Power Efficient Radar Signal Processor

Yassir Salama, ITT AES,
Dennis Fitzgerald, ITT AES,
Gerald Bright CSC,
John Rooks, AFRL

Key Words: Efficient, Radar, Signal Processor, WSSP, VHDL, DPMI, STAP

ABSTRACT

Power efficiency and ease of programming are key issues in the implementation of real-time systems in airborne and space-based applications. This paper describes a radiation tolerant processor designed for space-based Radar applications. The proposed system achieves real-time performance while minimizing power consumption. It is a fully programmable general purpose processor that can be used for existing and future Radar signal processing algorithms as well as other computationally intensive tasks. In addition to programmability, the processor and I/O design facilitate scaling to thousands of processors.

The system is based on synthesizable VHDL and can be Radiation hardened by design or by process technology. This paper gives an overview of the processor design. Further, it explains the software environment and tools available for the programmers. All of the software tools are open source and many are based on the GNU tools. [1]

In this paper we present the performance of some of the key routines such as Fast Fourier Transform, and Householder Q/R factorization used for matrix inversion. Also presented is a sample Radar application that includes a Joint Domain Localized (JDL) Space Time Adaptive Processing (STAP) algorithm.

1. BACKGROUND

This work was based on an existing Air Force Research Laboratory (AFRL) processor design that exhibited excellent power efficiency allowing it to be packaged into dense 3-D multi-chip-modules. [2] The design was referred to as the Wafer Scale Signal Processor (WSSP). It was only partially synthesizable and was implemented in 0.5 micron technology. Currently the design is targeted to 0.18 and 0.13 micron technology under two different efforts.

The major changes that have taken place include making the VHDL fully synthesizable, adding a cache, changing the I/O interfaces from PCI and Myrinet to 10 Gigabit Ethernet, and emulating the processor on Field Programmable Gate Arrays (FPGAs).

Original plan, in 0.13 micron technology, called for 16 processors each with one Megabyte of on-chip memory on a single die with a 10 Gigabit Ethernet switch. However the funding has been reduced requiring a major descope in the design. Currently it is planned to demonstrate a smaller number of processors excluding the on-chip Ethernet crossbar switch.

2. PROCESSING ELEMENTS

The main objective of this design is to provide the Signal Processing community with a flexible general purpose floating point processor that is power efficient and open source. Open source allows it to be customized as needed and facilitates system integration and debugging.

The WSSP is a general purpose floating point processor designed to produce maximum efficiency for vectored operations which constitute the main building blocks of signal processing algorithms. The WSSP can perform two single precision floating point multiplications and two single precision floating point additions per clock cycle, or two double precision operations per clock cycle. That produces a peak performance of 4 FLOPs/clock. Its estimated speed in 0.13 micron technology is 250 MHz, yielding 1.0 GFLOP/sec as a peak performance per processor. Power consumption is much more difficult to determine. A rough estimate is between 2 and 10 GFLOPs/watt.

The processor can be customized to each application or set of applications. For Radar signal processing, a configuration that is very effective is to have numerous processors each with a small (1 or 2 Mbyte) on-chip memory while one processor in 6 to one in 18 has a larger amount of external memory.

The processors with their cache and I/O interface are optionally used in groups of three. This allows voting of outputs to mitigate single event upsets. The on-chip memory is protected with an Error Correcting Code (ECC) providing single bit corrections and multi-bit detection. And the interface I/O has two Cyclic Redundancy Check (CRC) codes. The first protects the header information and the second CRC forms the CRC for the entire packet.

The processors can write directly to each other's memories using a special Ethernet direct write packet. The first CRC provides protection of the header data which contains the memory address where the remainder of the packet is to be stored.

The cache memory is fully associative allowing any memory location to be mapped into any line. With just 16 cache lines, 32 bytes wide, the processor's most complex

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **01 MAY 2005**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Power Efficient Radar Signal Processor**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **ITT AES**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**		
13. SUPPLEMENTARY NOTES **See also ADM002017. Proceedings of the 2005 IEEE International Radar Conference Record Held in Arlington, Virginia on May 9-12, 2005. U.S. Government or Federal Purpose Rights License**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**5**	

instruction (the radix 4 FFT) is able to sustain full speed. This case assumes a 4 cycle access latency to the on-chip memory.

