AFRL-SN-RS-TR-2006-292
Final Technical Report
September 2006

SYSTEM ON A CHIP REAL-TIME EMULATION (SOCRE)

University of California at Berkeley

STINFO FINAL REPORT

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* SEP 2006	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* Sep 05 – Apr 06

4. TITLE AND SUBTITLE
SYSTEM ON A CHIP REAL-TIME EMULATION (SOCRE)

5a. CONTRACT NUMBER FA8750-05-1-0275

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER 63739E

6. AUTHOR(S)
John Wawrzynek, Robert Brodersen, Brian Richards

5d. PROJECT NUMBER N584

5e. TASK NUMBER 15

5f. WORK UNIT NUMBER 01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Regents of the University of California, Berkeley
336 Sproul Hall
Berkeley California 94720-5940

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/SNRT
26 Electronic Parkway
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-SN-RS-TR-2006-292

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA #06-651

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The SOCRE program focused on developing a simulation engine for real-time emulation of large, complex mixed signal IC designs. The methodology utilizes the BEE2 reconfigurable computing platform with a compute capability at or near teraop/sec performance levels, allowing it to perform real-time evaluations of architectures and algorithms as well as real-time verification of the system performance. A silicon implementation was then synthesized using an automatic flow, avoiding re-entry of the design description. The design system has the ability to emulate chip performance at full-rate and in-system, allowing easy exploration of higher performance, lower power design options, and the automated chip generation insures rapid SOC IC implementation from a common design description.

15. SUBJECT TERMS
ASIC Emulation, CAD Design, Mixed Signal, System-on-a-chip

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Carl R. Thomas
U	U	U	UL	17	**19b. TELEPHONE NUMBER (*Include area code)***

Table of Contents

Preface 1

1 Introduction 1

2 Methods, Assumptions, and Procedures 1
Specification of Demonstration Designs 1
XMAC Multiple Antenna Correlator 2
Interactive Demonstration Design: Video Edge Detection 3

3 Results and Discussion 5
Migration and Enhancement of the existing BEE Design Flow 5
BEE2 Platform Development and Testing 5
Port BEE flow to Emulation Platforms, SOC Technologies 5
Mixed signal design flow extensions 5
Microprocessor Integration into the Design Flow 6
Insecta Simulink to Silicon flow 7

4 Demonstration Designs Mapped to BEE2 and Silicon 9
XMAC Multiple Antenna Correlator 9
Top-Down vs Bottom-Up Design 10
Mapping the Video Edge Detection Design to Silicon 11

5 Conclusions 12

6 Recommendations 12

7 References 13

List of Figures

Figure 1: Multiple antenna correlator architecture. 2
Figure 2: The architecture of the XMAC subsystem. 3
Figure 3: Video Edge Detection demonstration. 4
Figure 4: Microprocessor, mixed signal, and I/O abstractions in the BEE2 Simulink libraries. 7
Figure 5: The Insecta main user interface. 8
Figure 6: The 16 antenna correlator based on the BEE2. 10
Figure 7: The initial GDSII layout of the XMAC subsystem for the 256 antenna correlator design. 11

Preface

Military sensor systems require real-time processing of data which can only be achieved using a dedicated, custom IC design. Unfortunately, the design of complex, mixed signal ASICs can be cost prohibitive and the extended design times do not allow for iterative development or changing operational requirements. The SOCRE program demonstrates a platform for determining mission capability of complex ICs for use in processing real-time sensor input.

1 Introduction

The goal of the SOCRE seedling program was to develop a simulation engine for real time emulation of large, complex mixed signal IC designs. Leveraging a compute capability at or near teraop/sec performance levels, it allows real-time evaluations of architectures and algorithms in silicon and would allow real-time verification of the system performance. An automated flow was developed to generate a silicon implementation, using the same design description used by the emulation environment. With the ability to emulate chip performance at full-rate and in-system, higher performance, lower power design options can be explored. This lead to the development of a high-level Algorithm to Architecture selection flow leveraging the early power/area/speed estimates.

2 Methods, Assumptions, and Procedures

The Berkeley Emulation Engine (BEE) is a real-time hardware emulation engine which has a complete software environment for programming and debugging. The purpose of BEE is to provide a rapid prototyping method to facilitate and accelerate chip design. With BEE, complex algorithms can be automatically synthesized and tested on real hardware with external mixed signal components for a full-scale real-time functionality check and block-level timing verification. With the integrated automatic Simulink-to-Implementation design flow, the user can easily implement their design in both BEE system and silicon, with cycle-to-cycle and bit-true equivalence.

The BEE2 architecture was developed, using Xilinx Virtex2 Pro FPGA.

Specification of Demonstration Designs

A key element of the SOCRE research project was to identify a demonstration system with high system performance requirements, which would benefit from the proposed real-time emulation environment. To this end, a radio-astronomy application to correlate between a large number of antennas was identified, which utilizes a high-performance Cross-bar-based Multiple Antenna Correlator (XMAC). In addition, a smaller design performing real-time edge detection on real-time video data was selected as a tutorial example suitable for live demonstrations of the emulation environment and the SOC flow.

