Architectural Implications of Brick and Mortar Silicon Manufacturing

Martha Mercaldi Kim* Mojtaba Mehrara† Mark Oskin* Todd Austin†

*Computer Science & Engineering
University of Washington
Seattle, WA 98195
{mercaldi,oskin}@cs.washington.edu

†Electrical Engineering & Computer Science
University of Michigan
Ann Arbor, MI 48109
{mehrara,austin}@umich.edu

ABSTRACT

We introduce a chip fabrication technique called "brick and mortar", in which chips are made from small, pre-fabricated ASIC bricks and bonded in a designer-specified arrangement to an inter-brick communication backbone chip. The goal is to provide a low-overhead method to produce custom chips, yet with performance that tracks an ASIC more closely than an FPGA. This paper examines the architectural design choices in this chip-design system. These choices include the definition of reasonable bricks, both in functionality and size, as well as the communication interconnect that the I/O cap provides. To do this we synthesize candidate bricks, analyze their area and bandwidth demands, and present an architectural design for the inter-brick communication network. We discuss a sample chip design, a 16-way CMP, and analyze the costs and benefits of designing chips with brick and mortar. We find that this method of producing chips incurs only a small performance loss (8%) compared to a fully custom ASIC, which is significantly less than the degradation seen from other low-overhead chip options, such as FPGAs. Finally, we measure the effect that architectural design decisions have on the behavior of the proposed physical brick assembly technique, fluidic self-assembly.

Categories and Subject Descriptors: B.7 Integrated Circuits: Advanced technologies; B.4.3 Input/Output and Data Communications:Interconnections (Subsystems)[Interfaces,Topology]

General Terms: Design, Performance

Keywords: Chip assembly, Design re-use, Interconnect design.

1. INTRODUCTION

Technology scaling has produced a wealth of transistor resources and, largely, commensurate improvements in chip performance. These benefits, however, have come with an ever increasing price tag, due to rising design, engineering, validation, and ASIC initiation costs [8]. The result has been a steady decline in ASIC "starts" [9]. The cycle feeds on itself: fewer starts means fewer customers to amortize the high cost of fabrication facilities, leading to even higher start costs and further declining starts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ISCA'07, June 9–13, 2007, San Diego, California, USA.
Copyright 2007 ACM 978-1-59593-706-3/07/0006 ...$5.00.

To implement a design, engineers typically choose between two options. Either they must face the high fixed costs of ASIC production, and hope to amortize it over a large volume of parts, or they must use an FPGA with low fixed costs but high unit part cost. The trade-offs are not just financial. ASICs provide significant speed (3-4x) and power (up to 12x) savings [27], compared to FPGAs, and the technical demands of certain applications, for instance, cell phones, will demand an ASIC. However, FPGAs offer in-field reprogrammability, which is useful for accommodating changing standards. This drives the need for a manufacturing technology that provides the key advantages of FPGAs – low non-recurring costs, and quick turn-around on designs – coupled with the key advantages of ASICs – low unit cost, high performance and low power.

This paper introduces such a technology, which we call *brick and mortar* silicon. At the heart of this manufacturing technique are two architectural components: *bricks*, which are mass-produced pieces of silicon containing processor cores, memory arrays, small gate arrays, DSPs, FFT engines, and other IP (intellectual property) blocks; and *mortar*, an *I/O cap*, that is a mass-produced silicon substrate. Engineers design products with the brick and mortar process by putting pre-produced bricks of IP into an application-specific layout. This arrangement of bricks is then bonded to the I/O cap that interconnects them.

What differentiates brick and mortar from existing approaches, such as system on chip (SoC), is that bricks and I/O caps are manufactured separately and bonded together using flip-chip techniques. Existing approaches provide IP blocks to engineers as "gateware" netlists. Engineers integrate them into a chip design that is then manufactured. Brick and mortar provides IP to designers as real physical entities – small chips to be assembled into the final product.

Our vision is that bricks are the modern-day analogue of the 7400 series of logic, and the I/O cap is the modern wire-wrap board. Rather than spin custom ASICs for products, engineers could purchase these prefabricated components and bond them together as needed.

The key advantages of brick and mortar chip production stem from mass-production of its constituent parts. Bricks are produced in conventional ASIC processes, and hence brick and mortar chips gain the advantages of an ASIC: low power and high performance. Although they are ASICs, bricks are small, resulting in lower individual design and verification costs. Once designed and verified, they can be produced in bulk and used in a variety of end-user products. All of this reduces the cost of a brick and mortar chip. Brick and mortar chips can be mass-produced, using fluidic self-assembly or another low-cost physical assembly technique.

To make brick and mortar chip production successful, one must carefully design the architecture of the bricks and the I/O cap. The bricks must have appropriate sizes and useful function. Large bricks provide more area for physical connection to the I/O cap and, consequently, more inter-brick bandwidth. Large bricks can integrate more logic and/or memory on a single brick, thereby increasing circuit performance via the decreased intra-brick communication latency. In contrast, small bricks offer more design flexibility and, because they are less specialized, more potential re-use across designs. It is important to find a suitable balance between integration and generality in brick function for this technology.

The I/O cap implements inter-brick communication. It is an active silicon die containing wiring, routing, and logic resources. I/O caps that provide more sophisticated routing capabilities (such as packet networks) free logic space on the bricks. On the other hand, if the I/O cap is too specialized, it cannot be re-used across a variety of brick and mortar chips. Striking the correct balance between logic and wiring efficiency is the question driving the architecture of the I/O cap.

This paper is the first to describe and evaluate the brick and mortar assembly process. One must carefully engineer the architecture of both the bricks and the I/O cap to make this chip production method viable. We present a design study of these components and find that three physical sizes of bricks (0.25mm^2, 1mm^2, 4mm^2) are sufficient to contain the IP blocks we study. Using these bricks and an I/O cap designed for both packet-switched communication and FPGA island-style, configured communication, we show how to build a variety of CMP products. These CMPs can perform as close as 8% to an equivalent design built with a traditional ASIC design process. Finally, we describe how to build brick and mortar chips from a low-cost fluidic self-assembly process. We study how this manufacturing process interacts with the architectural decisions both brick and application chip designers will make. Specifically, we find that designing chip architectures permitting a small amount of slack in brick placement on the I/O cap, can lead to a factor of 10 improvement in the rate of brick and mortar chip production.

