Emulation Of Parallel Processors To Accelerate Multiprocessor System With FPGA

A. Dr. Ch. Ravi Kumar Principal,Dept.Of ECE,Prakasam Engineering College ,Kandukur A.P, . B. Dr. S. K Srivatsa Professor,Dept.Of ECE,Anna University,Chennai.

Abstract: **The idea of using Field-Programmable Gate Arrays (FPGAs) to emulate highly parallel architectures at hardware speeds. FPGAs enable very rapid turnaround for new hardware. We can tape out a FPGA design every day, with a new system fabricated overnight. Another key advantage of FPGAs is that they easily exploit Moore's Law. As the number of cores per microprocessor die grows, FPGA density will grow at about the same rate. Research Accelerator for Multiple processors system is developing infrastructure to support high-speed emulation of large scale, massively parallel multiprocessor systems using FPGA platforms. In this paper, we describe our proposal for a Research Accelerator for Multiprocessors Design Framework (RDF), which has a number of challenging goals. The framework must support both cycle-accurate emulation of detailed parameterized machine models and rapid functional-only emulations. The objective of this paper. is to accelerate multiprocessor systems research through the emulation of usable parallel processor prototypes. Criteria such as low cost, flexibility, observability, credible and repeatable results, and reasonable performance led to the choice of the FPGA platform,.**

Key words: FPGA, multiprocessor, RDL, Memory

1.Introduction.

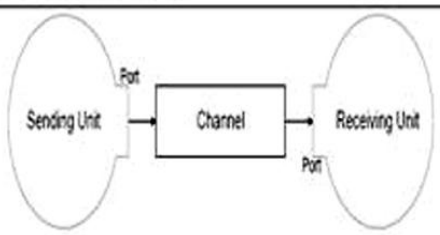
Berkeley Emulation Engine. as the primary host environment. A key goal of Research Accelerator for Multiprocessors is the extraction of credible performance results, even from tests which are no turn in real time. Credible results, however, also require the use of known tests, or applications, many of which are tied to existing instruction set architectures (ISAs). To run such tests, and finish the infrastructure phases of Research Accelerator for Multiprocessors quickly, it is highly desirable to be able to run pre-compiled code for existing architectures. This has led to a major push among the Research Accelerator for Multiple processors participants to find, adapt or develop HDL descriptions of existing ISAs and, with the addition of RDL wrappers, turn them into Research Accelerator for Multiprocessors -compatible processors. While this approach will lend significant credibility to any results, it presents critical challenges, and will delay the project from the longer term goal of parallel system research. Our approach was to develop a decoupled machine model and design discipline, together with an accompanying RAMP Description Language(RDL) and compiler to automate the difficult task of providing cycle-accurate emulation of distributed communicating components.

As the number of cores per microprocessor die grows, FPGA density will grow at about the same rate. Today we can map about 16 simple processors on to a single FPGA, which means we can construct a 1000-processor system in just 64 FPGAs. Such a system is cheaper and lower power than a custom multiprocessor at about $100 and about 1Watt per process.

A configured Research Accelerator for Multiple processors system models a collection of CPUs connected to form a cache-coherent multiprocessor. The emulated machine is called the target, and underlying FPGA hardware (e.g. BEE2) is the host. RAMP is an open-source project to develop and share the hardware and software necessary to create parallel architectures. RAMP is not just a hardware architecture project. Perhaps our most important goal is to support the software community as it struggles to take advantage of the potential capabilities of parallel microprocessors, by providing a malleable platform through which the software community can collaborate with the hardware community. The Research Accelerator for Multiprocessors design framework is based on a few central concepts. A Research Accelerator for Multiprocessors configuration is a collection of loosely coupled units communicating with latency-insensitive protocols over well-defined channels. Figure 1 gives a simple schematic example of two connected units. In practice, a unit will be a large component corresponding to tens of thousands of gates of hardware, e.g., a processor with L1 cache, a DRAM controller, or a network outer stage. All communication between units is via messages sent over unidirectional point-to-point inter-unit channels, where each channel is buffered to allow units to execute decoupled from each other. Each unit faithfully models the behavior of each target clock cycle in the component.