XMAC Multiple Antenna Correlator

The XMAC design driver is a high-performance DSP subsystem for processing the high volumes of data from large arrays of radio antennas. By using several smaller antennas, astronomers are able to create a large virtual antenna, or survey activity in the sky in multiple directions simultaneously. This design has many characteristics common to military sensor applications using techniques such as correlation or beam steering. The correlator is implemented by analyzing the spectrum of each antenna in real-time, with an FFT processor dedicated to each antenna. A crossbar then collects the coefficients from each frequency bin into streams of packets, where each stream is dedicated to a specific frequency bin. Each stream of coefficients is fed to a dedicated XMAC block, one for each frequency bin. The XMAC then receives a packet of several time samples from the selected frequency bin from Antenna 1, followed by samples from Antenna 2, and so on. Overall, the number of XMAC processors equals the number of frequency bins of interest, *k*.

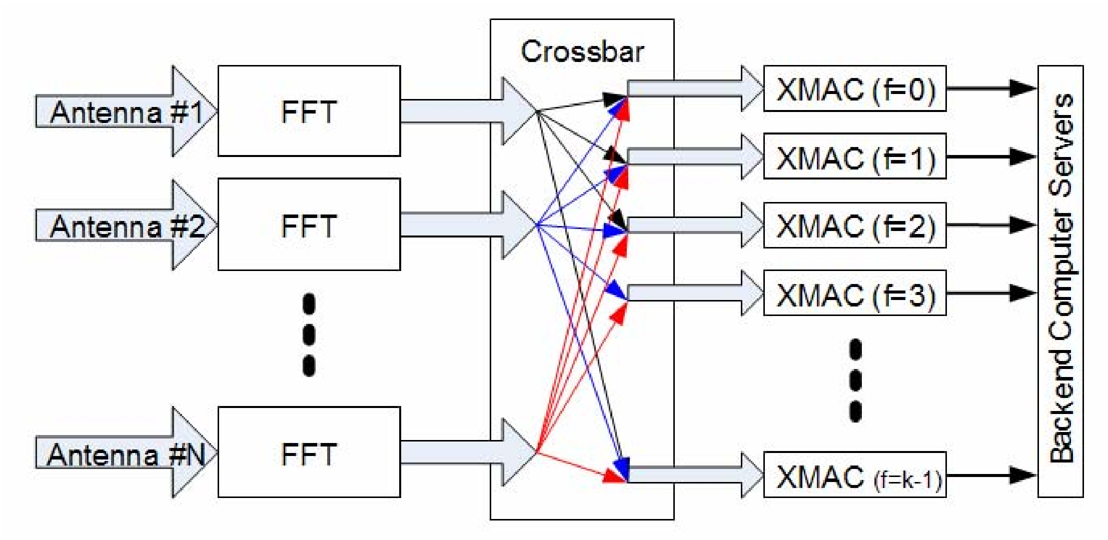


Figure 1: Multiple antenna correlator architecture: One FFT per antenna, One XMAC per frequency bin, and a Crossbar to separate the coefficients.

The XMAC architecture consists of a regular array of complex dual-polarity complex MAC units, as shown in Figure 2. The first CMAC calculates the autocorrelation of coefficients from a given antenna, the second CMAC correlates between adjacent antennas, and so on. By delaying the stream of packets by one packet length between each stage, consecutively more separated antenna data packets are processed, and ultimately correlations between all pairs of antennas are calculated. The number of CMAC units grows with the number of antennas, *N*. For a given output bandwidth, the total memory capacity needed to implement the delay lines increases as N^2, where both the size of the memories and number of memories increase as *N*.

The XMAC implemented was designed to support up to 256 antennas, using 128 delay lines and 129 dual-polarity CMAC blocks. The delay lines were implemented using compiled memory, each with 16Kbits organized as 64 bits by 256 rows. For the SOC implementation, the CMAC was implemented using RTL mapped to standard cells.