The next section describes brick and mortar chip production in more detail. We also foreshadow the quantitative results presented later in this paper with a qualitative discussion summarizing the key advantages of brick and mortar chips. Section 3 presents the architecture of bricks and the I/O cap and motivates those architectural choices through design synthesis results. Section 4 examines how architectural choices affect a sample chip design, a 16-way chip-multiprocessor, quantifying the benefits and costs of brick and mortar with respect to performance. In Section 5 we discuss how to assemble brick and mortar chips, which can be either via robots or via self-assembly. For cost and convenience reasons, we expect mass-produced brick and mortar chips to utilize fluidic self-assembly, and in Section 6 examine in more detail the behavior of the assembly process and how it interacts with the architectural decisions presented in Section 3. Section 7 summarizes related technologies before we conclude.

2. THE POTENTIAL OF BRICK AND MORTAR CHIPS

At the heart of the brick and mortar chip manufacturing process are two architectural components: a *brick* and an *I/O cap*. Bricks are physical pieces of silicon that contain an IP block size component such as a processor, network interface, or small gate array. An I/O cap is another silicon die containing an inter-brick communication infrastructure. A brick and mortar manufactured chip consists of several bricks, arranged into an application specific layout, that are bonded to an I/O cap. Once bricks are bonded to the I/O cap, the cap provides power and clock to the bricks and I/O capabilities that enable bricks to communicate with each other and the outside pins of the chip package. Figure 1 depicts a brick layout and the I/O cap to which the bricks are bonded.

Before delving into the architectural components of brick and mortar chip design, we outline qualitatively the key reasons we are pursuing this line of research. Later sections of this paper will revisit most of these issues with quantitative analysis.

Reduced cost: As already discussed in the introduction, the chief motivation for our research is to produce a low-cost alternative to ASIC chip production. Section 7 describes other technologies with related goals. With brick and mortar, cost reductions come from utilizing mass produced bricks and I/O caps in multiple different chip designs.

Compatible design flow: Today ASIC designers employ significant amounts of existing IP to produce chips. This improves design reliability and saves design time and cost. Brick and mortar is compatible with this design flow, merely moving the IP blocks from design modules, which fit into synthesis tool flows, to physical bricks, which fit into a manufacturing flow.

ASIC-like speed and power: Because most of the logic of a brick and mortar chip exists within a single ASIC component, its performance, in speed and power, will tend closer to an ASIC than an FPGA. Small gate array bricks can implement any small custom logic.

Mixed process integration: As we will show, bricks must to comply with a standard physical and logical interface. They do not, however, have to be built from the same underlying technology. This offers an easy way to mix and match bulk CMOS, SOI, DRAM and other process technologies into the same chip.

Improved yield: Large brick and mortar chips can have a higher yield than large ASICs. The advantage comes from assembling a large chip out of a many smaller components. The smaller the component, the higher the yield. One can test component bricks before assembling them, ensuring only functional bricks are included in any assembly, and resulting in an extremely high overall yield.

These advantages do not come for free, however. Brick and mortar assembly will be viable only if its components are well-architected. This paper presents the results of our architectural analysis. We begin in the next section by designing the brick and I/O components.

3. ARCHITECTURE

We now turn to the task of understanding two central architectural questions the brick and mortar approach poses, namely, "what is a brick?", and "what is an I/O cap?"

3.1 Bricks

There are three important architectural questions to answer about a brick. How do bricks communicate? How large is a brick? What is the appropriate functionality for bricks to provide? To answer these questions we begin by investigating how the physical constraints placed on bricks influence the architectural decisions.

What are the goals and constraints of inter-brick communication? The primary architectural constraint on inter-brick communication is that bricks must communicate with other bricks through the I/O cap. Flip-chip bonding connects the I/O pads of each brick to the I/O pads in the cap. Other studies [19] indicate that each bonding bump requires only $25\mu m$x $25\mu m$in area and can provide at least 2.5Gbps bandwidth.

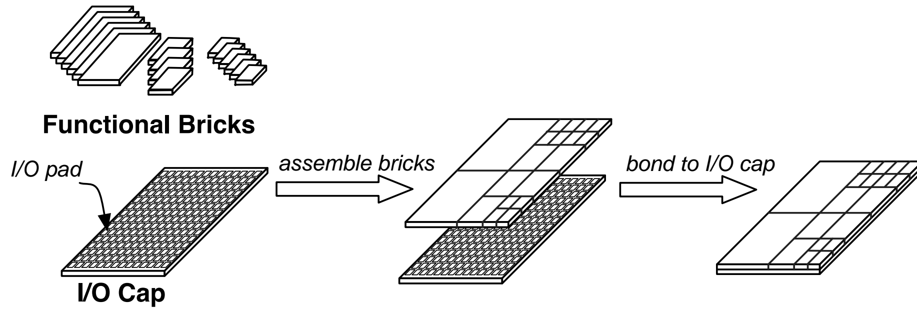

Figure 1: Brick and Mortar Process: With brick and mortar chip design, mass produced ASIC functional *bricks* are assembled in a custom, per-design fashion, and bonded to an ASIC I/O cap providing flexible, high-performance interconnect for bricks to communicate. I/O pads cover the surface of both the bricks and the I/O cap, so that the bricks can communicate when bonded together.

