A Fully-Synthesizable Single-Cycle Interconnection Network for Shared-L1 Processor Clusters

Abbas Rahimi[†], Igor Loi[‡], Mohammad Reza Kakoee[‡], and Luca Benini[‡]

[†] CSE Department, University of California, San Diego, La Jolla, CA, USA

[‡] DEIS, University of Bologna, Bologna, Italy

abrahimi@cs.ucsd.edu, igor.loi@unibo.it, m.kakoee@unibo.it, and luca.benini@unibo.it

***Abstract*—Shared L1 memory is an interesting architectural option for building tightly-coupled multi-core processor clusters. We designed a parametric, fully combinational Mesh-of-Trees (MoT) interconnection network to support high-performance, single-cycle communication between processors and memories in L1-coupled processor clusters. Our interconnect IP is described in synthesizable RTL and it is coupled with a design automation strategy mixing advanced synthesis and physical optimization to achieve optimal delay, power, area (DPA) under a wide range of design constraints. We explore DPA for a large set of network configurations in 65nm technology. Post placement&routing delay is 38FO4 for a configuration with 8 processors and 16 32-bit memories (8x16); when the number of both processors and memories is increased by a factor of 4, the delay increases almost logarithmically, to 84FO4, confirming scalability across a significant range of configurations. DPA tradeoff flexibility is also promising: in comparison to the max-performance 16x32 configuration, there is potential to save power and area by 45% and 12 % respectively, at the expense of 30% performance degradation.**

1. INTRODUCTION

With the flattening out of clock speed improvement and the increasing focus on energy-efficient architectures, chip multiprocessors and parallel computing have recently taken the center stage in research and development. As Moore's law continues to apply in the chip multiprocessor (CMP) era, we can expect to see a geometrically increasing number of processors and memories on-a-chip which places increasing pressure on the design of low-latency, high-bandwidth processor-to-memory interconnection fabrics. The performance of most digital systems today is indeed interconnect-limited, and the design of a high performance on-chip interconnection network is crucial, as the performance impact due to the latency of the interconnection network in a CMP can be as high as the cache miss [1].

Many-core architectures are a reality in products today, and GPUs are perhaps the most widely visible example of this trend. One of the most successful GPUs on the market today, Fermi [2] from NVIDIA, features 512 cores. The computing tile in Fermi is the *streaming multiprocessor* (SM) a tightly coupled processor cluster where 64-Kbyte configurable scratchpad memory/L1 cache is shared among 32 CUDA processor cores [3]. Accesses to local shared memory experience a latency of a couple of cycles. In contrast, global memory is accessed in hundreds of clock cycles.

In this work we focus on communication between the heavily shared multi-banked L1 memory and the cores of a tightly coupled processor cluster. Clearly, L1 processor-to-memory interconnects must provide a huge bandwidth, coupled with ultra-low latency. This level of performance is out of reach for traditional bus-based interconnects [4], even with advanced features like multiple outstanding transactions and out of order completion [5][6]. Networks-on-Chip (NoC) [7], provide bandwidth scalability, but the latency of traditional NoCs is not adequate for L1 processor-to-memory communication [8], and highly optimized special-purpose interconnects are required.

The design of NxM (where N is the number of processors, and M is the number of memory banks) networks for high-performance architectures usually relies on custom circuit design techniques such as pass transistors, low-swing drivers etc. [9]-[11]. Unfortunately this approach is not suitable for architectures featuring soft cores and third-party IPs, which must be compatible with standard technology libraries provided by silicon foundries. An L1 processor-to-memory network provided as a parametric synthesizable IP is therefore highly desirable in this context. Such an IP should come with a carefully tailored logic and physical synthesis strategy, as interconnect delays must be accounted for and minimized to achieve acceptable quality of results.

This paper provides three contributions. First, a fully combinational MoT interconnection network suitable for shared-L1 processor clusters is implemented featuring single-cycle transfer from processor to memory and vice versa. The network provides round-robin arbitration for fair access to memory banks, as well as fine-grained address interleaving to reduce memory bank conflicts. Second, we developed an advanced synthesis and physical optimization strategy which leverages standard design implementation tools, and orchestrates them to achieve high-quality results in terms of delay, power and area (DPA) even for large network instantiations. Third, we explored a wide range of network configurations to analyze scalability and DPA tradeoffs.