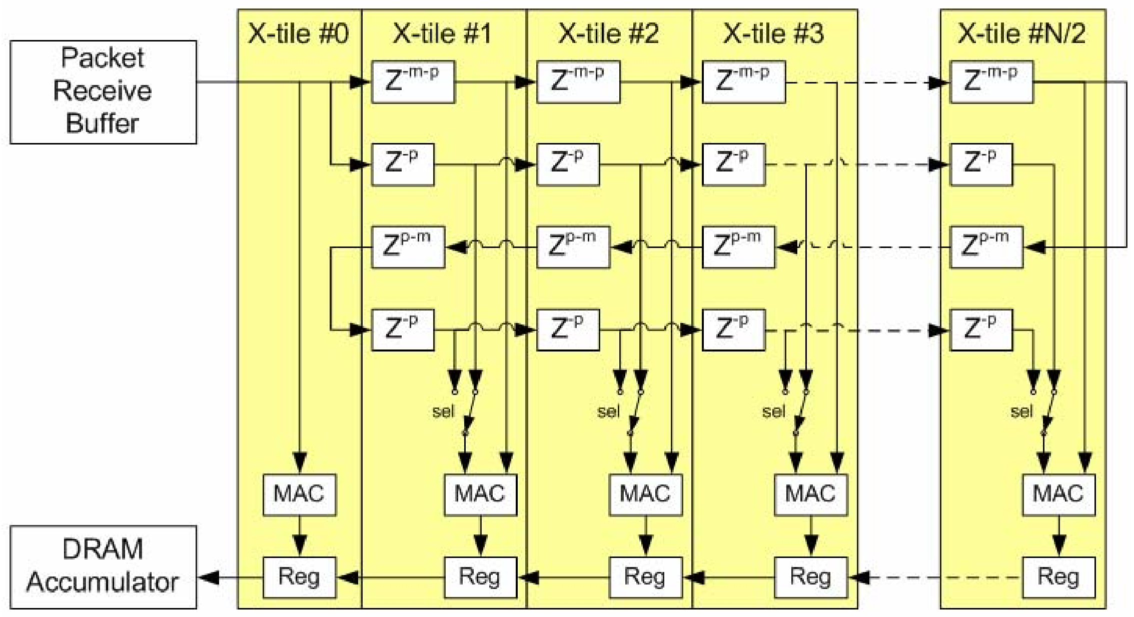


Figure 2: The architecture of the XMAC subsystem, showing the regular slices containing the delay lines and dual polarity complex MAC blocks.

The highly regular nature of the XMAC subsystem lends itself to a hierarchical design approach, and as a design driver, exercises bottom-up hierarchical design approaches to several phases of the design process, including design entry, automated retargeting to SOC HDL, foundry-specific logic syntheses, and place and route CAD tools.

Interactive Demonstration Design: Video Edge Detection

In addition to the complex XMAC design, a smaller image processing subsystem was developed as a tutorial to demonstrate the block-diagram based design flow. The design for a Sobel edge detection algorithm was described in Simulink, using Xilinx System Generator library components.

The edge detection emulation system takes advantage of a video test bench environment developed for the BEE2, shown in Figure 3, providing a source of real-time video data from a Gigabit Ethernet port to feed the filter design under test. The results from the filter can be viewed on an LCD video monitor via an HDMI video connector on the BEE2. In addition to processing the data in real time on the BEE2, the user can login to Linux running on the control processor, and remotely control simulation parameters, allowing the designer to interactively observe the results.

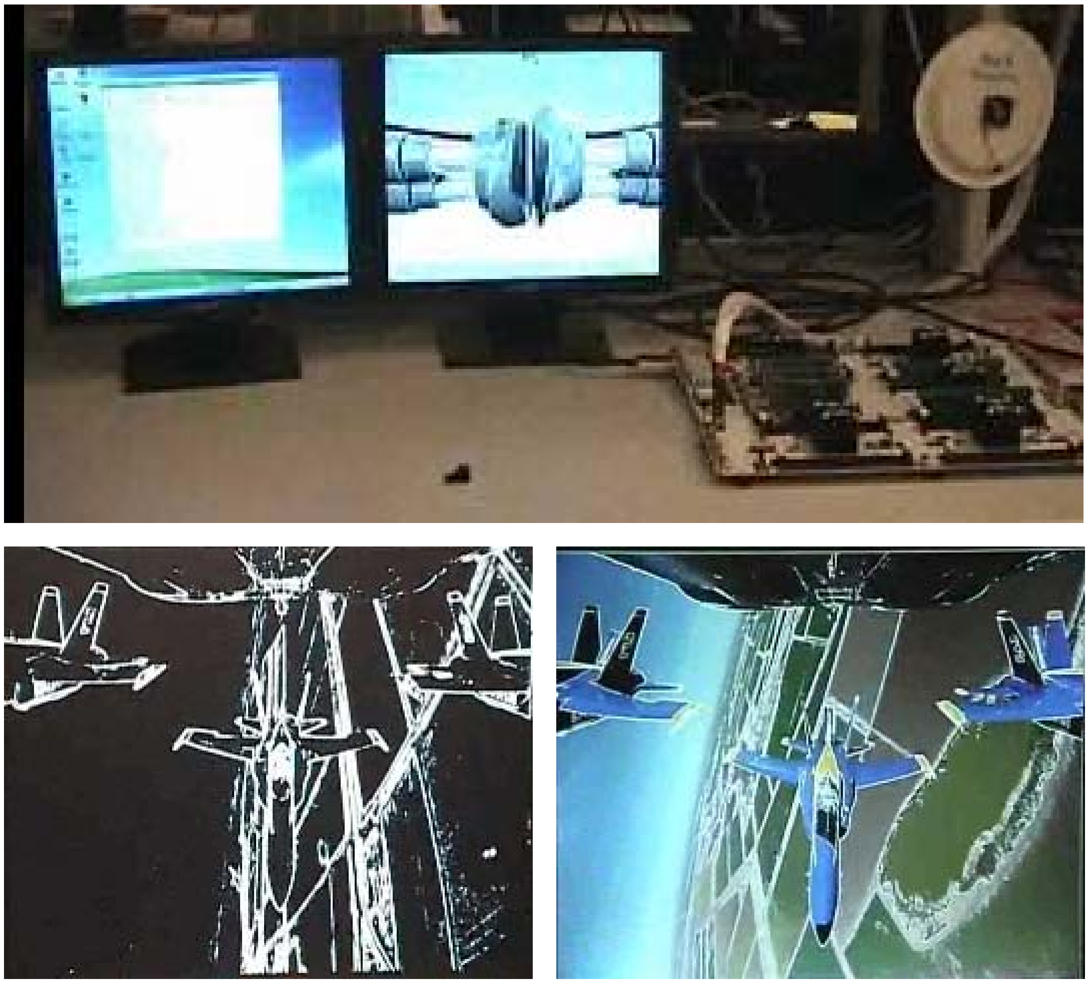


Figure 3: Video Edge Detection demonstration, showing the BEE2 console and filtered video on two LCD displays, with two examples of edge detection results, based on interactive user configuration from the console.

