001.

[12] Fouts, D. J., Pace, P. E., Karow, C., Ekestorm, S. R. T., *A Single-Chip False Target Radar Image Generator for Countering Wideband Imaging Radars,* IEEE Journal of Solid-State Circuits, Vol. 37, No. 6, June 2002.

[13] LeDantec, F., *Performance Analysis of a Digital Image Synthesizer as a Counter-Measure Against Inverse Synthetic Aperture Radars,,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 2002.

[14] Yalamanchili, S., *Introductory VHDL From Simulation to Synthesis,* Prentice-Hall, Upper Saddle River, New Jersey, 2001

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Douglas Fouts
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

4. Professor Phillip Pace
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor Man-Tak Shing
   Department of Software Engineering
   Naval Postgraduate School
   Monterey, California

6. Professor John Powers
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

7. LtCol Pete Boerlage
   Naval Postgraduate School
   Monterey, California

8. Dr. John A. Montgomery
   Naval Research Laboratory
   Washington, D.C.

9. Mr. Alfred A. Di Mattesa
   Naval Research Laboratory
   Washington, D.C.

10. Mr. Gregory P. Hrin
    Naval Research Laboratory
    Washington, D.C.

11. Mr. Daniel W. Bay
Naval Research Laboratory
Washington, D.C.

12. Dr. Frank Klemm
Naval Research Laboratory
Washington, D.C.

13. Mr. Brian W. Edwards
Naval Research Laboratory
Washington, D.C.

14. Mr. George D. Farmer
Naval Research Laboratory
Washington, D.C.

15. Dr. Preston W. Grounds
Naval Research Laboratory
Washington, D.C.

16. Dr. Peter Craig
Office of Naval Research
Arlington, Virginia

17. Dr Joseph Lawrence
Office of Naval Research
Arlington, Virginia

18. Mr. James Talley
Office of Naval Research
Arlington, Virginia

19. Swedish Armed Forces Headquarters
HKV/KRI LED
Stockholm, Sweden

20. Swedish National Defence College
MTI
Stockholm, Sweden

21. Swedish Defence Materiel Administration
Stockholm, Sweden

22. Swedish Defence Research Agency
Linkoping, Sweden