If a unique signal processing function is required and it does not map well to the existing set of vector instructions, new micro-code instructions can be added using the writeable control store. The assembler has been enhanced to allow dynamic addition of instructions without the need to recompile.

Finally, each processor core has an interrupt unit with 7 levels of interrupt requests. The seventh interrupt level is non-maskable.

3. *SOFTWARE DEVELOPMENT SUITE*

The processor comes with a complete suite of software development tools. The software suite consists of C/C++ cross compiler, assembler, linker, simulator, and debugger. Most of the software tools are derived from standard gnu tools.

All software tools have standard UNIX man pages that explain their use and different options for each tool. In addition to the man pages, there is documentation including a user manual and operating instructions explaining the use of the system.

3.1 ISA (Instruction Set Architecture) Simulator

The WSSP simulator is the central piece of software in the development suite. It simulates the behavior of the WSSP processing element at the level of the assembly instruction code. The ISA simulator simulates the behavior of the WSSP down to the number of clock cycles used by every assembly instruction.

In addition to the simulation of the WSSP element, the ISA simulator also simulates the memory attached to the processing element and the network through which messages are exchanged between processors and different nodes in the system.

With the capability of simulating the number of clock cycles taken by each assembly instruction, the ISA simulator can accurately profile the performance of the code. The ISA simulator provides the user with a complete list of performance measures such as total number of clock cycles, total number of vector or scalar floating point operations, and the power consumption taken by the execution of the simulated program. The power consumption data is for the existing older generation hardware. The simulator was given the measured power numbers for each instruction and they are used to calculate the power for any given program. In addition, the ISA simulator's profiler can also be set to report the percentage of time in each subroutine relative to the total time of the whole program. It can also be set to report the number of branch calls, or cache hits/misses to get a good evaluation of the system performance.

The ISA simulator is also used as an interface to the hardware. During software development it makes all the resources of the host system available, providing file access and display services to the embedded processors. It is also possible to run a system of simulated processing elements mixed with actual hardware processors. This enables localizing the debugging process by replacing the failing hardware processor with a simulated one and watching the exact replica of the program running in the simulation environment where more information is available.

3.2 Debugger with (Data Display Debugger) DDD GUI Interface

It is important for the developer to be able to monitor and debug the execution of an algorithm. The WSSP software suite includes the WSSP debugger, which is derived from the gnu debugger. The WSSP debugger comes also with the DDD GUI interface.

With the assistance of the WSSP debugger, the developer can monitor the status of any program. It provides the ability to step through the code in a single or multiple step modes. The debugger can also be accessed remotely through a secure shell with X-Windows capability. The remote access is done through communication with sockets. This approach reduces the response time by running the core software on the remote access machine, and exchanging only data through the remote socket.

The remote accessibility is one of the major advantages of the software development tools of the WSSP environment. Users can run and debug their applications without having to be physically on-site. This feature has been used with an FPGA emulation of the processor, allowing remote programmers to emulate on multiple processors over the internet.

3.3 Real-time operating system RTEMS

The Army developed Real-Time Executive for Military/Multi-processor Systems (RTEMS) was chosen as the operating system for the WSSP. RTEMS is a multi-processing real-time system with desirable features for real-time applications. [3] Extra features have been added to RTEMS in order to meet the requirements of the new platform. One of the added features is the dynamic message passing mechanism. The new system is called DMPI (Dynamic Message Passing Interface). The DMPI is based on the subscriber/publisher model. When a process publishes a message, either one or many subscribers can subscribe to receive that message. This gives more flexibility for exchanging data between processes. At the same time, new receivers or transmitters can be spawned based on the throughput and the dynamic loading of the system.