3 Results and Discussion

Migration and Enhancement of the existing BEE Design Flow

BEE2 Platform Development and Testing

As the SOCRE project began in Fall 2005, the initial versions of the BEE2 platform were assembled and tested, and applications including the BEE design flow and related support were ported to the BEE2. Testing suites for the BEE2 emulation platform were developed to verify the hardware integrity and performance. Early porting of applications to the BEE2 emulation platform included LDPC decoders, A/V and radio applications

Port BEE flow to Emulation Platforms, SOC Technologies

One of the key tasks of the SOCRE program was to port the BEE design flow to the BEE2 FPGA platform. To make the most of this effort, the design flow was made extensible, and rather than being limited to a single platform like the original BEE ISE design flow, the new version can select between one of several target FPGA platforms. Similarly, the Insecta flow for mapping the design to an SOC has been extended to be more portable, and includes the ability to map a design to 90nm and 130nm process technologies. The flow can be customized for each target technology, with the goal of using the foundry-provided design flows where possible.

The original BEE flow focused on mapping data path architectures to both the BEE platform and SOC layout. The BEE2 platform, however, offered new capabilities including the ability to connect to high-speed I/O devices, and a total of 10 embedded PowerPC™ processors. To make the most of these resources, the BEE_XPS environment was developed, providing abstractions for the new capabilities. Figure 4 shows a Simulink design containing A/D devices and microprocessor I/O ports combined with a datapath architecture.

Mixed signal design flow extensions

The BEE2 processor itself does not have direct analog I/O, but is able to interface with external interface boards including the Infiniband Break-Out Board (IBOB). This board is programmed in the same fashion as the BEE2, but a different target design is selected. The IBOB can be connected to both A/D and D/A interfaces, including a 1.2 GS/s front-end board, allowing the designer to interface to radios for real-world communication algorithm and architecture testing.

The ability to interface with analog devices in the emulation environment suggests that the SOC design flow could be augmented to support the mapping to analog IP subsystems, for mixed signal SOC designs. This is similar to the problem of including foundry-compiled IP such as SRAM memory circuits into the design flow. To this end, Insecta was augmented so that custom IP could be added to the flow, and both the XMAC and edge detection design driver circuits exercise this capability.

Custom IP is integrated into the Simulink to Silicon flow by creating a VHDL wrapper for the custom IP that conforms to Simulink I/O conventions. For the test designs, a single-port memory block is needed, which is mapped to the FPGA platforms by using a Xilinx-provided SPRAM (Single-Port RAM) block.

To map the same design to an SOC, the SPRAM can be replaced by a “Black Box” built by referencing the VHDL wrapper file. By making the ports and behavior of the wrapper match the ports and behavior of the SPRAM, the designer need only substitute one part for another before generating the net list.

Microprocessor Integration into the Design Flow

In addition to supporting datapath and mixed signal designs, microprocessor support was added to the BEE2 design flow, to take advantage of the PowerPC and other processors supported within the FPGA environment. Leveraging the Xilinx Platform Studio (XPS) technology to describe microprocessor-based platforms in the emulation environment, a library of Simulink blocks was developed to provide abstractions for I/O, control registers and shared memory interfaces, some of which are shown in Figure 4. This approach allows the designer to develop high-performance datapath designs without considering the details of the software platform implementation. In many cases, the datapath can be simulated early on without the complete processor implementation. Once the design has been described within Simulink, the designer runs the BEE design flow within Matlab using the bee_xps interface. At this point, the designer selects the target platform, such as the BEE2 or IBOB processor, and the CPU. The design flow then maps the register and memory abstractions into platform-specific implementations, and ultimately produces FPGA configuration files, optionally uploading the complete subsystem to the selected platform for real-time emulation.

