Appears in the Proceedings of the 2002 International Conference on Computer Design

VLSI Design and Verification of the Imagine Processor

Brucek Khailany, William J. Dally, Andrew Chang, Ujval J. Kapasi, Jinyung Namkoong, Brian Towles
Computer Systems Laboratory
Stanford University
Stanford, CA 94305 USA
{khailany, billd, achang, ujk, namkoong, btowles}@cva.stanford.edu

Abstract

The Imagine stream processor is a 21 million transistor chip implemented by a collaboration between Stanford University and Texas Instruments in a 1.5V 0.15 μm process with five layers of aluminum metal. The VLSI design, clocking, and verification methodologies for the Imagine processor are presented. These methodologies enabled a small team of graduate students with limited resources to design a high-performance media processor in a modern ASIC flow.

1. Introduction

Imagine is a 21-million transistor chip implemented in a 1.5V 0.15 μm CMOS Texas Instruments (TI) process with five layers of metal.[1] A block diagram of the Imagine stream processor [6] is shown in Figure 1. The chip was designed by a collaboration between Stanford University and the ASIC group at TI. Stanford designed the architecture, logic, and did the floorplanning and cell placement. TI completed the layout and layout verification.

This paper highlights the two major challenges with the Imagine VLSI design: building a relatively high-performance datapath-style design in a standard ASIC design methodology and doing this with a small team of graduate students. To accomplish this task, a *tiled region* design methodology was used, an approach where timing-driven placement is avoided and the designer controls the placement of small regions of standard cells. Results obtained with this methodology and techniques used for functional verification of the Imagine processor are presented.

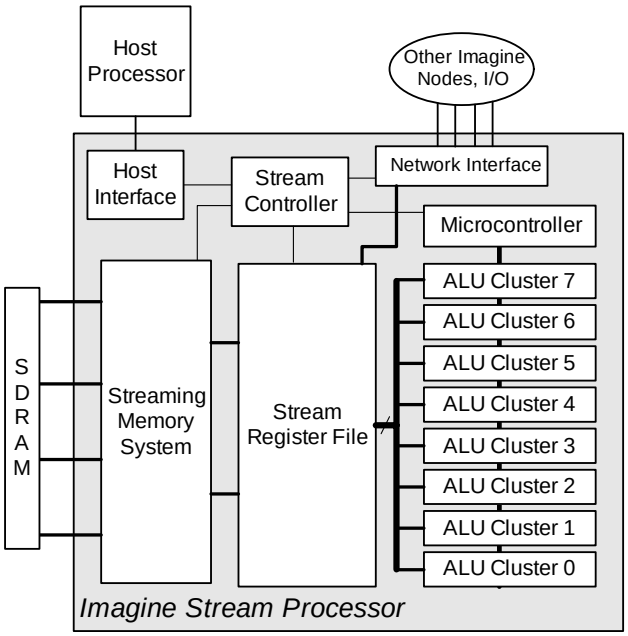

Figure 1: Imagine architecture block diagram

2. Schedule

By summer 1998, the Imagine architecture specification had been defined and a cycle-accurate C++ simulator for Imagine was completed and running. In November 1998, logic design had begun with one Stanford graduate student writing the RTL for an ALU cluster. By December 2000, the team working on Imagine implementation had grown to five graduate students and the entire behavioral RTL model for Imagine had been completed and functionally verified.

The Imagine floorplanning, placement, and layout was

[1] The fabrication process has a drawn gate length of 0.15 μm, an effective gate length of 0.13 μm, and has five layers of aluminum with metal width and spacing rules typical to a 0.18 μm CMOS process [5] [9]. A fan-out-of-4 inverter delay in the typical corner of this process is 73 ps.

The research described in this paper was supported by an Intel Foundation Fellowship, the Defense Advanced Research Projects Agency under ARPA order E254 and monitored by the Army Intelligence Center under contract DABT63-96-C0037, by ARPA order L172 monitored by the Department of the Air Force under contract F29601-00-2-0085, by Intel Corporation, by Texas Instruments, and by the Interconnect Focus Center Program for Gigascale Integration under DARPA GrantMDA972-99-1-0002.