Function	Cite	Circuit Area (um^2)	Max. Circuit Freq. (MHz)	Min. Perf. (Mbps)	0.25 mm^2 brick	1.0 mm^2 brick	4.0 mm^2 brick
					Valid Freq. Range (MHz)		
Small Bricks							
USB 1.1 PHYSICAL LAYER	[34]	2,201	2941	12	**2 - 2941**	No benefit	No benefit
VITERBI	[46]	2,614	1961	-	**N/A - 1961**	No benefit	No benefit
VGA/LCD CONTROLLER	[34]	4,301	1219	-	**N/A - 1046**	N/A -1219	No benefit
WB DMA	[34]	13,684	1163	-	**N/A - 521**	N/A - 1163	No benefit
MEMORY CONTROLLER	[34]	29,338	952	-	**N/A - 843**	N/A - 952	No benefit
TRI MODE ETHERNET	[34]	32,009	893	1000	**125 - 893**	No benefit	No benefit
PCI BRIDGE	[34]	76,905	1042	-	**N/A - 610**	N/A - 1042	No benefit
WB Switch (8 master, 16 slave)	[34]	81,073	1087	-	**N/A - 88**	N/A - 353	N/A - 1087
FPU	[34]	85,250	1515	-	**N/A - 505**	N/A - 1515	No benefit
DES	[34]	85,758	1370	1000	**16 - 1203**	16 - 1370	No benefit
16K SRAM (Singleport)	[7]	195,360	2481	-	**N/A - 2481**	No benefit	No benefit
AHO-CORASIK STR. MATCH	[51]	201,553	2481	-	**N/A - 1331**	N/A - 2481	No benefit
RISC CORE (NO FPU) / 8K CACHE	[34] [7]	219,971	1087	-	**N/A - 1087**	No benefit	No benefit
8K SRAM (Dualport)	[7]	230,580	1988	-	**N/A - 1988**	No benefit	No benefit
Medium Bricks							
TRIPLE DES	[34]	294,075	1282	1000	No space	**16 - 1282**	No benefit
FFT	[45]	390,145	1220	-	No space	**N/A - 1220**	No benefit
JPEG DECODER	[34]	625,457	629	-	No space	**N/A - 629**	No benefit
64K SRAM (Singleport)	[7]	682,336	2315	-	No space	**N/A - 2315**	No benefit
32K SRAM (Dualport)	[7]	733,954	1842	-	No space	**N/A - 1842**	No benefit
RISC CORE + 64K CACHE	[34] [7]	864,017	1087	-	No space	**N/A - 1087**	No benefit
Large Bricks							
256K SRAM (Singleport)	[7]	2,729,344	2315	-	No space	No space	**N/A - 2315**
128K SRAM (Dualport)	[7]	2,935,817	2882	-	No space	No space	**N/A - 2882**
RISC CORE + 256K CACHE	[34] [7]	3,111,025	1087	-	No space	No space	**N/A - 1087**

Table 1: IP Block Synthesis and Brick Assignment: This table shows the synthesis-produced area and timing characteristics of each brick-candidate IP block. Each block has been assigned to the smallest brick which met its area and bandwidth constraints. Note how some of the blocks that we have assigned to small bricks could take advantage of the increased I/O bandwidth afforded by larger bricks (indicated by the increased frequency range).

What are the goals and constraints on brick size? The constrained I/O sets a lower-bound on feasible brick size. Early VLSI engineers observed a phenomenon dubbed "Rent's rule". Rent's rule states that a circuit's required I/O is proportional to its area ($IO \propto Area^{\beta}$). While the precise constants used in the rule change depending upon the type of circuit, the structure of the rule does not [10]. It is important for our purposes, however, that the I/O required by a block of circuitry grows at just above the *square root* of the area. Prior work[10] suggests that $\beta = 0.45$ for processors and memory, and $\beta = 0.6$ for less structured logic. Because the I/O available to a brick grows *linearly* with its area, there must be some minimum brick size, below which the brick area will not be sufficient to support the I/O demands of the circuitry the brick contains.

Bricks will also have a maximum useful size. Rent's rule also means that beyond some larger size, bricks will not be able to utilize all of the I/O available to them. Brick designers should design bricks that use the available I/O, because it is this I/O that connects the fixed, inflexible brick designs in unique ways to produce unique chip designs.

Finally, there can be multiple brick sizes. The more brick sizes offered, the better the area and I/O offering of the bricks can match the true area and I/O requirements of the circuit. We require that the bricks conform to "standard" sizes because it is very difficult to design an I/O cap to interconnect arbitrarily-sized bricks.

What are the goals and constraints on brick functionality? The applications for which we envision using brick and mortar manufacturing are those which currently employ traditional ASICs. For example, wireless transceivers, media encoding/decoding, system-on-chip (SoC) integrations, etc. In these realms, the functional blocks forming a design are fairly large: FFT engines, JPEG compressors, embedded microprocessors.

Below, we address each of these three questions quantitatively, based on synthesis data from candidate brick functions.

Brick size determination To begin assembling a brick family, we used freely available IP cores to produce a "benchmark suite". Starting with Verilog source code from OPENCORES.ORG [34] and other sources of publicly available IP [51, 45, 46], we compiled the designs with the Synopsys DC Ultra design flow[49], targeting the 90nm TSMC [50] ASIC process. We used a commercial memory compiler [7] to generate optimized memory IP blocks.

Based on this data and the constraints outlined above, we conclude that three brick sizes are reasonable: small (0.25mm^2), medium (1.0mm^2) and large (4.0mm^2). Table 1 shows the specifications of the resulting brick assignments. Each brick size offers a fixed I/O bandwidth based on the brick area. In Table 1 we have converted these bandwidth limitations into upper bounds on the brick clock speed. Based on prior work [19] we assume 2.5Gbps per pin. This upper bound is also subject to the speed at which the IP block can operate in a 90nm TSMC standard cell process. When present, the lower frequency bound indicates the minimum speed required to meet application requirements (e.g. an ethernet device must process data at the line rate).

We have organized bricks according to their sizes. We assign IP blocks to the smallest brick size which could meet their area and application bandwidth needs. Note that none of the medium bricks benefits from increasing the brick size, indicating that none of them is I/O constrained. This is a direct effect of Rent's rule. The higher maximum clock frequency at a larger brick size indicates, however, that five of the thirteen small bricks could take significant advantage of the increased I/O bandwidth that a larger brick affords. In these cases, we envision brick builders will do one of two things: (1) provide two different brick sizes, with the smaller brick supporting only lower frequency designs, or (2) more likely, they will redesign the bricks to take advantage of the added area of a larger brick. We did not investigate this aspect of brick design in this study, but one option would be to group blocks of similar functionality (e.g., an ethernet and USB controller on the same "general purpose computing I/O" brick). Another option is to tune buffer sizes on the design. For example, the Aho-Corasik [51] block can use additional buffer space to support more complex matching patterns.

3.2 I/O cap

The I/O cap is a silicon die that has four primary functions: (1) power for the bricks; (2) clocks for the bricks; (3) I/O pads for connectivity to external package pins; and (4) connectivity between bricks. The first three offer little in the way of brick and mortar-specific architectural questions, so we focus on the fourth to drive the I/O cap design. Within this, two key questions are: Given an application space and brick family, what is the best use of the limited number of communication pins into and out of a brick? How do we design a single I/O cap that functions with a variety of brick sizes? To answer these questions we return to our synthesis data.