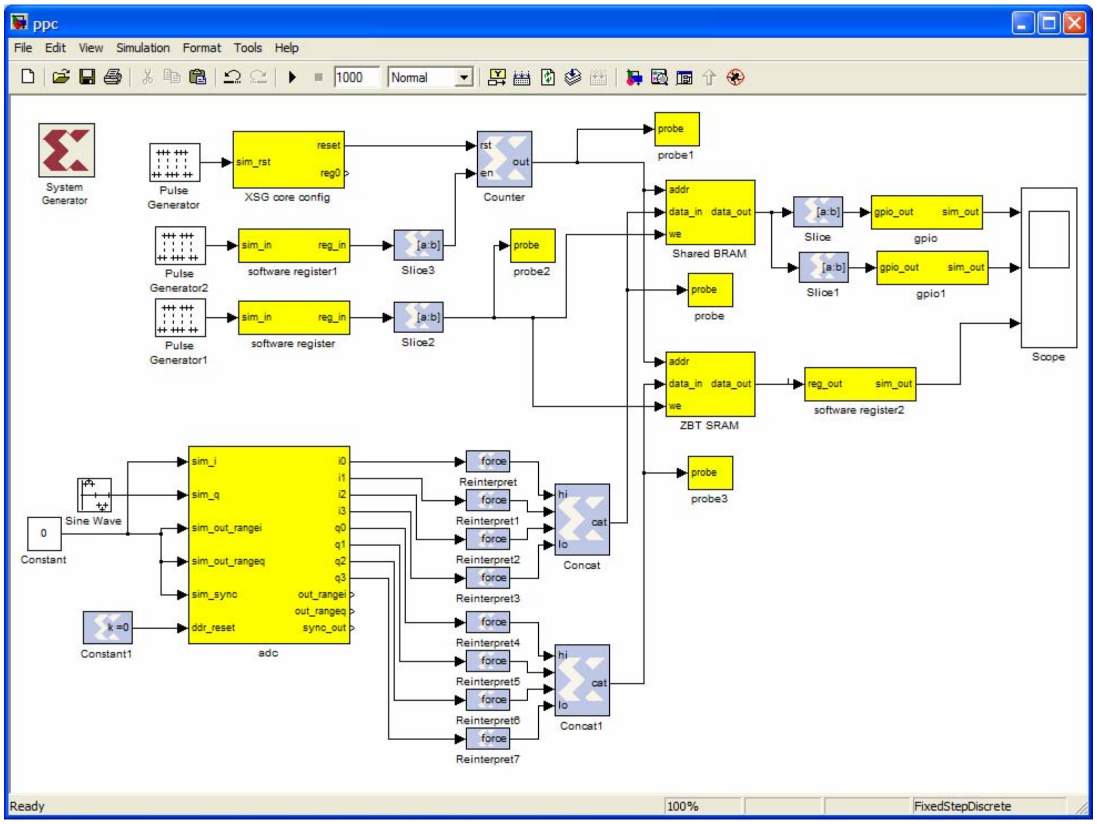


Figure 4: Microprocessor, mixed signal, and I/O abstractions in the BEE2 Simulink libraries, showing an A/D interface, microprocessor dual-port memories and registers, and debugging probes.

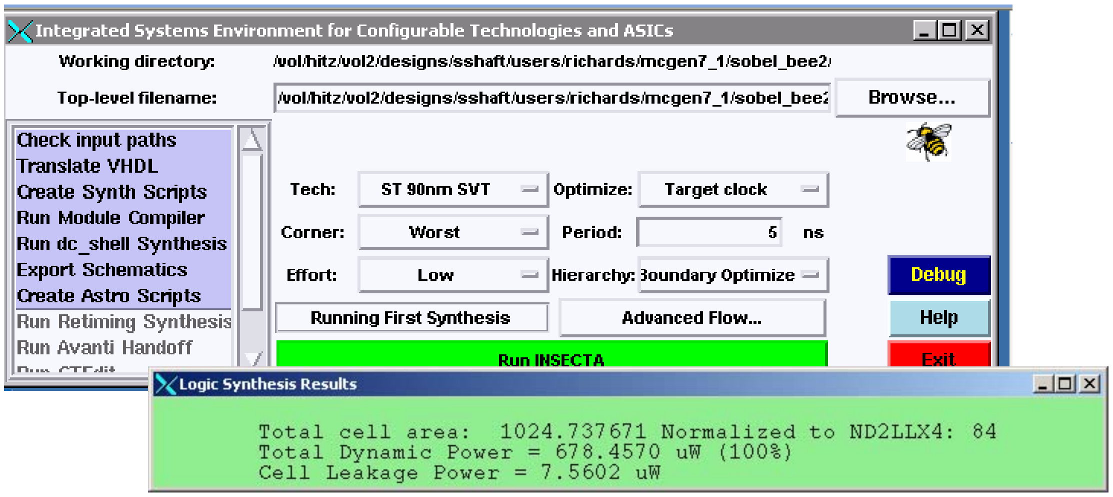
To take advantage of the registers and memory on the uploaded design, the user can login to Linux running on the control FPGA on the BEE2, and connect to the processor generated above using a simple command shell. This allows registers and memory to be read from and written to interactively, or the designer can run a custom C-based application to exercise the system under development. The BEE2 platform allows up to four designers to connect remotely, and manipulate designs on the four user-FPGAs on the BEE2.

Insecta Simulink to Silicon flow

Once an architecture has been described in Simulink, and successfully mapped to the BEE2 or similar platform, the same design can be mapped to a selected SOC technology using the Insecta Simulink to Silicon design flow, using the GUI shown in Figure 5. Early in the design process, Insecta can be run to generate estimates for power, area and speed, with minimal user intervention. Once the architecture has been selected, the Insecta flow can automate many of the final place and route steps to generate GDSII layout.

The Insecta design flow is configurable so that design flow steps can be modified or added for a given choice of technology. A key goal of the interface is to provide a turnkey interface into design flows provided by each target foundry, building the design workspace and configuration files needed by the commercial design flow for the given design described in Simulink. This workspace can then be used as a starting point for further customization of the foundry provided design flow.