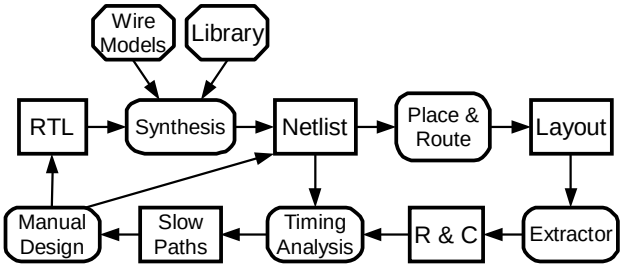

Figure 2: Standard ASIC Design Methodology

carried out by splitting the design into five unique *subchips* and one top-level design. In November 2000, the first trial placement of one of these subchips, an ALU cluster, was completed by Stanford. By August 2001, the final placement of all five subchips and the full-chip design was complete and Stanford handed the design off to TI for layout and layout verification. In total, between November 1998 when behavioral RTL was started and August 2001 when the placed design was handed off to TI, Stanford expended 11 person-years of work on the logic design, floorplanning, and placement of the Imagine processor. Imagine parts entered a TI fab in February 2002. First silicon was received in April, 2002 and functionality has been verified in the lab.

3. Design Methodology Background

Figure 2 shows a typical ASIC tool flow. RTL is written in a hardware description language such as Verilog and is mapped to a standard-cell library with a logic synthesis tool such as Synopsys Design Compiler [10]. Wire lengths are estimated from statistical models and timing violations are fixed by resynthesizing with new timing constraints or by restructuring the logic. After pre-placement timing convergence, designs are then passed through an automatic place and route tool, which usually uses a timing-driven placement algorithm. After placement, wire lengths from the placed design are extracted and back-annotated to a static timing analysis (STA) tool. However, when actual wire lengths do not match predicted pre-placement statistical-based wire lengths, this can cause a timing problem and can lead to costly design iterations, shown in the bottom feedback loop.

Recent work in industry and academia has addressed many of the inefficiencies in ASIC flows. This work can be grouped in two categories: improving timing convergence and incorporating datapath-style design in ASIC flows.

Physically-aware synthesis approaches [11] attempt to address the shortcomings of traditional flows by concurrently optimizing the logical and physical design, rather than relying on statistically-based wire-length models. While these approaches deliver modest improvement in timing performance and area, their principal benefit is to reduce the number of iterations required for timing convergence.

Many researchers have demonstrated that identifying and exploiting regularity yields significant improvements in density and performance for datapath structures in comparison to standard ASIC place and route results [2]. In particular, researchers have shown numerous automated techniques for extracting datapath structures from synthesized designs and doing datapath-style placement [7] [8] [3]. However, widespread adoption of these techniques into industry-standard tools had not yet occurred by the time the VLSI design for the Imagine processor was started.

4. Imagine Design Methodology

Given the small size of the Stanford design team and the need to interface with the Texas Instruments flow for doing layout, the design methodology for Imagine was constrained to use the basic tool flow shown in Figure 2. However, to take advantage of the datapath regularity in Imagine and to expedite timing convergence, this tool flow was modified. Physical-aware synthesis techniques were not available while the VLSI design was carried out, so a *tiled region* design methodology was used. This methodology provides similar advantages in gate density to the techniques presented in Section 3 for doing datapath-style design in a standard cell technology.

In the ASIC methodology used on Imagine, the design is partitioned into subchips. A flat placement is run for each subchip and hierarchical placement is used only for top-level assembly. Table 1 shows the number of instances, area, and gate area in equivalent NAND2 gates for each of the five subchips: the ALU cluster (CLUST), the microcontroller (UC), the stream register file (SRF), the host interface / stream controller / network interface (HISCNI), and the memory bank (MBANK). Each of these subchips corresponds directly to units in Figure 1 except the MBANK. The streaming memory system is composed of 4 MBANK units: 1 per SDRAM channel. Also shown is the top-level design, which includes glue logic between subchips and I/O interfaces.