Because the bricks come in three sizes, and because the particular arrangement of bricks will vary on a per-chip basis, the interconnect in the I/O cap must be both multigranular and flexible. Figure 2 illustrates the two network designs we propose.

Packet-switched interconnect: The first network is a dynamically switched packet network. Panel (a) shows an example brick layout, with an overlay of the logical packet-switched network. Each circle represents a network node. The black nodes are leaf nodes which are valid packet destinations. The nodes represent routers in the interconnect. Within each $4\text{mm}^2$of silicon, the interconnect is a fat-tree, and at the topmost level it is a grid. We coded and synthesized a 64-bit packet implementation of this network using Synopsys DC Ultra. The synthesis results indicated that this network could operate at 800Mhz, with one cycle per hop, and would consume 43% of the I/O cap area. For a 64mm^2I/O cap, the bisection bandwidth of this network is 3.3Tbps.

FPGA-style interconnect fabric: The second interconnect option is an island-style reconfigurable interconnect, shown in Figure 2, panel (b), over which pins in the I/O cap are programmatically connected. Just as with an FPGA, the connecting wires are routed through this mesh. Since wires are constrained at the brick-to-I/O cap interface, we utilize the same physical wires as the packet network and mux them between the two networks dynamically.

As with the packet-switched network, we synthesized a configurable single-bit wiring node. Area results from DC Ultra indicate that such a node requires 155 square microns. Leaving area for remaining logic on the I/O cap (power distribution, clock, pad drivers, etc), we estimate room for approximately 500 switches per small brick. We devote 400 of these switches to a 20 by 20 fully configurable mesh, and another 64 switches to a partially configurable one. Figure 2 illustrates this design. The benefit of this approach is that by enforcing a small amount of standardization on the pin interface, bricks can utilize the mesh to route large 64 bit items to their neighbors. We also retain some flexibility with the 20 fully configurable routes. The bisection bandwidth of this network is 0.26Tbps (fully switchable) and 0.8Tbps (partially configurable).

While the packet-switched network is most useful for routing data between dynamically changing sources and destinations, this mesh is better suited to tightly and statically coupling two bricks, particularly two that are physically near by one another. We utilize this, for example, to bind the FPU to the CPU for one CMP configuration in the next section.

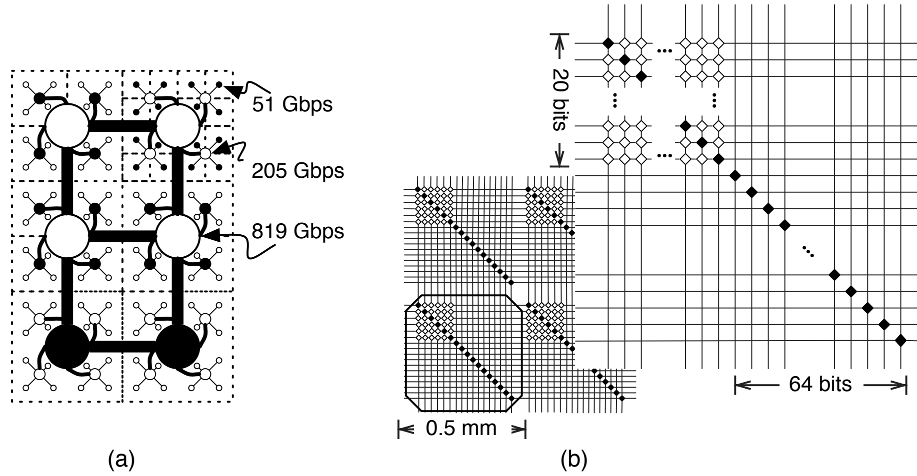

Figure 2: I/O Cap Interconnects: The I/O cap offers two inter-brick interconnects. The first, in panel (a), is a dynamically-routed, packet-switched network. The routers, represented by circles in the figure, are organized into a 4-ary fat tree. The network runs at 800MHz and requires a single cycle hop between routers. The black routers represent valid routing destinations for the example brick layout. The second interconnect in panel (b) is an island-style, statically programmable, mesh interconnect that can connect pins directly to one another. These two interconnects support two different styles of inter-brick communication. The first supports dynamic and variable communication while the second can tightly couple bricks in a fixed pattern.

Chip Multiprocessor Designs						
	CMP-L		CMP-M		CMP-S	
Total Area (mm^2)	193.5		177.5		200.5	
Chip Composition						
	Count	% Area	Count	% Area	Count	% Area
Small Bricks (.5x.5 mm)						
RISC CORE (NO FPU) + 8K CACHE	-	N/A	-	N/A	16	1.99%
FPU	-	N/A	-	N/A	16	1.99%
ETHERNET NIC	1	0.13%	1	0.14%	1	0.12%
MEM CNTL	1	0.13%	1	0.14%	1	0.12%
USB PHYS LAYER	1	0.13%	1	0.14%	1	0.12%
DMA	1	0.13%	1	0.14%	1	0.12%
PCI BRIDGE	1	0.13%	1	0.14%	1	0.12%
VGA/LCD CNTL	1	0.13%	1	0.14%	1	0.12%
Medium Bricks (1x1 mm)						
RISC CORE + 64K CACHE	-	N/A	16	9.01%	-	N/A
Large Bricks (2x2 mm)						
RISC CORE + 256K CACHE	16	33.07%	-	N/A	-	N/A
256K SRAM	32	66.15%	40	90.14%	48	95.29%
Simics/GEMS Performance Simulation						
	Brick & Mortar	*ASIC*	*Brick & Mortar*	*ASIC*	*Brick & Mortar*	*ASIC*
Number of Cores	16	16	16	16	16	16
L1 Cache / Core (KB)	256	256	128	128	8	8
L2 Cache Size (MB)	8	8	10	10	12	12
L2 Associativity	4	4	5	5	6	6
L2 Block Size (B)	64	64	64	64	64	64
L2 Set Size (KB)	32	32	32	32	32	32
Processor Cycles to L2	31	22	41	22	50	22
Exe. Time (Avg.)	108%	100%	120%	100%	136%	100%

Table 2: CMP configurations: The following table describes the three CMP configurations used in our study. We have focused on building CMPs from three different size RISC core bricks. CMP-L features high integration with a large brick combining processor and L1 cache. CMP-M integrates a much smaller L1 cache onto a medium brick with the processor, while CMP-S offers only 8K of L1 cache with the processor on a small brick.