Recent improvements include the ability to integrate foundry-provided IP blocks, such as high-density static memory, which is used in both of the design drivers described previously. This is critical for many memory intensive designs, offering at least a 15x improvement in area over flip-flop-based designs in 90nm CMOS, as formerly supported by the Insecta flow.



Working directory: /vol/hitz/vol2/designs/sshaft/users/richards/mcgen7_1/sobel_bee2/
Top-level filename: /vol/hitz/vol2/designs/sshaft/users/richards/mcgen7_1/sobel_bee2

Tech: ST 90nm SVT
Optimize: Target clock
Corner: Worst
Period: 5 ns
Effort: Low
Hierarchy: Boundary Optimize

```
Total cell area:  1024.737671 Normalized to ND2LLX4: 84
Total Dynamic Power = 678.4570 uW (100%)
Cell Leakage Power = 7.5602 uW
```

Figure 5: The Insecta main user interface showing user-selectable design steps, and in intermediate summary of power and area estimates from first synthesis.

The Insecta was pre-released to Rice University, where a test decoder design was successfully defined. Library support was enhanced through this collaboration to include the addition of new functions such as a barrel shifter into the flow, and to add a new foundry back-end.

4 Demonstration Designs Mapped to BEE2 and Silicon

XMAC Multiple Antenna Correlator

The XMAC subsystem of the multiple-antenna correlator described earlier was fully developed on the BEE2 platform, both to validate the algorithm and architecture, and to provide a complete solution for a 16 antenna array. The SOC mapping began early on, before the architecture was finalized, to provide power, area and performance estimates to guide the planning of a 256 antenna solution.

Once the initial simulations were running, the same Simulink block diagram was used to drive the SOC flow, with the exception of one library component for implementing the large delay elements used in the datapath. A key difference between the FPGA fabric and the SOC target was the availability and behavior of the high-density static memory blocks. On the FPGA fabric, 4Kbit block memory primitives called BRAMs are available, with several configuration options, including single port and dual port data access. In this case, a delay line is implemented by merely applying the output of a counter to the address lines of the BRAM, and accessing the memory with read-before-write behavior.

For the SOC solution, a single-port memory is used, to provide higher layout density on silicon. The memory was generated by the silicon foundry using a memory compiler, generating memory that does not support simultaneous read and write. The implementation of the delay line was completely different from the FPGA solution, interleaving read and write operations. Once the library component for the delay line was implemented, the block could be substituted for the BRAM-based equivalent.
To verify the equivalence of the two delay line implementations, both were instantiated in a test design side-by-side to confirm that they have the same behavior. These blocks could then be interchanged depending on whether the design was being targeted to an FPGA or an SOC. A future task could include the automation of this substitution to create a truly target-independent Simulink design.

In parallel with the development of the design flow extensions for targeting an SOC, the correlator design for a 16 antenna solution was fully assembled, using a BEE2 processor board, four IBOB boards, and eight A/D 1.2 GHz dual channel front-end boards, shown in Figure 6. The IBOB boards connect to the BEE2 using Infiniband connectors, one per board, and two A/D boards plug into each of the IBOB edge connectors.