The operating system environment also allows for dynamic process re-allocation. The dynamic process re-allocation feature provides the flexibility to change the number and type of processes during run time. This allows the user to switch between different modes of operation. One example is to have two modes of operations such as GMTI and SAR. During the mission the user can switch between the two operating modes in a gradual manner. As processors complete one mode they are switched over to the next without having to have them all complete the first mode prior to switching any to the second mode. This prevents dead time in the Radar time-line while waiting for the processing of one mode to complete.

3.4 FPGA Emulation on HHPC

The processor is available as fully synthesizable VHDL code. The code has been successfully loaded and tested on a Heterogeneous High Performance Computer (HHPC) containing Xilinx VirtixII FPGAs. The HHPC consist of 48 dual Pentium nodes each with an Annapolis WildstarII card. Each WildstarII card contains two Xilinx Virtex II 6 million gate FPGAs.

The core of a processor consumes 35% of the area of one 6 million gate Xilinx FPGA. Currently the core has been tested at 20 MHz. Work is planned to improve the speed on the FPGA version of the processor to 70 MHz. Currently the emulation is being used to find any rarely occurring bugs. It allows a much larger set of tests to be run relative to the VHDL simulations that run a factor of one million times slower.

Fixing problems in the emulation stage is faster and less expensive than doing so after building the ASIC. That is why the FPGA emulation is an important concept in the design cycle of the processor.

The FPGA emulation is available on the HHPC in Rome, NY. US DOD users can remotely access the emulation through Kerberos secure access. The emulation is also available for DOD users who want to test or verify software on the processor.

4. PERFORMANCE OF KEY ROUTINES

The following section summarizes some of the simulated results that were run using the ISA simulator. Tests have been run on common functions widely used in Radar signal processing algorithms such as FFT, Pulse Compression and QR factorization. In addition JDL STAP was chosen as an example of an algorithm used in ground moving target detection applications.

4.1 FFT Performance

The WSSP ISA Simulator was used to capture flop and clock count performance of single precision complex one dimensional FFT functions for different size data sets. Simulation results of both Radix-4 and Radix-2 cases are shown in the following figures. Figure 1 shows the flops/clock ratio and the percentage of peak efficiency recorded for different FFT sizes where 4 flops per clock equals 100% of peak computation efficiency for the processor. Figure 2 shows memory requirements and total FLOPs for different one dimensional FFT sizes.