4. CASE STUDY

In this section we examine and quantify the costs and benefits of the brick and mortar assembly process. We focus our attention on a design familiar to processor architects: a CMP.

4.1 Methodology

This section relies upon several analysis tools, which we describe below.

CMP simulator: To understand the performance impact of brick and mortar assembly on a CMP design we used the Virtutech Simics [28] simulation framework and GEMS [29] tool set. We specify the exact configurations for the CMP designs in Table 2. We used the Splash2 [44] suite for multithreaded benchmarks.

I/O Cap Model: To gauge the performance of the I/O cap, we developed a simulation model of the synthesized I/O cap from the previous section. We use the latency information from this model to provide appropriate interconnect delays to Simics for the CMP.

4.2 Made-to-order chip multiprocessors

Large customers of computing systems recognize that their application requirements do not always match the one-size-fits all processors available today. For example, network servers need throughput on task-based parallelism, while scientific computing requires extensive floating point capabilities. Thanks to the low overhead associated with starting and producing a run of chips with brick and mortar assembly, it is possible to produce made-to-order CMPs. For this work, we select a basic CMP design, and examine the costs and benefits of fabricating it using brick and mortar.

We examine three different 16-way CMPs, built from the bricks in Section 3, all of which fit within 200mm^2. Each design consists of 16 combined processor and L1 cache bricks. These bricks differ in size (small, medium and large), cache capacity (6K, 64K, and 256K), and whether or not the FPU is on the same brick. After some general-purpose I/O bricks, the L2 cache fills out the remainder of the 200mm^2area budget. Table 2 summarizes the designs.

Performance: Across the three designs we study, three significant factors change. First, the L1 caches are constrained by the choice of brick size to implement the processor. Second the network delay between the L1 and L2 depends on brick layout. Third, the CMP built from smallest RISC core brick (CMP-S) requires an additional FPU brick per processor. The configurable mesh interconnect connects the processor and FPU bricks.

For each brick and mortar design we modeled a corresponding, equally-provisioned ASIC version of the CMP for comparison. Between the ASIC and brick and mortar designs the primary difference is in the component latencies. The brick and mortar inter-brick latencies come from the I/O cap network model, while the ASIC latencies come from the published latencies of the UltraSparc T1 [47] on-chip interconnect. The “Simics/GEMS Performance Simulation” section of Table 2 gives the specifics of each pairwise comparison.

Figure 3 shows the performance results from the simulations. The performance of each brick and mortar chip was normalized to the performance of the corresponding ASIC design. On average, the benchmarks took 8%, 20% and 36% more time to complete on the three brick and mortar CMPs than on the corresponding ASICs. This was due to the increased interconnect latency that the I/O cap introduces. The principle difference between the three CMP designs was the size of the L1 cache in the processor brick. Naturally, the smallest (8KB cache in CMP-S) incurred the most L1 misses while the largest (256KB cache in CMP-L) incurred the fewest. Each L1 miss sends a request to the L2, which on the brick and mortar designs required communication via the I/O cap. Thus, the L1 miss rate in the brick and mortar CMP designs largely determined performance.

5. MANUFACTURING BRICK AND MORTAR CHIPS

Previous sections have described the basic structure of a brick and mortar chip, namely ASIC bricks bonded to an I/O cap. What we have not described is how to arrange bricks in an application-specific layout prior to bonding to the I/O cap. A straightforward approach is to use robotic assembly. Manipulating and aligning devices at the $\approx 1\mu m$scale that is required for brick and mortar assembly is a solved mechanical engineering problem. Commercial systems already exist for manipulating devices at this scale [59]. The drawback of these devices is cost. For low-volume manufacturing this cost might be manageable. However for larger volume manufacturing a lower-cost alternative is preferable. Fortunately, there is one.

Bricks can also be assembled via fluidic self-assembly (FSA) [57]. As shown in Figure 4, FSA begins with one assembly template per brick size. Bricks begin loosely and randomly placed en masse on this substrate. The substrate is then agitated (shaken), causing the bricks to move around. The assembly substrate is pitted with brick-sized holes into which the bricks fall as they move about. Eventually all holes on the substrate will fill, completing the arrangement of bricks. Previous work [22] has demonstrated this technique using devices that are approximately the same size as the bricks in Section 3.

Assembly of a brick and mortar chip with FSA requires a handful of additional techniques. First, the assembly template is an active silicon device. It therefore makes sense to design a template that can be used in manufacturing a variety of chips; i.e., one that is not application-specific. Second, our architectural study (Section 3) indicates that multiple brick sizes are useful to match the area and bandwidth offerings of a brick to the area and bandwidth demands to the functional logic it contains. Unfortunately, a single FSA process cannot assemble bricks of differing sizes, because different brick sizes require different agitation forces. Simply running at the maximum agitation force will work fine for the larger bricks, but the small bricks will never settle onto the substrate. This means that the assembly process would require one (or more) separate FSA substrates per brick size. The final technique is to assemble bricks of the same size, but different functionality together. This requires a method of controlling the bonding of a particular brick type to a particular site on the assembly substrate.

We propose the following process for mass-production of brick and mortar chips.

Brick modification: We modify the basic architecture of a brick to include a simple AC coupled power and communication device that is capable of transmitting a unique identification tag. This tag identifies the type of the brick to the substrate. This technology is well-established and in wide use in RFID tags [16, 20, 31]. A simple version is suitable here, as the communication of the identification tag need travel only a few micrometers, instead of a few feet.