In addition to the gates listed in Table 1, some of the subchips also contain SRAM's instantiated from the TI ASIC library. The UC contains storage for 2048 576-bit VLIW instructions [6] organized as 9 banks of single-ported, 1024-word, 128-bit SRAM's. The SRF contains 128 KBytes of storage for stream data, organized as 8 banks of single-ported, 1024-word, 128-bit SRAM's. There is a dual-ported, 256-word, 32-bit SRAM in each ALU cluster for scratchpad memory. Finally, the HISCNI subchip contains SRAM's for input buffers in the network interface and for stream instruction storage in the stream controller.

Table 1: Subchip statistics

	Instances	mm^2	Gate Area	#
CLUST	130,000	5.1×0.8	304K	8
UC	27,000	6.2×1.4	27K	1
SRF	314,000	9.0×4.0	1.31M	1
HISCNI	98,000	8.0×1.4	320K	1
MBANK	57,000	1.2×1.8	169K	4
Top Level	75,000	16×16	351K	1
Full Chip	1,782,000	16×16	5.12M	1

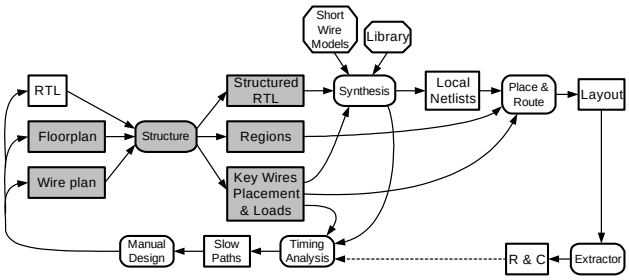

Figure 3: Tiled Region Design Methodology

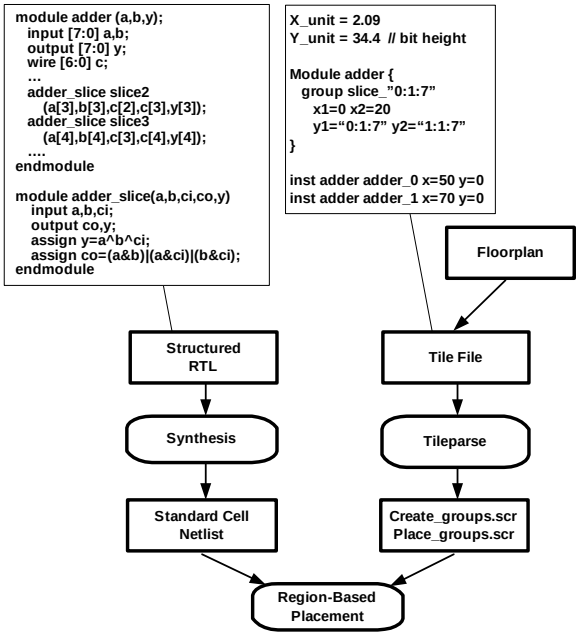

Figure 4: Tiled Region Floorplanning Details

Several of the subchips listed above benefit from using datapath-style design. Specifically, each ALU cluster contains six 32-bit floating-point arithmetic units and fifteen 32-bit register files. Exploiting the datapath regularity for these units keeps wire lengths *within* a bitslice very short, which in turn leads to smaller buffers, and therefore a more compact design. In addition, control wires are distributed *across* a bitslice very efficiently since cells controlled by the same control wires can be optimally aligned. The SRF, which contains 22 8-entry 256-bit streambuffers [6], also benefits from the use of datapaths. The 256 bits in the streambuffers align to the 8 clusters' 32-bit-wide datapath, keeping wires predictable and short and allowing for efficient distribution of control wires.