The target unit models can be developed either as RTL code in a standard HDL (currently Verilog, VHDL, and Blue specare supported) for compilation onto the FPGA fabric, or as software models that execute either on attached workstations or on hard or soft processor cores embedded within the FPGA fabric.

Figure 1 Basic RAMP communication model



.

Each unit has a single clock domain. The target clock rate of a unit is the relative rate at which it runs in the target system. For example, the CPUs will usually have the highest target clock rate and all the other units will have some rational divisor of the target CPU clock rate (e.g., the L2 cache might run at half the CPU clock rate). The physical clock rate of a unit is the rate at which the FPGA host implementation is clocked. In some cases, a unit might use multiple physical clock cycles to emulate one target clock cycle, or even require a varying number of physical clock cycles to emulate one target clock cycle. At least initially, we expect that the whole RAMP system will have the same physical clock rate (nominally around 100MHz), perhaps with some higher physical clock rates in I/O drivers. The basic principle is that a unit cannot advance by a target clock cycle until it has received a target clock cycle's worth of activity on each input channel and the output channels are ready to receive another target cycle's worth of activity. This scheme forms a distributed concurrent event simulator, where the buffering in the channels allows units to run at varying physical speeds on the host while remaining logically synchronized in terms of target clock cycles.

2.Vision:

The vision is a scalable, moldboard FPGA-based system that would allow construction of up to a 1024-CPU multiprocessor. This size is an interesting target because many problems that are invisible at 32 processors and awkward at 128 become glaring at 1024.

This scale challenge is true across the architecture, network, operating system, and applications disciplines. This shared artifact would consist of hardware and a collection of "gate ware" (RTL) and software that members of the community would help create. The vision for RAMP centers around the idea that it could become the shared platform for multiprocessor architecture and systems research, in such a way as to supplant all others. Ideally, this would include a simple set of tools, perhaps with a GUI, to create a usable computer system, including processors, memory, network, storage and I/O, which has been tuned for efficient implementation and can run existing applications. Such a system could include the flexibility needed to make architectural changes, e.g. ATLAS transactional memory support. It could also include flexibility in the network design, allowing a researcher to experiment with the performance of various topologies. Because such a system would naturally require specification at a very high level, efficient implementations could be generated, increasing the number of processors, while decreasing the slowdown relative to a full custom ASIC design.

Our goal is to create an infrastructure to attract experts in architecture, operating systems, compilers, programming languages, and applications at many locations, and creating a community that could share tools and techniques since they share a common platform Furthermore, with the ability to run precompiled binaries, modifications could be made independent of operating system and application compilation, drastically reducing the time to do detailed performance studies. RAMP has the potential of offering scalable multiprocessors that are attractive to both architecture and software researchers.

It would provide architects with an economical, easily modified, large-scale platform with tremendous tracing facilities, and software researchers with an economical, large-scale multiprocessor with extensive measurement and debugging support. A higher-level aspect of the RAMP design discipline is that the operation of a unit cannot depend on the absolute or relative latency of messages between units (i.e., all inter-unit communication must be latency insensitive).

By synthesizing processor cores from high level descriptions, including both the ISA, and overall architecture (number of pipeline stages, cache size, etc), we can more easily and efficiently target FPGAs.

3.Emulation Methodology:

Micro architectural techniques are typically evaluated by modifying an existing software cycle-level simulator, such as Simple scalar, to model the interaction of the proposed new hardware with a conventional superscalar out-of-order processor.