Template assembly: The specific FSA process on which we base our experiments is a semi-dry one [57, 36, 40, 41]. In this process, bricks of the same size are mixed with a small amount of water and poured over an assembly substrate with correspondingly-sized bonding sites. Parts arrive at binding sites by random motion, thanks to the shaking of the substrate. When a brick nears a binding site, there are three forces that determine how it falls into the site. First, bricks are not pure squares, but rather shaped such that they

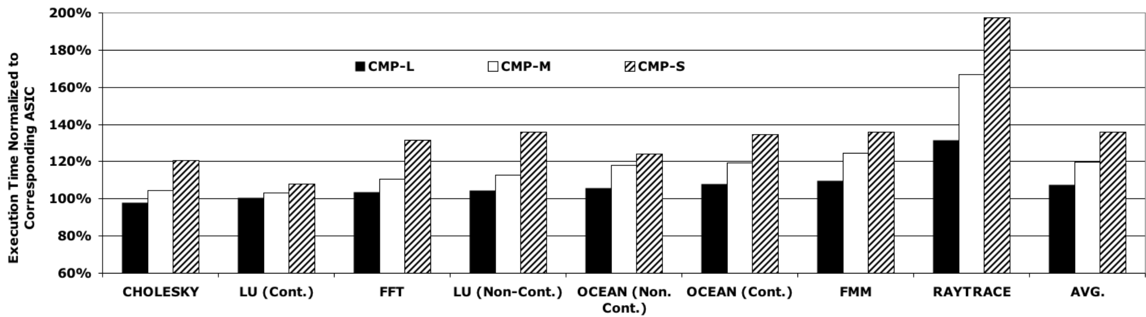

Figure 3: Application Runtime on Brick and Mortar CMP v. ASIC CMP: This graph shows the execution time of multithreaded Splash2 workloads running on a brick and mortar CMP relative to their execution time on an equally-provisioned ASIC CMP. The larger the L1 cache in the CMP design, the less L2 traffic arose, resulting in less inter-brick communication and thus less slowdown relative to an ASIC.

can fit into binding sites only when aligned properly [13, 22]. Second, they fall due to gravity. Third, capillary forces assist gravity. This sort of capillary force-driven self-assembly [48] relies on the minimization of interfacial energy.

The substrate on which bricks are assembled is coated with a polymer pNIPAM (poly-N-isopropylacrylamide [25, 35, 26, 55]) which can reversibly switch between hydrophobic and hydrophilic states, allowing bricks to be attracted to or repelled from the assembly surface. During assembly, the substrate functions as an active silicon die. As bricks fall into binding sites, it queries the brick, using the AC-coupled communication device, to determine the brick's type. The substrate can then reject bricks that fall into the wrong location by setting the polymer's state to hydrophobic through a small change in the local temperature of the binding site.

Brick Testing: Bricks, because they are unpackaged, bare silicon dies, are not trivial to test, including functional testing at speed and burn in. Historically, one had either to package a die or bond it to a board in order to perform these tests. Packaging is not an option for a technology such as brick and mortar or multi-chip modules (MCMs), the field which named this particular problem the "known good die" problem. This threatens overall chip yield because an inability to test component parts until after assembly means that a single bad component will ruin a whole assembly.

There are several solutions currently available, including temporary bonding [1], and advanced communication technologies, such as capacitive coupling [56, 42] or more classic interconnect systems [21], which allow power and at-speed communication with a bare die. Any one of these technologies could work to weed out the bad bricks from brick and mortar chip assemblies. We have selected capacitive coupling because it does not require maneuvering the bricks in any way other than what the described assembly system already requires.

Composite assembly: Once a complete template of same-size bricks has been formed, it is robotically lifted off the substrate and placed onto the I/O cap (See Figure 4). It is necessary to perform this step once for each size brick used in the design. It is also possible to perform this step more than once for each size brick if so desired. The following section, Section 6 will outline circumstances under which this would be desirable. This process does not completely eliminate the need for high-precision robotic assembly. This last, composite assembly step still requires a mechanical device. However, it replaces the lion's share of the work, i.e., brick assembly, with the low-tech, and inexpensive FSA processes.

6. INTERPLAY OF ARCHITECTURE AND FLUIDIC SELF-ASSEMBLY

An interesting fact of brick and mortar manufacturing is that there is an extremely close interaction between *what* one manufactures and *how* one manufactures. In this section we explore how the architecture of the chip affects and should be affected by the FSA process.

For this study we simulated the FSA process itself. The simulator matches production capabilities of current experimental FSA systems [22]. It models assembly of bricks onto the substrate, programmatic discard if they are the wrong type, and accidental discard (a feature of the FSA process which, in a well-tuned system, happens 5% as often as bricks bind to sites [22]). For this study, we utilize synthetic chip designs to examine assembly times as the number of bricks and number of kinds of bricks in a design varies. We varied the number of kinds of bricks from 1 to 5, as well as the number of bricks in the design.

The data in Figure 5 show that increasing the number of kinds of bricks (their heterogeneity) or the number of bricks required increases the assembly time exponentially. This means that there are strategic decisions to make at each stage of the process, from brick architecture, to chip architecture, to how one makes use of the assembly process to control assembly time. In the sections below we explore the trade-offs in these decisions.

Brick design: The data in Figure 5 indicate that assembly time grows exponentially with the number of kinds of bricks present in an FSA process. For example, one will reap significant gains in assembly time by offering one single, slightly reconfigurable brick instead of two different brick types with largely similar functionality. Brick functions where this might be possible include byte- vs. word-aligned memories and bus interface standards.

Interconnect design: Another interesting interaction between architecture and manufacturing is in the placement of bricks. Specifically, if we can design architectures where the placement of bricks need not be exactly the same for every chip produced, the assembly time will shorten dramatically. The data in Figure 6 indicate that by relaxing constraints on brick placement, the assembly process will complete significantly more quickly. One can relax placement constraints by carvint the grid of bricks into square regions $slackRadius$ bricks on a side. A brick that falls anywhere within a region can satisfy any of the bricks required by that region (provided a brick of that type was indeed required by the region). Strict brick placement is simply a $1x1$ region ($slackRadius = 1$).