The tiled-region basic flow used on Imagine is shown in Figure 3. It is similar to the typical ASIC methodology shown previously in Figure 2. However, several key additional steps, shown in gray, have been added in order to allow for datapath-style placement and to reduce costly design iterations. First, in order to make sure that datapath structure is maintained all the way through the flow, two RTL models were used. A second RTL model, labeled *structured RTL*, was written. It is logically equivalent to the behavioral RTL, but contains additional logical hierarchy in the RTL model. Datapath units such as adders, multipliers, and register files contain submodules that correspond to datapath bitslices. These bitslices correspond to a physical location along the datapath called a region. Regions are also used with typical ASIC methodologies, but the tiled-region flow has a much larger number of smaller regions (typically 10 to 50 instances per region) when compared to timing-driven placement flows.

In addition to the floorplanning of regions, the subchip designer also must take into account the *wire plan* for a subchip. The wire plan involves manually annotating all wires of length greater than one millimeter with an estimated capacitance and resistance based on wire length between regions. By using these manual wire-length annotations during synthesis and timing analysis runs, statistical wire models generated during synthesis are restricted to short wires. Manual buffers and repeaters were also inserted in the structured RTL for long wires. With wire planning, pre-placement timing more closely matches post-placement timing with annotated wire resistance and capacitance.

A more detailed view of the floorplanning and placement portion of the tiled-region methodology is shown in Figure 4. Consider an 8-bit adder. It would be modeled with the statement $y = a + b$ in behavioral RTL. However, the structured RTL is split up by hand into bitslices as shown in Figure 4. The structured RTL is then either mapped by hand or synthesized into a standard-cell netlist using Synopsys Design Compiler [10]. In conjunction with the netlist generation, before placement can be run, floorplanning has to be completed. In the tiled-region design methodology, this is done by writing a *tile file*. An example portion of a

Table 2: Imagine placement results

	Occ	# Regions	Placement
CLUST	65.1%	1,556	Tiled-Region
UC	56.3%	102	Tiled-Region
SRF	54.5%	6,640	Tiled-Region
HI/SC/NI	38.9%	237	Tiled-Region
MBANK	69.1%	15	Timing-Driven
Top Level	63.3%	1,095	Tiled-Region

tile file is shown in Figure 4. The tile file contains a mapping between logical hierarchy in the standard cell netlist and a bounding box on the datapath given in x-y coordinates. The example tile file shows how the eight bitslices in the adder would be tiled in bit positions 0 to 7 along the datapath at x coordinates 0 to 20. Arbitrary levels of hierarchy are allowed in a tile file, allowing one to take advantage of modularity in a design when creating the floorplan. The tile file is then passed through a tool developed by Stanford called *tileparse*. Tileparse flattens the hierarchy of the tile file and outputs scripts which are later run by the placer to set up the regions. Once the regions have been set up, but before running placement, the designer can look at the number of cells in a region and iterate by changing region sizes and shapes until a floorplan that fits is found. Finally, the Avant! Apollo-II automatic placement and global route tool [1] is used to generate a trial placement on the whole subchip. These steps are then iterated until a floorplan and placement with satisfactory wiring congestion and timing has been achieved. The steps following placement in the tiled-region design methodology do not differ from the typical ASIC design methodology.

5. Imagine Implementation Results

Table 2 shows the placement results for the subchips and top level design. Standard cell occupancy is given as a ratio of standard cell area to placeable area. Area devoted to large power buses or SRAMs is not considered placeable area. It is also important to note that occupancy is dependent on the characteristics of the subchip. For example, aspect ratio considerations contributed to the lower occupancies of the HISCNI subchip. Also, the SRF has regions of low occupancy for interfacing with the SRAM's and other subchips that reduce its overall occupancy. However, in regions where large numbers of datapaths were used such as in the streambuffer datapaths, occupancy was over 80%. Tiled-region placement was used on all of the subchips except for the smaller MBANK subchip, which did not have logic conducive to datapath-style placement. Timing results are included in Table 3. Maximum clock frequency and critical path for each clock domain in fan-out-of-4 inverter delays (FO4s) are shown. Results were measured using standard RC extraction and STA tools at the typical process corner.