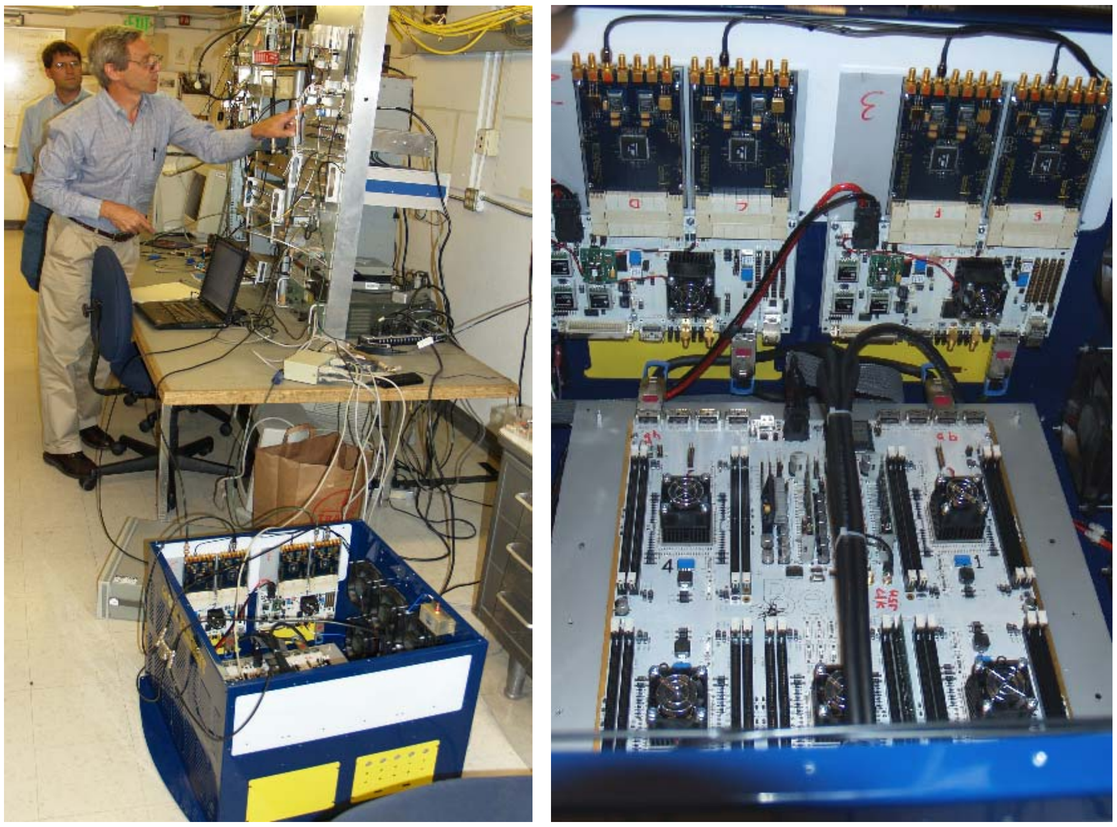


Figure 6: The 16 antenna correlator based on the BEE2 with 4 IBOB boards and 8 A/D cards, deployed in the U. C. Berkeley Undergraduate Radio Astronomy Lab.

Top-Down vs Bottom-Up Design

The XMAC prototype for 16 antennas was run through synthesis, place and route to produce GDSII layout in 90nm CMOS, running the design flow in a top-down fashion, with minimal designer interaction This approach is recommended, allowing the system designer to explore architectural tradeoffs without manual design partitioning.

The top-down tool flow, however, can fail during one or more design step for large designs. When the top-down design flow was tried on a 256 antenna version of the XMAC, the CAD tools and compute resources were inadequate, and the tools failed to map the design, running out of memory after 48 hours. In particular, the commercial HDL generation flow currently in use is intended to target state-of-the-art FPGAs, and is not designed for mapping larger designs to SOCs. To avoid this limitation, either a new netlisting approach must be developed, or a hierarchical bottom-up design style must be used to limit the complexity of subsystems that are run through the tool flow.

To accommodate the larger XMAC design, a bottom-up design entry and synthesis style was applied within the BEE and Insecta flow. Initially, a small 4-antenna version of the XMAC was compiled top-down in System Generator, and synthesized to target libraries using Insecta. The resulting design contained fully compiled and synthesized subsystems for key elements of the XMAC, including the SRAM based delay line (*sram_delay*) and the Dual Pole Complex MAC (*dual_pol_cmac*). These blocks were then inserted as pre-compiled VHDL subsystems, or *black boxes*, into a larger Simulink design, with many references to the same pre-synthesized *sram_delay* and *dual_pol_cmac* blocks. The resulting complete system was then compiled to VHDL in less than 30 minutes, after which a similar top-down synthesis approach was used to complete the mapping of the 256-antenna design to foundry library and memory primitives. The complete design was then run through Place and Route steps which took several hours to generate the GDSII layout shown in Figure 7.

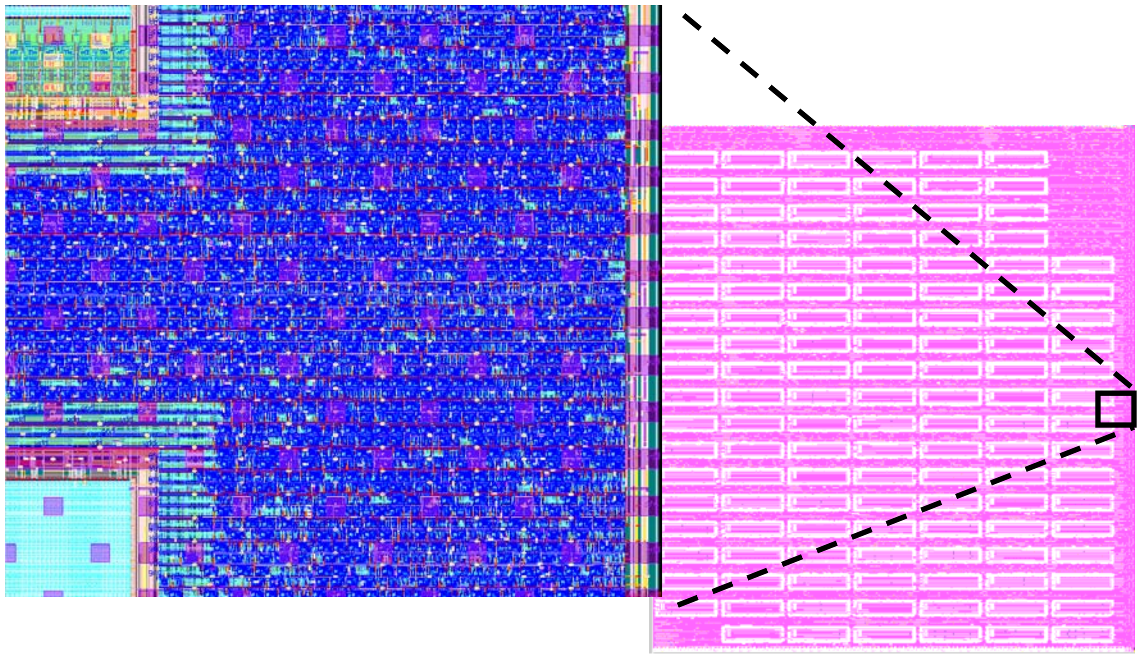


Figure 7: The initial GDSII layout of the XMAC subsystem for the 256 antenna correlator design, showing a detail of the IP memory and logic. The design is 3.7mm x 3.7mm.