Full layout results on realistic floorplans confirm high-performance and scalability across a significant range of configurations: delay is increased almost logarithmically when the number of processors and memories rises. DPA trade-off results demonstrate the flexibility of our soft-IP approach.

2. RELATED WORKS

Parallel computing requires large memory bandwidth [12]: maximizing memory bandwidth is essential in many-core, where the large quantity of parallelism places a heavy request on the memory system [2][3][13]. For this reason, many research efforts have focused on developing ultra-high bandwidth interconnects for multi-banked on-chip memories.

A memory-centric NoC is implemented as an on-chip interconnection to support efficient data transactions for a multi-core processor with ten processing elements [14]. The memory-centric NoC consists of five custom-designed crossbar switches,

978-3-9810801-7-9/DATE11/©2011 EDAA

and eight dual port SRAMs provide shared buffers for inter-PE data transactions. Although the star-connected on-chip network supports 11.2GB/s bandwidth [10], its crossbar switches fabric is partitioned by 4x4 tiles which are implemented by non-synthesizable NMOS pass-transistor logic [15]. As another alternative for custom design technique of crossbars, a 128×128 XRAM-a switched swizzle network- is fabricated in 65nm which achieves a bandwidth 1Tbit/s [8]. These crossbars are not compatible with standard technology libraries provided by silicon foundries.

In openSPARC T1 micro-architecture [16], a processor-cache crossbar has been implemented to accept packets from each of eight SPARC CPU cores, and deliver the packet to one of the four L2-cache banks, the I/O bridge, or the floating-point unit. Although crossbar uses three stage pipelines, it takes more than one cycle to deliver the packet. Another low latency interconnection network based on MoT is implemented in [17] [18], to connect the processing clusters and the memory modules on-a-chip. It provides unique path between each processor and memory module using binary trees of switches. Processing clusters and memory modules are located at the root of trees, while in the traditional MoT [19]-[22], they will be located at the leaves of trees which could degrade performance due to the interference. On the other hand, each packet spends one clock cycle in each switch: when the number of processors and memories increases this architecture becomes a latency bottleneck.

The HyperCore architecture [23] is an example of shared-L1 cluster with high performance-per-Watt. The architecture consists of a hardware synchronizer/scheduler, compact 32-bit RISC cores, shared on-chip memory which is accessed by a high-performance interconnect network. Every path from a RISC processor to the shared memory passes through a series of combinational switches where data and addresses move at hardware speed. Another shared memory system for a tightly-coupled multiprocessor is patented in [24]. Its Baseline interconnection network is implemented as a combinational circuit which contributes to making the interconnection network simple, lean and light-weight. No information is publicly available on the DPA of these networks, and their scalability properties have never been assessed in the open literature.

3. NETWORK ARCHITECTURE

This section provides an architectural description of the proposed interconnection network based on MoT [17]. It supports non-blocking communication between the processing clusters (PCs) and memories modules (MMs), within a single clock. As shown in Figure 1, a combinational path is created through a network of primitive building blocks: routing primitives (circles blocks) and arbitration primitives (square blocks). The former are used to create independent routing paths (routing trees) from the PCs to the arbitration tree (and vice-versa). The latter are used to arbitrate concurrent requests (arbitration tree) and to route them up to the MMs ports and vice-versa.

3.1 Routing Switches

The routing tree consists of simple routing switches which route each packet from processor side to memory side, and vice versa. The packets are routed individually based on packet's address field. As shown in Figure 2, the switch has two directions: forward (PC ports) which sends out the incoming packet form its input port at processor side to one of its output ports at memory side; backward which rolls packet back from memory side to processor side (MM ports). The forward packet contains address, data write, and control signal of memory while the backward packet contains the read data, and acknowledgment signal.