By using tiled-regioning, large subchips such as the SRF and CLST with logic conducive to datapath-style placement were easily managed by the designer. For example, placement runs for the SRF, which contained over 300,000 instances took only around one hour on a 450 MHz Ultrasparc II processor. This meant that when using tiled-region placement on these large subchips, design iterations proceeded very quickly. Furthermore, the designer had fine-grained control over the placement of regions to easily fix wiring congestion problems. For example, the size and aspect ratio of datapath bitslices could be modified as necessary to provide adequate wiring resources. In order to more extensively compare the effectiveness of tiled-regioning to timing-driven placement on these subchips, gate density, timing performance, and design time would need to be measured on subchips optimized using both techniques. Unfortunately, quantitative data for such a comparison on the Imagine design was not available. The benefits from using datapath-style placement in ASIC designs has been well-studied by other researchers, as presented in Section 3.

Figure 5 shows a die photograph of the Imagine processor with the five subchips highlighted. Its die size is 16 mm $\times$ 16 mm. The IO's are peripherally bonded in a 792-pin BGA package. There are 456 signal pins (140 network, 233 memory system, 45 host, 38 core clock and debug), 333 power pins (136 1.5V-core, 158 3.3V-IO, 39 1.5V-IO), and 3 voltage reference pins. The additional empty area in the chip plot is either glue logic and buffers between subchips or is devoted to power distribution.

6. Imagine Clocking Methodology

Most ASIC's use a tree-based clock distribution scheme. This approach was also used on Imagine, but distributing a high-speed clock with a large die size and many clock loads with low skew was challenging. Typical high-performance custom designs use latch-based design to enable skew tolerance and time-borrowing. However, a large variety of high-performance latches were not available in Imagine's standard cell library, so an edge-triggered clocking scheme where clock skew affects maximum operating frequency was used. Latches, instead of flip-flops, were used in some register file structures in the ALU clusters in order to reduce area and power dissipation.

In order to distribute a clock to loads in several subchips while minimizing skew between the loads, the standard flow in the TI-ASIC methodology was used. First, after each subchip was placed, a clock tree was expanded within each

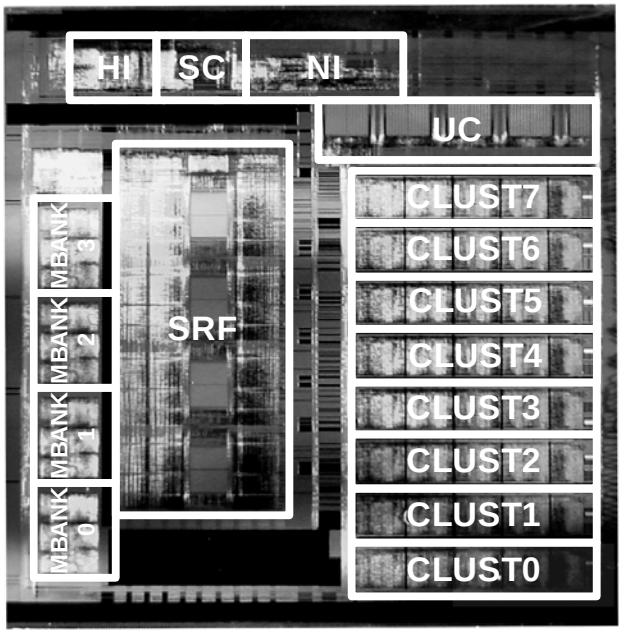

Figure 5: Die Photograph

Table 3: Imagine timing results

Clock	Max Freq	T_{cycle} (FO4s)	Clock Loads
iclk	296 MHz	46.3	160K
sclk	148 MHz	92.6	8.8K
hclk	175 MHz	78.3	2.7K
mclk	233 MHz	58.6	19K
nclkin	296 MHz	46.3	166
nclkout	296 MHz	46.3	55