Mapping the Video Edge Detection Design to Silicon

The video edge-detection demonstration architecture described earlier was run through the Insecta flow after successful mapping to the BEE2 platform. This design was small enough to run top-down through the flow, and demonstrated the newly added ability to include founder IP for memory into the design. As part of a demonstration at the DARPA PACE Workshop in April 2006 at U. C. Berkeley, this complete flow was interactively run through the Insecta SOC flow, running net list translation, Synthesis, Place and Route steps to produce and preview GDSII in less than 25 minutes.

5 Conclusions

The SOCRE research project successfully demonstrated that a single design description can be automatically mapped both to FPGA targets such as the BEE2 emulation platform, and to SOC libraries, with little or no redesign effort. The ability to retarget a design to foundry-optimized IP allows designs to be more area/speed/power efficient, in one case demonstrating a 15x area improvement for a memory-intensive design by leveraging SRAM library components. By using this approach, the designer can rapidly generate power/area/speed estimates to help guide with the architecture selection process for a given algorithm to SOC flow (Insecta). At the DARPA PACE Workshop hosted at the BWRC, this flow was interactively demonstrated, targeting both the BEE2 and a commercial silicon flow.

6 Recommendations

Several students at the BWRC have used the BEE and Insecta flow to design complex systems, and this approach was demonstrated as an effective way to explore design tradeoffs. This lead to the partitioning of high-level system design into two phases: Algorithm to Architecture selection, and Architecture to Silicon mapping. Insecta is used for both collecting estimates to guide the architecture selection process, and also generating layout for inclusion in an SOC or to generate more accurate estimates. Future work is suggested to automate the architecture selection process, including floating point to fix-point conversion, and energy optimization.

For large designs, the BEE2 and Insecta flows can help speed up design by emulating complex systems with cycle and bit accuracy at or near real time. There are several opportunities, however, for improving the performance of the tools as design complexity continues to grow. To enable top-down design mapping for larger designs, new netlisting software would need to be developed and tested with the current libraries. Alternatively, tools can be developed to automate the faster, but more complex bottom-up use of design flows..

Early experiments to incorporate IP and microprocessors into the design flow show promising areas of continued research. Analog IP has been integrated into the emulation design flow, and similar techniques to those used for including SRAM components into an SOC should apply to generating mixed signal designs with analog subsystems. Similarly, the same approach described earlier to include microprocessors into an FPGA emulation can be applied to including embedded microprocessors into an SOC.

A key observation at the DARPA PACE Workshop in April 2006, and from many research efforts collaborating with the BWRC, is that many groups would like to use the BEE2 or a similar platform for a variety of applications. A clear opportunity is presented to DARPA to help increase the availability of the BEE2 platforms to the research community.

7 References

For more information see the web site: *http://bee2.eecs.berkeley.edu*

Chen Chang, John Wawrzynek, Robert W. Brodersen. "Design & Application of BEE2 - a High-end Reconfigurable Computing System," HotChips 17, A Symposium on High Performance Chips, Aug 14-16, 2005.

C. Chang, J. Wawrzynek, R. W. Brodersen, "The Design and Application of a High-End Reconfigurable Computer System," Proceedings of the 2005 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA2005), pp. 129-136, June 2005.

C. Chang, J. Wawrzynek, and R. W. Brodersen. "BEE2: A High-End Reconfigurable Computing System," IEEE Design and Test of Computers, 22(2):114--125, Mar/Apr 2005.

C. Chang, J. Wawrzynek, R. W. Brodersen. "BEE2: A High-End Reconfigurable Computing System," *IEEE Design and Test of Computers*, vol. 22, no. 2, pp. 114-125, March/April 2005.

K. Kuusilinna, C. Chang, etc, "Real-time System-on-Chip Emulation," Chapter 10, Winning the SoC Revolution, p. 229-253, 2003.

C. Chang, K. Kuusilinna, B. Richards, A. Chen, N. Chan, R. W. Brodersen, B. Nikolić, "Rapid Design and Analysis of Communication Systems Using the BEE Hardware Emulation Environment," Proc. IEEE Rapid System Prototyping Workshop, June 2003.

C. Chang, K. Kuusilinna, B. Richards, and R.W. Brodersen, “Implementation of BEE: a Real-time Large-scale Hardware Emulation Engine,” Proc. FPGA 2003, pp. 91-99, Feb. 2003.

K. Kuusilinna, C. Chang, M. J. Ammer, B. Richards, and R. W. Brodersen, “Designing BEE: a Hardware Emulation Engine for Signal Processing in Low-Power Wireless Applications,” EURASIP Journal on Applied Signal Processing, special issue on Rapid Prototyping of DSP Systems, 2003. (pdf)