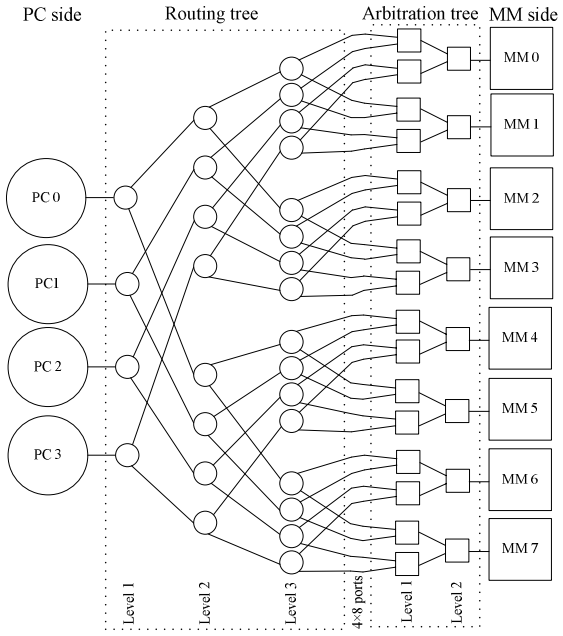

Figure 1. Mesh of trees 4x8: empty circles represent routing switches and empty squares represent arbitration switches.

Each routing switch consists of a MUX, DEMUX, and a simple combinational control logic which provides a fine-grained address interleaving. By using the fully combinational routing switch, a packet with an active request does not need to spend any clock cycles for traveling from PC side up to end of routing tree. Similarly, the backward path used to get back the read data, is active with the request and hold for the entire clock cycle. No arbitration is performed in this block, and the selection (MUX selector) between the two MM_in ports is univocally based on the request address field (PC_in).

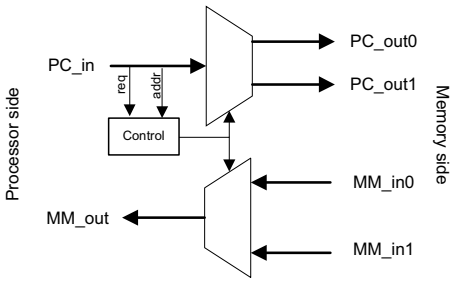

Figure 2: Routing switch.

3.2 Arbitration Switches

The arbitration tree consists of simple arbitration switches which route packets through the routing trees to memory, and vice versa. As shown in Figure 3, the switch has two parallel input ports at the processor side (PC_in ports) and uses a MUX to route data from PC_in to MM_out. On the other side, the response packets (MM_in) from the memory side will be routed to one of the two

possible outputs (PC_out), based on pending grant status. The round-robin algorithm is used to provide a starvation-free arbitration; it means if a request from one processor loses the arbitration in the current clock cycle, it is quarantined to allocate the output in the next clock cycle. The clock signal is used in controller of arbitration switches in order to switch the round-robin flag in case of simultaneous requests.

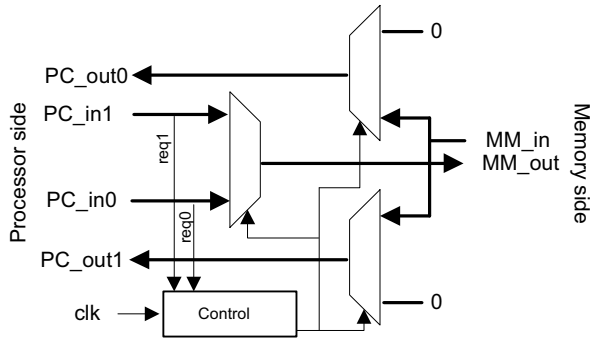

Figure 3: Arbitration switch.

3.3 Network Datapath

As highlighted in Figure 1, the MoT network connects $N=2^n$ PCs and $M=2^m$ MMs. It contains Log_2M levels of routing primitives and Log_2N levels of arbitration primitives. Each memory request issued by PCs must pass through Log_2M levels of routing primitives to reach at one of $M \times N$ leaf nodes in the arbitration switches; and arbitrates among Log_2N levels of arbitration primitives to reach at MM side. In reverse order, memory responses propagate through arbitration and routing levels to reach at PC side. Although there is a unique combinational path between each processor and each memory, packets from different processors direct to the same memory module are arbitrated while passing through arbitration primitives.