subchip using available locations in the floorplan to place clock buffers and wires. Skew between the clock loads was minimized using Avant! Apollo [1]. Later, when all of the subchips were instantianted in the full-chip design, delay elements were inserted in front of the clock pins for the subchips so that the insertion delay from the inputs of the delay elements to all of the final clock loads would be matched for the average insertion delay case. Next, the same flow used on the subchips was used to synthesize a balanced clock tree to all of the inputs of the delay elements and the leaf-level clock loads for clocked elements in the top-level design.

Imagine must interface with several different types of I/O each running at different clock speeds. For example, the memory controller portion of each MBANK runs at the SDRAM clock speed. Rather than coupling the SDRAM clock speed to an integer multiple of the Imagine core clock speed, completely separate clock trees running at arbitrarily different frequencies were used. In total, Imagine has 11

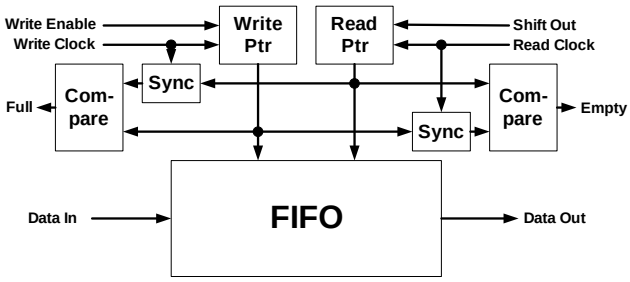

Figure 6: Asynchronous FIFO Synchronizer

clock domains: the core clock (iclk), a clock running at half the core clock speed (sclk), the memory controller clock (mclk), the host interface clock (hclk), four network input channel clocks (nclkin_n, nclkin_s, nclkin_e, nclkin_w), and four network output channel clocks (nclkin_n, nclkout_s, nclkout_e, nclkout_w). These clocks and the loads for each clock are shown in Table 3, but for clarity, only one of the network channel clocks is shown. The maximum speed of the network clocks were architecturally constrained to be the same speed as iclk, but can operate slower if needed in certain systems. Mclk and hclk are also constrained by the frequency of other chips in the system such as SDRAM chips, rather than the speed of the logic on Imagine. Sclk was used to run the SRF and stream controller at half the iclk speed. The relaxed timing constraints significantly reduced the design effort in those blocks and architectural experiments showed that running these units at half-speed would have little impact on overall performance.

The decoupling provided by Imagine's 11 indepenedent clock domains reduces the complexity of the clock distribution problem. Also, non-critical timing violations within one clock domain can be waived without affecting performance of the others. To facilitate these many clock domains, a synchronizing FIFO was used to pass data back and forth between different clock domains. Figure 6 shows the FIFO design used [4]. In this design, synchronization delay is only propagated to the external inputs and outputs when going from the full to non-full state or vice versa, and similarly with the empty to non-empty state. Brute force synchronizers were used to do the synchronization. By making the number of entries in the FIFO large enough, write and read bandwidths are not affected by the FIFO design.

7. Imagine Verification Methodology

Functional verification of the Imagine processor was a challenge given the limited resources available in a university research group. A functional verification test suite was written and run on the behavioral RTL. The same test suite was subsequently run on the structured RTL. Tests in the

suite were categorized either as module-level or chip-level tests. Standard industry tools performed RTL-to-netlist and netlist-to-netlist comparisons for functional equivalency using formal methods.

Module-level tests exercised individual modules in isolation. These tests were used on modules where functionality was well-defined and did not rely on large amounts of complex control interaction with other modules. Module-level tests that exercised specific corner cases were used for testing Imagine's floating-point adder, multiplier, divide-square-root (DSQ) unit, memory controller, and network interface. In each of these units, significant random testing was also used. For example, in the memory controller, large sequences of random memory reads and writes were issued. In addition, square-root functionality in the DSQ unit was tested exhaustively.