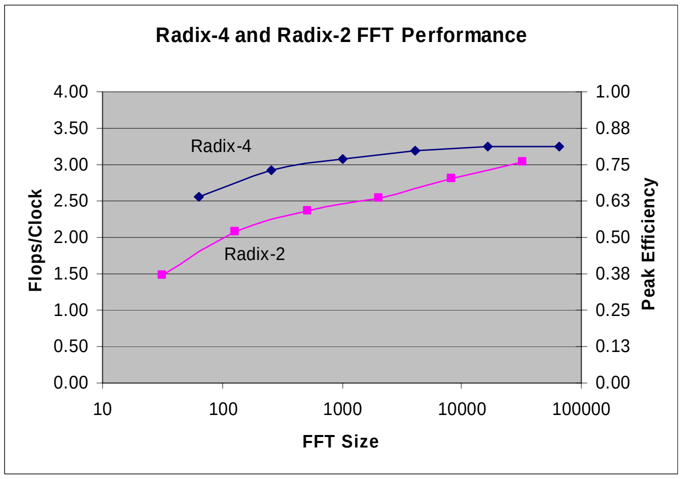

Figure 1 Flop/Clock Ratio and Efficiencies for various FFT sizes

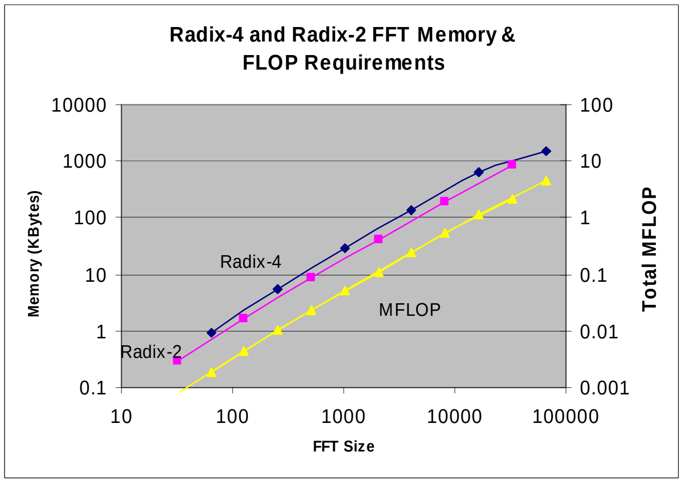

Figure 2 Memory and FLOP requirements for various FFT sizes

4.2 Pulse Compression Performance

After fully characterizing the FFT library, a pulse compression test was then run to determine its performance. The received signal pulse compression was implemented with a forward FFT, a complex multiply with the reference chirp signal spectrum and an inverse FFT.

Figure 3 shows the flops/clock ratio and the percentage of peak efficiency recorded for different compression lengths, again where 4 flops per clock is the 100% peak computation efficiency of the processor. The performance for the larger data sets is approximately 50% of peak. The memory requirements are only slightly larger than the FFT requirements to include the reference radar transmit signal storage.

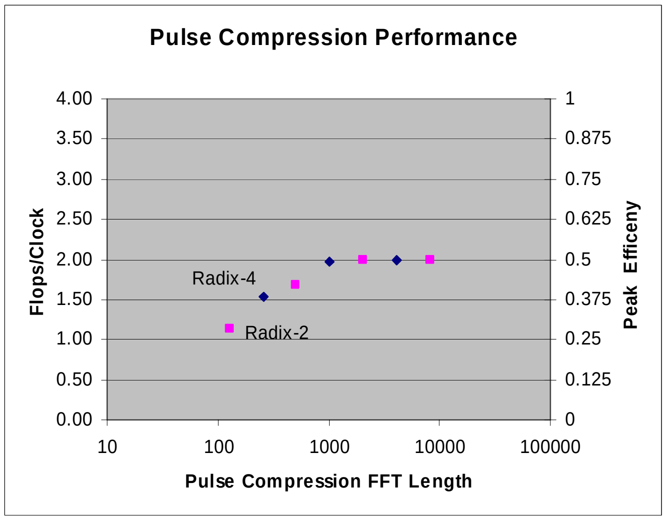

Figure 3 Compression Flop/Clock Ratio and Efficiencies for different FFT sizes

4.3 QR Factorization Performance

QR Factorization performance is very significant to the overall system processing requirements. Depending on the parameters it can account for 80% or more of the total system signal processing requirement. Here the least squares estimation replaces the traditional matrix inverse and a sliding window R matrix block update and QR back solve is applied as the weight calculations are moved along the range dimension. With that in mind, the performance of the block update as shown in Figure 4 was determined. A processing efficiency of approximately 60% was achieved on a 9 degrees-of-freedom (DOF) case, or 9 by 9 matrix. The efficiency improves as the DOF increases.

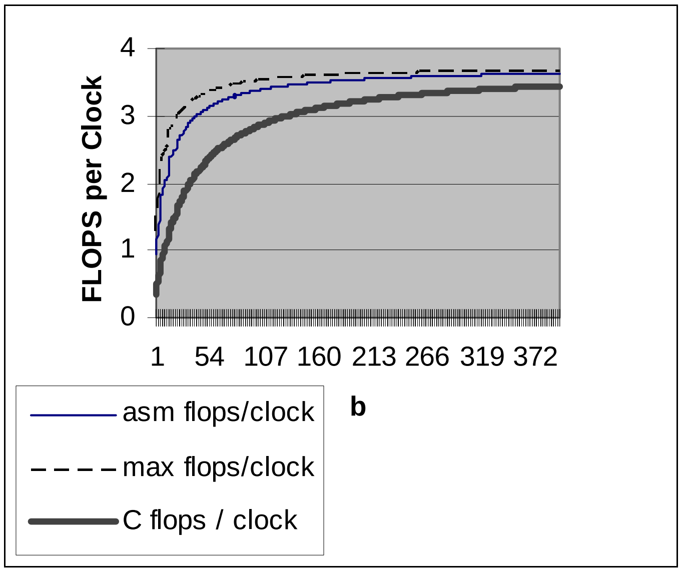

Figure 4 QR block update FLOPS/clock for various lengths.