The equations 1 and 2 show the total number of routing and arbitration switches needed to connect M PCs to N MMs:

$$\text{total number of routing switches} = \sum_{i=0}^{Log_2M-1} 2^i \times N \qquad (1)$$

$$\text{total number of arbitration switches} = \sum_{i=1}^{Log_2N} \frac{N \times M}{2^i} \qquad (2)$$

Thanks to the modularity features, the network can be easily customized for different cardinalities and different architectural features.

3.4 Network Operation

During a read/write operation, data and control signals are asserted by the PCs in the form of packet, as introduced at the beginning of this section. This packet is routed through routing switches (Figure 1), until it reaches one of $N \times M$ ports of routing tree. In order to reach the memory module the packet must be arbitrated among the other simultaneous packets for the same memory module. After passing through all levels of arbitration switches, the packet reaches the memory module, and the read/write operation can be performed.

Packet routing and arbitration are performed in a combinational way, by using a request and acknowledgment signals for arbitration, and address for routing across the switches. Once the request reaches the last level of the arbitration tree and gets the grant, a valid acknowledgement is asserted and propagated back to the related PC through the routing switches (backward). By receiving the acknowledgment signal, PC is able to issue the next read/write operation at the next clock cycle, otherwise it waits until it is received.

In order to provide a single-cycle latency system, each read/write request must be concluded within the clock period. To achieve this goal, we assume that PCs and network (only for round robin priority switching) are clocked with the main clock CLK, whereas MMs are clocked with a skewed_CLK (same frequency and typically 180° phase shift) as depicted in Figure 4. These two signals are available at the input ports of the network, and are generated through an external block (PLL). Thanks to this strategy, requests asserted on the rising edge of the CLK (1) are propagated in the timing window (1) to (2); after (2) data must be stable, in order to be sampled by MMs during the rising edge of the Skewed_CLK (3). At this point, in case of write, the transaction is done. In case of read, elapsed the access time, the memory presents at its interface a stable data in (4), which is propagated back to PC side through the already allocated backward path (no arbitration is performed here). Data must reach the PC side before (5) and be stable up to (7) in order to avoid setup/hold violations and data corruptions. At the successive rising edge of the CLK in (6), the PC checks the acknowledgment signal: in case of active acknowledgment (active high), PC samples the read data (if any) and injects the next transaction; if zero it waits the next clock cycle keeping stable the packet on the PC ports.

In order to avoid timing violations, three conditions must be satisfied:

- The clock period must be equal or greater than the sum of the worst case delays from PCs to MMs and from MMs to PCs, plus the access time of memory module and setup time of the MM and PC (ref equation 3a). There exists another path from PCs to PCs, when the request signal is not granted in case of simultaneous request for a memory module. This latency is bounded by the entire clock period minus the setup time of the PC.
- Latency from PCs to MMs must be equal or smaller than the clock skew minus the setup time of MM (ref equation 3b).
- Latency from MMs to PCs must be equal or smaller than the clock period minus the skew, the memory access time and the setup time of PC (ref equation 3c).

These conditions are determined by the equation system 3 and graphically depicted in Figure 4. As introduced in section 3.3, the entity of the network latencies increases with the number or switch levels. Is it clear that, for a given network configuration (NxM) and technology, the forward and backward latencies are lower bounded for maximum performances. By playing with the clock frequency, phase shift and memory access time, it is easy to solve the equation system, thus avoiding timing violations.

$$\begin{cases} (a): Max(lat_{PC\_2\_MM} + lat_{MM\_2\_PC} + MM_{accesst\_time} \mid lat_{PC\_2\_PC}) \le Period \\ (b): lat_{PC\_2\_MM} + t_{setup\_MM} \le Period - Skew \\ (c): lat_{MM\_2\_PC} + t_{setup\_PC} \le Skew \end{cases} \qquad (3)$$

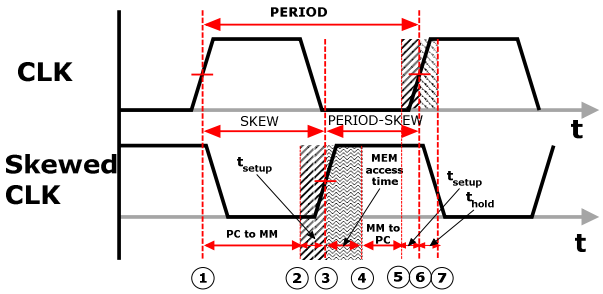

Figure 4: Clock and Skewed Clock representation, and related timing requirements. In (1) the PC injects the request and related datapath signals; stable request is received at MM side in (2), and sampled in (3). In case of write the operation is concluded, while in read case, after the memory access time (4) the read data is stable and is propagated back to PC side; in (5) stable read data reaches the PC ports and it is sampled in (6).