Such a simulator is slow; RAMP is not. RAMP is also able to collect and process a tremendous number of measurements with little to no performance impact. Virtually all statistics can be gathered by dedicated logic and state within the FPGA and thus will generally not slow down the simulation Since the largest FPGAs today have less than 1.25MB of block memory total, it will be impossible to place the L2 cache in its entirety on an FPGA. Such an L2would have to be emulated

using off-chip memory. If the L2 cache is associative in any way, multiple reads to external memory may also be required. Such structures will obviously incur more latency and thus will require processor core stalls to accurately simulate. By permitting such emulation, however, the space of architectures that can be evaluated using RAMP becomes much larger. Another alternative is to implement the functionality of a structure in the FPGA and maintain the timing accuracy relative to the whole system. For example, one may model nodes in a parallel system with an FPGA-based functional model augmented with timing information and the interconnection network accurately. Transactions made on the interconnection network returns not only the result but also timing information which can be used by the system to calculate performance. Some architectures, such as the ubiquitous Intel IA-32, do not have a freely available reference implementation and/or are too large/complicated to implement a full micro architecture. In such cases, simulation is probably more appropriate. Most software-based cycle-level simulators are divided into an instruction set interpreter that models the effect of each instruction on architectural state, and parameterizable timing models that calculate the cycles required for each stage of execution. This division of responsibility reduces the effort required to model changes to the micro architecture as only the timing models are modified.

4. Implementation Details:

4.1 Unit Interface

Figure 2 shows the interfaces a RAMP unit must support. Ports comprise the input and output interfaces between units and each port is connected to another port via a channel. The example unit has two input ports (A & B), and one output port (C).

In addition to the ports there are two connections: Start, which is used to trigger the unit to perform one target clock cycle's worth of action, and -Done, which is used to report back to the harness when the unit has completed the target clock cycle. While the ports in the example have a simple fixed message size, the RDL compiler supports complex messages through structures and tagged unions.

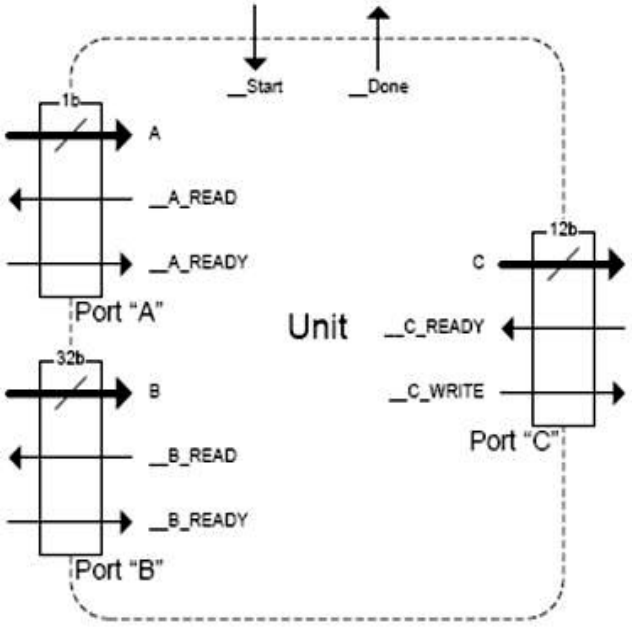


Fig2. Target level interface

By automatically building the support machinery (be it hardware or software) to marshal and transport complex messages, the compiler automates a large and tedious part of the emulation design process.

4.2 Channel Model:

The key to inter-unit communication lies in the channel model, which, along with the inside edge interface, forms the core of the target model. The channel model can be quickly summarized as lossless, strictly typed, point-to-point, and unidirectional with ordered delivery. This should be intuitively viewed as being similar to a FIFO with a single input and output, which carries strictly typed messages. This section expands the above description with the timing parameters necessary for timing-accurate simulations. There are three parameters associated with every channel: bit width, latency and buffering, as illustrated in Figure 3. The bit width of a channel (the number of bits it can carry per target cycle) is the size of a fragment. Latency is the minimum number of target cycles which a fragment must take to traverse the channel. A message in RAMP is the unit of data carried between units, however to perform cycle-accurate simulations, messages may be split into fragments. Fragmentation allows RAMP to decouple the size of messages, which is a characteristic of a unit design, from the size of data moving through the channels. This simplifies experimentation with varying target channel bandwidths, as target channel bandwidths can be modified without changing the target unit design.