Note that the solid lines in Figures 5 and 6 are identical. The 25-brick line in Figure 5 is the same as the $slackRadius = 1$ line in Figure 6, because the data in Figure 6 was collected using

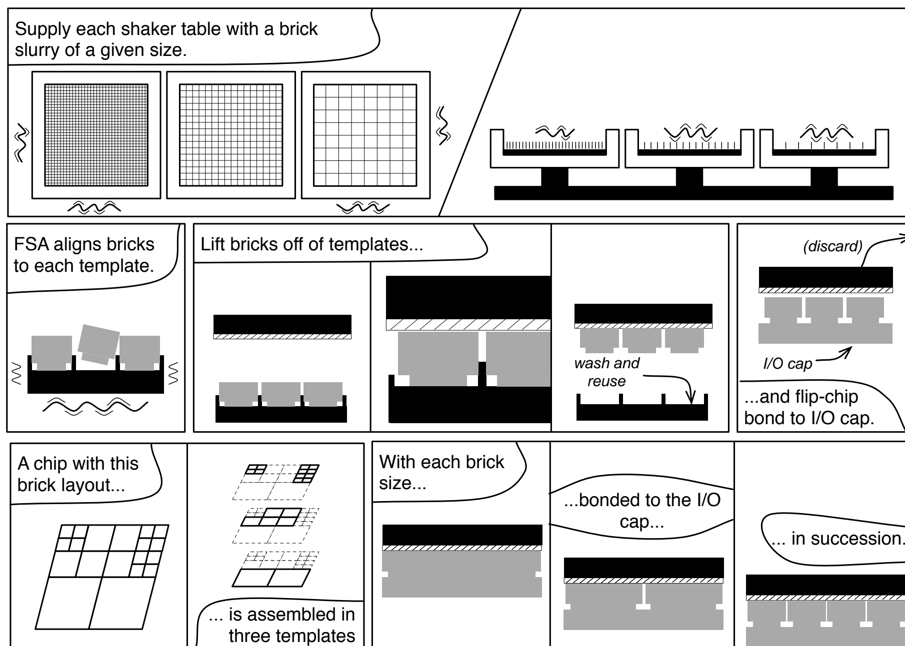

Figure 4: Illustrated Brick and Morter Fluidic Self-Assembly Process: To assemble a brick and mortar chip, we arrange the bricks programmatically using fluidic self-assembly (FSA). Assembly requires one FSA process (and optionally more) per brick size. Assembly occurs on a template with holes matching the bricks in shape and size. The assembly process can select bricks based on the desired type and functionality. Once assembled, bricks are robotically lifted from the templates and bonded to the I/O cap.

25-brick designs and the data in Figure 5 with $slackRadius = 1$. These graphs indicate that one can save approximately the same amount of time introducing a $slackRadius$ of 5 to a 25-brick chip (Figure 6) as by reducing the size of the design from 25 to 9 bricks (Figure 5).

In order to accelerate the assembly process in this way, the communication interconnect must be sufficiently general or programmable to handle the unpredictability in placement. If placement slack is allowed, a single chip design could result in a different brick layout for each chip that is assembled. Therefore the interconnect must be flexible not only on a per-design basis, but a per-chip basis. It is up to the chip designer to determine how much slack to introduce. He or she will have to make a decision based on the tradeoffs between assembly time, desired performance, and the latency tolerance of the design itself.

Assembly management: Thus far we have described the brick and mortar assembly process as one that utilizes different FSA stations to assemble bricks of the same size. These partial-chip assemblies are then robotically assembled onto an I/O cap. However, nothing constrains the use of multiple templates to be used *only* for different size bricks. Given some number of template assembly stations, what mix of bricks should each station assemble?

One way to utilize multiple assembly stations is to spread the heterogeneity of the desired bricks across a range of brick sizes. For example, if a design requires 12 different kinds of bricks, a design consisting of 12 kinds of large bricks will assemble significantly more slowly than a design consisting of 4 small, 4 medium, and 4 large bricks. The type of chip one is designing will determine how much flexibility the designer has in this respect. For example, not all kinds of bricks are likely to have small, medium, and large versions. For the bricks where that option is available, for example with the memory bricks outlined in Section 3, the chip designer will still need to decide how much he or she is willing to adjust the design, perhaps sacrificing some performance, in order to optimize assembly time.

Taken to the limit, the data in Figure 5 indicate that, in theory, one should use a separate self-assembly process for each brick type. In practice, that would require one assembly substrate per brick type. This might not be feasible, as it essentially requires purchasing sufficient equipment for the most heterogeneous chip design one might ever make. Instead, one would conserve equipment and either (1) assemble two brick types (of the same size) at the same time on the same template, or (2) assemble the two brick types in sequence, first the first brick type, then the second, on the same table. The data in Figure 5 indicates that the second option will result in faster assembly.

7. RELATED WORK

There are a handful of technologies that, like brick and mortar, target the gaps between the performance, cost, power and convenience of ASICs and FPGAs. We outline them and their relationship to brick and mortar here.

System on a Package (SoP): SOP is a technology which packages multiple silicon dies together in a single package. Commercially available SoPs that we know of [6] are essentially multi-chip modules (discussed below). However, there are research devices that stack multiple dies [19]. This physical technology could be employed in a brick and mortar system. What distinguishes brick and mortar from these devices is the architectural work. SoPs are a way to lower package costs, but still rely upon users to design and pay for fabrication of the constituent ASICs that are bonded together. Our goal, instead, is to develop a market of pre-fabricated ASIC bricks that interconnect in a standard way.

Multi-chip modules (MCMs): An MCM or multi-chip module consists of multiple silicon integrated circuits which share a single package. MCMs have been in commercial use for over 30 years,

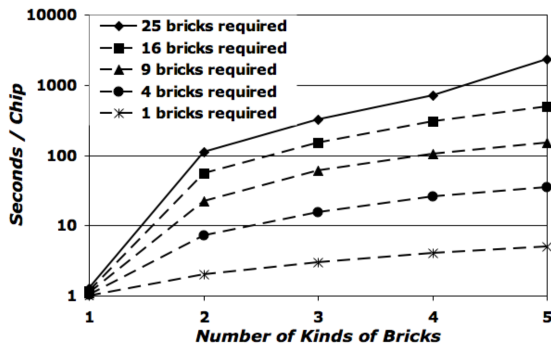

Figure 5: Effect of Design Size on Assembly Time: Assembly time grows exponentially with both the number of bricks to be assembled and the number of types of bricks.

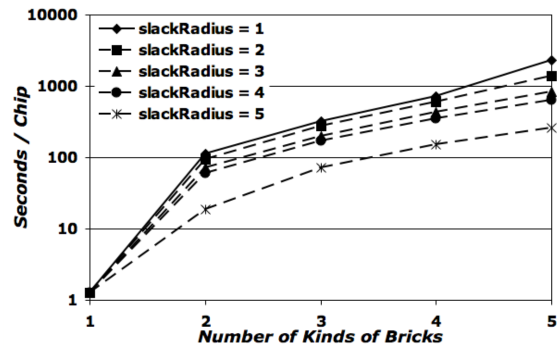

Figure 6: Effect of Slack in Placement on Assembly Time: By allowing some flexibility in the brick placement, one can significantly reduce the time required to assemble a complete set of bricks.

with packages as large as 10cm on a side in use [39]. MCMs amortize the packaging area overhead across multiple components. The closer a pair of communicating chips is, the faster they can transmit signals to each other. Sharing a single package brings these chips closer together. Like SoP technology, MCM chips bear a superficial resemblance to brick and mortar fabrication, but the two technologies have otherwise very different goals.