4. EXPERIMENTAL RESULTS

In this section, we discuss the experimental results for the single-cycle network in terms of delay, power, and area (DPA). Several configurations have been analyzed. We quantified the cost of the entire network for several PC and MM cardinalities. To get these results, we synthesized the network with the TSMC 65nm technology library (general purpose process). The front-end flow (Multi V_{TH}) has been performed with Synopsys Design Compiler in topographical mode, while the back-end with Cadence SoC Encounter. The sign off has been made with PrimeTime, while functional verification is performed with Mentor Graphics' ModelSim.

4.1 Design Flow

Figure 5 depicts an overview of the specialized design flow used to maximize speed and explore DPA trade-offs. The network is generated through our high-level generator which takes as input the network template (number of PCs and MMs) and user constraints (BW, datawidth etc.). The output of this stage is a Verilog synthesizable RTL description of the network, and the timing constraints at each port of the network. The synthesis stage is performed in two passes: the first is a pure logic synthesis, and the network is synthesized without physical constraints. This preliminary output netlist is used in the P&R tool to perform the power planning and floorplanning (with pin budgeting), and at the end of this step, the physical information are exported in a "def" file and back-annotated in the synthesis tool, with topographical features enabled. The second synthesis run performs both remapping and coarse placement plus physical optimization taking into account physical geometries based on the back-annotated floorplan[1]. Using this methodology, good convergence between post-synthesis and post-layout results is achieved early in the flow with only one iteration cycle (correlations in the order or 90-95% is achieved).

The P&R flow is based on the top-down approach. To save runtime, we used dummy hard macros to mimic the timing behavior and physical obstructions of PCs and MMs. The network is flatten and placed in a single tile in the center of the die area as depicted in Figure 9.

[1] A Steiner tree model (instead of wire load model) is used to measure wiring distances and capacitive loads are computed using these lengths.

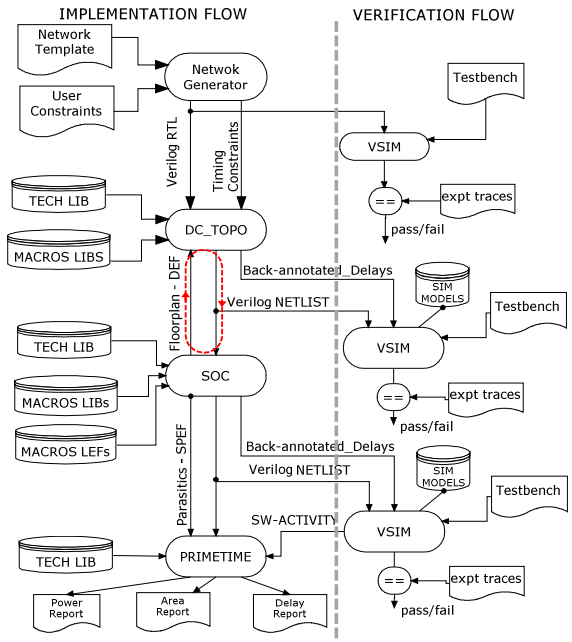

Figure 5. Overview of the design flow.

Finally, the extracted netlist, back-annotated parasitic and delays are used to perform area, delay and power analysis with PrimeTime. To ensure the correctness and quality of the achieved results, in parallel with the implementation flow we run the verification flow. A pass/fail test has been adopted for this purpose.

4.2 Combinational Network

In this sub-section we present the DPA results for the entire interconnection network for the following configurations (32bits data and address):

- 8PCs and 8, 16 and 32MMs
- 16PCs and 16, 32 and 64MMs
- 32PCs and 32 and 64 MMs

We choose to analyze the configurations where the numbers of MMs are equal or greater than the number of PCs, because essentially each PCs leads to one or more MMs. Moreover, the address space can be interleaved in different ranges, and assigned to different MMs, to reduce the memory contentions while increasing overall system bandwidth.