The final channel parameter, buffering, is then defined as the number of fragments which the sender may send before receiving any acknowledgement of reception (as in a credit-based flow-control scheme). In general, a channel which must support maximum-bandwidth

communications will require buffering _ 2 _ latency to tolerate the latency in
both directions: the data transfer (fragments movingforward), and the handshaking (credits moving backward).

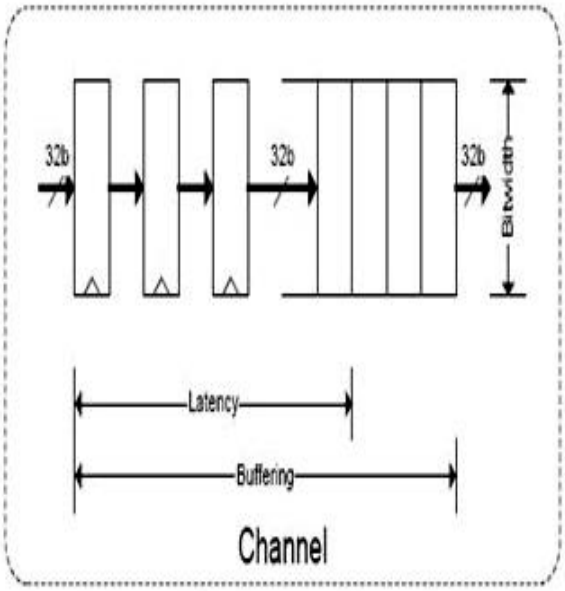


Fig3. Channel model with parameters.

4.3 Types:

We are developing three prototype systems, named RAMP Red, RAMP Blue, and RAMP White. Each of our initial prototypes contains a complete gate ware/ software configuration of a scalable multiprocessor populated with standard processor cores, switches, and operating systems. Once the base system is assembled and software installed, users will be able to easily run complex system benchmarks, and then modify this working system as desired or start from the ground up using the basic components to build a new system. We expect that users will release back to the community any enhancements and new gate ware/software modules. A similar usage model has led to the proliferation of the Simple Scalar framework which now covers a range of instruction sets and processor designs. three prototype systems, named RAMP Red, RAMP Blue, and RAMP White. Each of our initial prototypes contains a complete gate ware/ software configuration of a scalable multiprocessor populated with standard processor cores, switches, and operating systems.

A. RAMP Red.
RAMP Red is the first multiprocessor system with hardware support for transactional memory (TM).Transactional memory transfers the responsibility for concurrency control from the programmer to the system . It introduces database semantics to the shared memory in a parallel system, which allows software tasks (transactions) to execute atomically and in isolation without the use of locks.

Hardware support for TM reduces the overhead of detecting and enforcing atomicity violations between concurrently executing transactions and guarantees correct execution under all cases. RAMP Red implements the Stanford TCC architecture for transactional memory

B. RAMP Blue. RAMP Blue is a family of emulated message-passing machines, which can be used to run parallel applications written for the Message-Passing Interface (MPI) standard, or for partitioned global address space languages such as Unified Parallel C (UPC). RAMP Blue can also be used to model a networked server cluster. The first RAMP Blue prototype is currently under development at UC Berkeley.
C. RAMP White.
RAMP White is a distributed shared memory machine that will demonstrate the open component nature of RAMP by integrating modules from RAMP Red, RAMP Blue and individual RAMPants contributions. .
RAMP White will use the embedded PowerPC processors. Each processor unit contains one processor that is connected to an Intersection Unit (IU) that provides connections to a memory controller (MCU), a network interface unit (NIU), and I/O if the processor unit supports it. The NIU will be connected to a simple ring network..

5. Multiprocessor Systems:

To this point we have repeatedly mentioned the need for small and efficient generated processor in order to build large multiprocessors, but we have not addressed e.g.the challenges of binary translation on these systems. Clearly the universal processor generator must handle multiprocessor synchronization in a predictable manner.
It will need to include support for everything from networked or shared nothing, to cache coherent systems. Work on such project as the CRF memory model, perhaps with added support for transactions such as those under investigation or those used by the Crusoe, should provide a basis for these various synchronization systems and translation between them. Because even RAMP Blue exhibits under utilization of some system level resources, in this case memory bandwidth, we see opportunities for sharing, especially among processors on one FPGA. For example, with a translation cache, translation of code is a relatively infrequent operation, meaning that many processors might share a single hardware translator, or processor

dedicated to translation. Furthermore, as most software scenarios for RAMP machines call for all processors to run a single OS, most translations could be shared by multiple cores, thereby reducing the high memory requirements as reported by Embra.