4.4 JDL Performance

The JDL STAP processing efficiency for a 9 DOF size was 30% including the additional Q matrix back solve, local data handling and final weight calculation. Large scale simulations of the complete processing chain are currently in progress.

Since the JDL STAP processing has a high FLOP to I/O ratio numerous processors can be applied to the task with a relatively minor impact on the overall processing efficiency.

Figure 5 shows a block diagram of the complete processing chain from pulse compression through Constant False Alarm Rate (CFAR) detection.

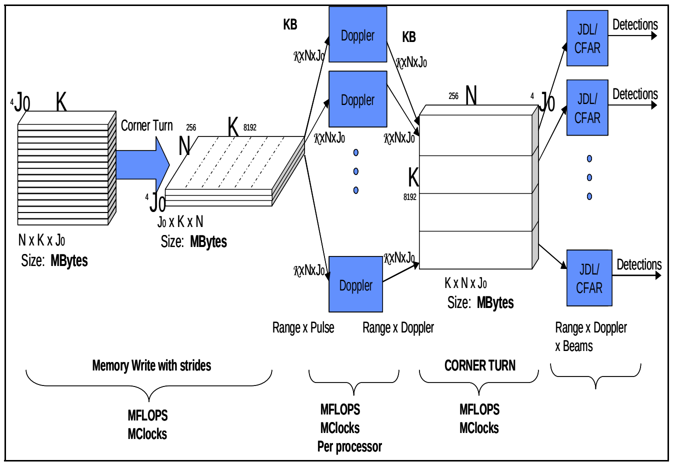

Figure 5 Block diagram for Joint Domain Localized Space Time Adaptive Processing.

A typical data flow would start with pulse compression across numerous processors, followed by saving of the data to a large DRAM. Next the data is taken out across the pulses as opposed to in the range direction. In other words the DRAM is used to corner turn the data by either writing it or reading it with non-unit stride addresses. There are ways to accelerate the non-unit stride access so that they do not become a bottleneck. The pulse ordered data is then Doppler processed by numerous processors and returned to the DRAM. JDL STAP processing can then be assigned to a large number of processors due to its high FLOP to I/O ratio. Generally the required latency for a given Coherent Processing Interval (CPI) is large enough to allow numerous CPIs to be processed in parallel.

5. SUMMARY

Power efficiency and ease of programming are key issues in the implementation of real-time processors for space-based applications. This paper reviewed a synthesizable processor and its development environment as it applies to space-based Radar applications. The system consists of fully programmable general purpose processors that can be used with existing and future Radar signal processing algorithms as well as for other computationally intensive tasks. The I/O and support software allow thousands of the processors to be used together in a high performance computer.

In addition to the synthesizable hardware a complete suite of integrated development software tools is available, including a compiler, assembler, debugger, simulator, emulator, real-time multi-processing operating system, math libraries, and message passing routines. All of the software tools are open source and many are based on the GNU tools.

This paper presented the performance of some of the key routines such as Fast Fourier Transforms, and Q/R Household factorization used for matrix inversion. Benchmark performance data was presented for Joint Domain Localized Space Time Adaptive Processing (JDL STAP).

6. REFERENCES

1. GNU web site www.gnu.org
2. J. Rooks, J. Lyke, R. Linderman, “Wafer Scale Signal Processors and Reconfigurable Processors in a 3-Dimensional Package”, GOMAC 2002 Digest of Papers, Volume XXVII, March, 2002
3. RTEMS web site www.rtems.com