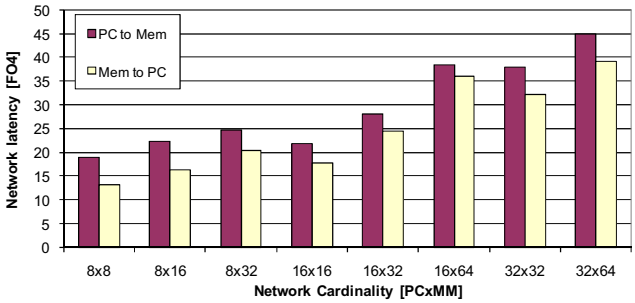

Figure 6. Forward and Backward network delay for different cardinalities (32bit).

Figure 6 shows the post-layout delay results for the explored network cardinalities (32bit channels width). As described in section 3.4, equations 3, network performance is determined by two paths: forward path from PCs to the MMs, backward path in the reverse direction. The first involves both path arbitration and switch routing and network traversal, whereas in the backward, the path is already established and the delay contribution is entirely due the switches traversal. As shown in Figure 6, the delay of the network for roundtrip traversal ranges from 32FO4 (8x8) to 84FO4 (32x64). The delay for 8x16 configuration is 38FO4. When the number of both processors and memories rises by a factor of 4, the delay closely increases logarithmically by 2.2 because the levels of routing and arbitration trees are a logarithmic function from the number of processors and memories.

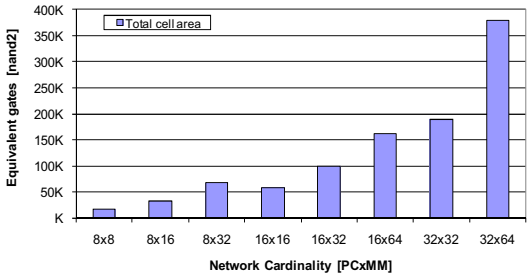

Figure 7: Network area cost for different cardinalities.

Figure 7 depicts the total cell area of the network for several cardinalities. As described in section 3.3, the area cost is directly related with the amount of routing and arbitration switches with O(*NM*). For the 8x16 configuration the area cost is limited to 32K equivalent gates, while for 32x64 it strictly increases less than O(*NM*), 380K equivalent gates . It shows the great ability of network for saving area while supporting large number of PCs and MMs.

Finally, Figure 8 shows the power consumption results for a 32bit based networks in terms of cell internal, switching and leakage power. Starting from the 16x32 setup, the contribution of net switching power dominates the overall consumption, mainly because as the number of macros increases (length of wires is dominated by the size of the die), the wire-length and related power rises as well.

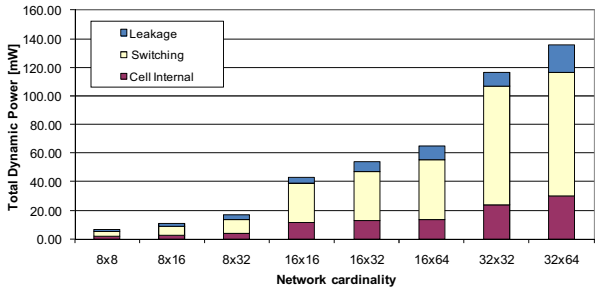

Figure 8: Network power consumption for different cardinalities.

The layout view of the 32x64 configuration is shown in Figure 9, where the Network is surrounded by PCs and MM clusters. Since the network is centralized, in order to minimize the wire-length, the Network tile is placed in the center of the die, while no constraints are applied to MMs and PCs (any path from MM from/to PC is routed through the network).

DPA results show the potential of the proposed to network to handle such a big system as a high-performance interconnect with acceptable area and power cost.