6. Conclusion:

This paper presented RAMP, an open-source, community developed, FPGA-based emulator of parallel architectures. We presented the RAMP design framework to enable an emulator comprising reusable and compos able design modules to be maintained and developed by a large collaborative community. We highlighted the on-going development of three reference full-system designs. There exists a large body of research in binary translation, and yet opportunities like an FPGA implementation, and challenges like a large multi core system present opportunities for new research. There have also been many projects aimed to create application specific, or parameterized processors and descriptions and yet the relatively performance insensitive application of RAMP, coupled with it's research nature offers a new opportunity to use these tools.

7.References:

[1]J. Wawrzynek, D. Patterson, M. Oskin, S. Lu, C. Kozyrakis,J. Hoe, D. Chiou, and K. Asanovic, "RAMP: Research Accel-erator for Multiple Processors," IEEE Micro, vol. 27, no. 2,2007.

[2] A. Schultz, "RAMP Blue: Design and Implementation ofa Message Passing Multi-processor System on the BEE2,"Master's thesis, University of California, Berkeley, 2006.

[3] James Ball. The Nios II Family of Configurable Soft-Core Processors. In Hot Chips 17, August 2005.

[4] Kenneth Barr, Heidi Pan, Michael Zhang, and Krste Asanovi´c. Accelerating multiprocessor simulation with amemory timestamp record. In IEEE International Symposium on Performance Analysis of Systems and Software,Austin, TX, March 2005.

[5] Ali-Reza Adl-Tabatabai, Christos Kozyrakis, and Bratin Saha. Tutorial: Transactional programming in a multi-core environment. Tutorial at the 15th International Conference on Parallel Architecture and Compilation Techniques (PACT), September 2006.

[6] Nathan L. Binkert, Ronald G. Dreslinski, Lisa R. Hsu, Kevin T. Lim, Ali G. Saidi, and Steven K. Reinhardt. The m5 simulator: Modeling networked systems. IEEE Micro, 26(4):52–60, 2006.

[7] Greg Gibeling, Andrew Schultz, and Krste Asanovic. Ramp architecture and description language, 2005..

[8]Kenneth Barr, Heidi Pan, Michael Zhang, and Krste Asanovi´ c. Accelerating multiprocessor simulation with a memory timestamp record. In IEEE International Symposium on Performance Analysis of Systems and Software,Austin, TX, March 2005.

A). Ch. Ravi kumar bom at Nellore in Andhra Pradesh in India in 1967..He received B.Sc(Physics)degree from Nagarjuna University,in 1985, B.E(Electronics) from MarathwadaUniversity in 1991 and ,M.E(Electronics)with specialisation in Systems and Signal Processing from Osmania University in 1998.He received Doctorate degree from Bharath University Chennai India in 2011 in field of VLSI.

From 1992 on words he is working as Assistant.Associate and Professors levels in reputed Engineering colleges in A.P in India. He is currently working as Principal and Head Department of Electronics in Prakasam Engineering College,Kandukur,Prakasam(Dt)in A.P in India. He is senior member of IACSIT.

B).Dr.S.K Srivatsa received Ph.D in Engg. From IISC. Bangalore,India. He worked as a Senior Professor In Anna Universit,Chennai,India and he got more than 35 years of teaching experience. He guided and produced more than 32
Doctorates from different universities in India He also guided more than 100.Masters degree candidates. He is also a visiting Professor to various universities in India and other countries. Presently working as a senior Professor in St.Joseph College of Engineering In Chennai and also as an academic director for Vignan University. of Engineering In Chennai and also as an academic director for Vignan University.