Chip-level tests were used to target modules whose control was highly coupled to other parts of the chip and for running portions of real applications. Rather than relying only on end-to-end correctness comparisons in these chip-level tests, a more aggressive comparison methodology was used for these tests. A cycle-accurate C++ simulator had already been written for Imagine. During chip-level tests, a comparison checker verified that the identical writes had occurred to architecturally-visible registers and memory in both the C++ simulator and the RTL model. This technique was very useful due to the large number of architecturally-visible registers on Imagine. Also, since this comparison occurred every cycle, it simplified debugging since any bugs would be seen immediately as a register-write mismatch. A number of chip-level tests were written to target modules such as the stream register file and microcontroller. In order to generate additional test coverage, insertion of random stalls and timing perturbations of some of the control signals were included in nightly regression runs.

In total, there were 24 focused tests, 10 random tests, and 11 application portions run nightly as part of a regression suite. Some focused tests included random timing perturbations. Every night 0.7 million cycles of focused tests, 3.6 million cycles of random tests, and 1.3 million cycles of application portions were run as part of the functional verification test suite on the C++ simulator, the behavioral RTL and the structured RTL. These three simulators ran at 600, 75, and 3 Imagine cycles per second respectively when simulated on a 750 MHz UltrasparcIII processor.

8. Conclusion

The VLSI design and functional verification of the Imagine processor was a challenge with a small team of graduate students. However, the tiled-region design methodology enabled the team to accomplish this task and retain the performance advantages of a datapath-style design in an ASIC tool flow. An exciting area of future work is the extension of the Imagine architecture to a custom methodology or to future VLSI technologies where even greater performance gains could be achieved.

9. Acknowledgments

The authors would like to thank the ASIC division and the San Jose CDC at TI for their support in the design and fabrication of the Imagine processor. We would also like to thank Scott Rixner, John Owens, Mohamed Kilani, Ghazi Ben Amor, and Abelardo Lopez-Lagunas for their contributions to the Imagine VLSI implementation and verification.

References

[1] D. Chen. Apollo II adds power capabilities, speeds VDSM place and route. *Electronics Journal*, page 25, July 1999.

[2] D. Chinnery and K. Keutzer. *Closing the Gap Between ASIC and Custom: Tools and Techniques for High-Performance ASIC Design*. Kluwer Academic Publishers, May 2002.

[3] A. Chowdhary, S. Kale, P. Saripella, N. Sehgal, and R. Gupta. Extraction of functional regularity in datapath circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(9):1279–1296, September 1999.

[4] W. J. Dally and J. Poulton. *Digital Systems Engineering*, pages 485–486. Cambridge University Press, 1998.

[5] K. Diefendorff. The race to point one eight. *Microprocessor Report*, pages 1–5, September 1998.

[6] B. Khailany, W. J. Dally, S. Rixner, U. J. Kapasi, P. Mattson, J. Namkoong, J. D. Owens, B. Towles, and A. Chang. Imagine: Media processing with streams. *IEEE Micro*, pages 35–46, Mar/Apr 2001.

[7] T. Kutzschebauch and L. Stok. Regularity driven logic synthesis. In *Proceedings of the International Conference on Computer Aided Design*, pages 439–446, November 2000.

[8] R. X. Nijssen and C. van Eijk. Regular layout generation of logically optimized datapaths. In *Proceedings of the International Symposium on Physical Design*, pages 42–47, 1997.

[9] M. Rodder, Q. Z. Hong, M. Nandakumar, S. Aur, J. C. Hu, and I.-C. Chen. A sub-0.18 μm gate length CMOS technology for high performance (1.5v) and low power (1.0v). In *International Electron Devices Meeting*, pages 563–566, Dec 1996.

[10] Synopsys. *Design Compiler User Guide*, 2000.11 edition.

[11] Synopsys. *Physical Compiler User Guide*, 2000.11 edition.