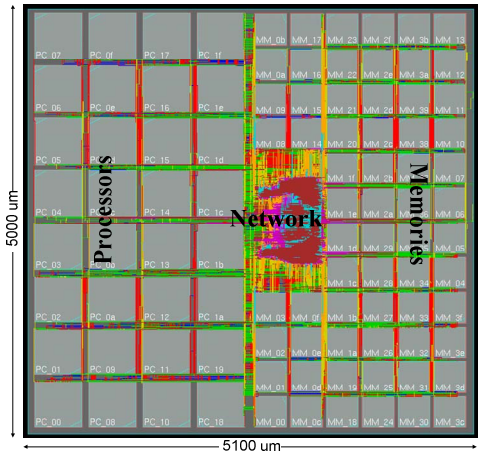

Figure 9. Layout of 32x64 configuration.

4.3 Delay-Power-Area Trade-offs

We explore DPA trade-offs for different design constraints for the 16x32 configuration. The trade-offs show the potential of our synthesizable interconnection network to work with different target frequencies and achieving area and power saving. So the interconnection network can be easily adapted to the demands of the new architectures (featuring soft cores and third-party IPs), thanks to the fully-synthesizable and fully automated flow. Figure 10 shows the trade-off between total area cell of the network and its target frequency. Relaxing the target frequency, the tools are able to infer small (driving strength) and less power hungry cells (regular or high threshold voltage), thus saving both area and power. In comparison to the max-performance 16x32 configuration (310 MHz), we can save area up to 12%, at the expense of 30% performance degradation.

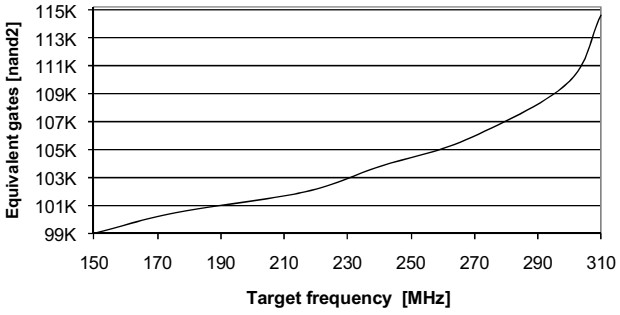

Figure 10. Area-Performance trade-off for 16x32 network.

The trade-off between power and performance is illustrated in Figure 11. By changing the target frequency form 310Mhz down to 210Mhz, 50% total power saving is achieved for network. The plot shows three regions: starting from the relaxed timing constraint, the tool maps the architecture on high threshold (HVT) cells which yield lower power. As the target frequency increases, the design is mapped on regular voltage threshold (RVT) cells, in order to achieve simultaneous timing requirements and area

minimization. This trend is sustained until 290MHz; after this point the timing constraints are very tight and design is dominated by low voltage threshold (LVT) cells, which are fast and power hungry (leaky cells).

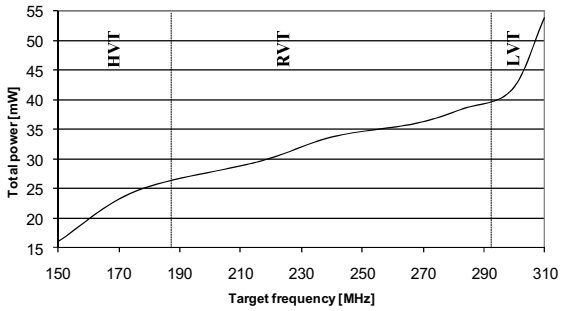

Figure 11. Power-Performance trade-off for 16x32 network.

5. CONCLUSION

In this paper we presented a fully-synthesizable single-cycle interconnection network for shared-L1 processor clusters which is coupled with an advanced design automation strategy mixing advanced synthesis and physical optimization. We explored the network in terms of delay, power, and area metrics for different system configurations and design constraint. Our post placement & routing results show that the delay of interconnects is just 38FO4 for the 8x16 configuration. Rising the number of PCs and MMs by a factor of 4 leads to only 2.2X in delay incensement which is close to a (highly desirable) logarithmic growth. In terms of DPA trade-offs, the 16x32 configuration has the potential to power and area saving of 45% and 12% respectively, at the expense of 30% performance degradation. In future work, we will focus on multicast support, configurable address interleaving and topology optimization to further reduce delay.

6. ACKNOWLEDGMENTS

The financial contribution of FP7 projects Pro3D (GA n. 248776) and Therminator (GA n.248603) is gratefully acknowledged.

7. REFERENCES

[1] S. Akram, R. Kumar, D. Chen, "Workload adaptive shared memory multicore processors with reconfigurable interconnects," *in Proc. of the 7th IEEE Symposium on Application-Specific Processors, SASP*, July 2009, pp. 7-14.

[2] NVDIA, The next generation CUDA architecture, code named Fermi, [online]. Available: www.nvidia.com/object/ fermi_architecture.html

[3] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, no. 2, pp 56 - 69, April 2010.

[4] F. Angiolini, P. Meloni, S. Carta, L. Raffo, and L. Benini, "A layout-aware analysis of networks-on-chip and traditional interconnects for mpsocs, "*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*," vol. 26, no. 3, pp. 421-434, March 2007.

[5] STMicroelectronics, The STBus Interconnect. [Online]. Available :www.st.com

[6] ARM Ltd., The Advanced Microcontroller Bus Architecture (AMBA) Homepage. [Online]. Available: www.arm.com /products/solutions/AMBAHomePage.html

[7] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm, " *Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[8] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, 2006.

[9] S. Satpathy, Z. Foo, B. Giridhar, D. Sylvester, T. Mudge, D. Blaauw, "A 1.07 Tbit/s 128x128 swizzle network for SIMD processors," *IEEE Symposium on VLSI Circuits (VLSI-Symp)*, pp. 81-82, 2010.

[10] K. Lee et al., "A 51 mW 1.6 GHz on-chip network for low-power heterogeneous SoC platform," *in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2004, pp. 152-153.

[11] Se-Joong Lee, et. al., "An 800MHz star-connected on-chip network for application to systems on a chip", *IEEE Digest of International Solid State Circuits Conference*, vol. 1, 2003, pp. 468-489.

[12] M. Horowitz, and W. Dally, "How scaling will change processor architecture," *In IEEE International Solid-State Circuits Conference (ISSCC)*, 2004, pp. 132-133.

[13] George L. Yuan , Ali Bakhoda , Tor M. Aamodt, "Complexity effective memory access scheduling for many-core accelerator architectures, " *in Proc. of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 34-44.

[14] D. Kim et al., "Memory-centric network-on-chip for power efficient execution of task-level pipeline on a multi-core processor," *IET Computers & Digital Techniques*, vol. 3(5), pp. 513-524, September 2009.

[15] Kim et al., "Solutions for real chip implementation issues of NoC and their application to memory-centric NoC, " *Int.Symp. on Networks-on-Chip*, 2007, pp.30-39.

[16] OpenSPARC T1. [online].Available: www.opensparc.net/ opensparc-t1/index.html

[17] A. O. Balkan, G. Qu, and U. Vishkin, "A mesh-of-trees interconnection network for single-chip parallel processing," *In Proc. of the App.-Specific Systems, Architectures and Processors (ASAP)*, 2006, pp. 73 – 80.

[18] Aydin O. Balkan , Gang Qu , Uzi Vishkin, "An area-efficient high-throughput hybrid interconnection network for single-chip parallel processing," in Proc. of the 45th annual conference on Design automation, 2008, pp. 435-440.

[19] R. I. Greenberg and L. Guan, "On the area of hypercube layouts, " *Information Processing Letters*, 84:41–46,2002.

[20] A. DeHon, "Compact, multilayer layout for butterfly fat-tree, " *In Proc. of Symposium on Parallel Algorithms and Architectures (SPAA),* 2000, pp. 206–215.

[21] C.-H. Yeh, "Optimal layout for butterfly networks in multilayer VLSI, " *In Proc International Conference on Parallel Processing (ICPP)*, 2003, pp. 379 – 388.

[22] C. E. Leiserson, "Fat trees: Universal networks for hardware-efficient supercomputing, " *IEEE Trans. Computer*, 34(10):892–901, Oct. 1985.

[23] Plurality, Ltd. The hyperCore architecture, *white paper*, January 2010.

[24] N. Bayer, A. Peleg, "Shared memory system for a tightly-coupled multiprocessor," Pub. no. WO/2009/060459, 2009.