Systems on a chip (SoC): People build SoCs in a very brick and mortar-like fashion, purchasing IP blocks and integrating them into a single design. However, at the end of the day, an SoC consists of a single custom silicon die, while brick and mortar entails assembling the IP blocks which have already taken the physical, silicon form of bricks.

Structured ASICS: Structured ASICs, also sometimes called platform ASICs, are multi-layer circuits, where the circuitry in the bottom layers is fixed, and only the top couple of layers (typically 2 to 3) is custom[33, 58, 52]. The bottom layers form an array of logic units (i.e., lookup tables, flip-flops). These units are connected as dictated by the designer via custom wiring implemented in the top layers. Implementing a circuit in this way reduces the non-recurring costs of an ASIC, as only the top layers are custom, and thus fewer layers must be designed, verified and have masks built. Furthermore the circuit is largely composed of fixed logic, so if an application maps well onto the array, it will perform better than an FPGA implementation and consume less area, thereby reducing the unit cost. The structured-ASIC market is expected to reach $1.3B by 2010, siphoning off 3.5% of the anticipated $31.4B ASIC market [11]. Structured ASICs are currently commercially available at the 180 and 250 nm nodes[8] through companies such as AMI Semiconductor [5], ChipX [12], eASIC [17], Faraday [23], Fujitsu [24] and NEC [32].

In some sense, brick and mortar is the dual of a structured ASIC. While structured ASIC provides a fixed array of blocks with a custom interconnect on top, brick and mortar offers a custom array of bricks with a non-custom, reconfigurable interconnect to connect them. In addition, brick and mortar's bricks offer larger, more complex functions than the lookup tables and RAMs typical of structured ASICs. Because brick and mortar requires that bricks fit a standard form factor to interact with the I/O cap, some logic area might go to waste. Structured ASICS have a custom interconnect and are therefore not subject to this restriction on the logic blocks.

Coarse-grained reconfigurable devices: In the gap between structured ASICs and FPGAs are a new class of coarse-grained reconfigurable devices. These chips consist of relatively large reconfigurable "objects", which are configurably connected FPGA-style. One startup, MathStar, Inc. [30], recently introduced its second generation Field-Programmable Object Array (FPOA) family called Arrix, which supports 400 individually configured 16-bit objects connected via a 1GHz programmable interconnect. A second startup, CSwitch [15], has announced an architecture consisting of configurable control, compute and switch nodes, connected via a 20-bit wide, 2 GHz interconnect fabric.

In some devices these objects resemble processors. In some cases they are targeted towards a specific class of applications. For example the picoArray from picoChip [37] targets wireless signal processing. In other cases, such as QuickSilver [38], Abric [4], and Cradle Technologies [14], the compute nodes are more general. In still other cases [18], the entire device operates as a single reconfigurable processor.

Certain applications, such as HDTV decoding, map well onto these devices. Applications that map well generally contain significant amounts of traditional data parallelism and operate on word-size chunks of data. Applications that do not map well are those that require specialized bit-level operations and those with specific circuit requirements (e.g. analog to digital converters). These technologies and brick and mortar are actually synergistic. Brick and mortar fabrication can be used to produce coarse-grained, configurable devices. Similar to the comparison to structured ASICs above, brick and mortar offers the opportunity to mix and match both part types and fabrication technologies to produce a wider variety of these coarse-grained parallel technologies.

FPGAs with hard IP cores: For years, FPGA manufacturers have provided complex fixed-logic cores inside their FPGA fabrics. For example, Virtex2Pro [54] provides both fixed multipliers, SRAM blocks, and entire PowerPC cores. Recent products from Xilinx[53] and Altera[3] have specialized further, with specific FPGAs targeting different market segments (e.g., the Xilinx "FX" series targets embedded processing, and the "SX" series aims for signal processing). The advantage to having these cores is that, if a design requires them, they incur little area/delay/power overhead relative to an ASIC. The disadvantage is that the core selection is set by the FPGA manufacturers and the product offerings that are necessarily limited. Brick and mortar's ability to synthesize a variety of complex logic functions cheaply into the same chip is a potential advantage over these domain-specific FPGAs.

Inexpensive Fabrication Techniques and Fluidic Self-Assembly: Self-assembly has been studied extensively for years [57]. Its promise has always been a low-cost alternative for bulk manufacturing that would otherwise require robotic assembly. To the best of our knowledge, only one company has attempted to employ FSA commercially, Alien Technologies [2]. Alien intended to use FSA to bond the antenna to the processing device

for RFID tag production. At this point in time, it is unclear to the authors whether Alien will continue to use FSA, given recent financial reports [43]. To make FSA commercially viable, one must integrate of architectural and manufacturing decisions, as the studies in Section 6 demonstrated.

8. CONCLUSION

This paper introduced brick and mortar chip manufacturing. *Bricks* are the modern day 7400 series logic, consisting of processors, ethernet engines, gate arrays, and other IP-size blocks. The *mortar* is an I/O cap, on which engineers bond bricks to produce the resulting product. It explored the architecture of bricks, determining reasonable design sizes (0.25mm^2, 1mm^2, 4mm^2). It presents an I/O cap design that include both a high-bandwidth packet-switching network and lower-bandwidth, but more flexible and lower-latency, island-style mesh. We explored how to use brick and mortar to assemble made-to-order CMPs and found that such chips perform comparable to fully custom ASIC versions. Finally, we examined the interplay of architecture and manufacturing and found that flexibility in the architecture can be used to dramatically increase manufacturing production rate.

9. ACKNOWLEDGMENTS

This work has been made possible through the generous support of the Gigascale Systems Research Center, an NSF CAREER Award (ACR-0133188), Sloan Research Foundation Award (Oskin), Intel Fellowship (Kim), and support from Intel and Dell. We would like to thank our reviewers, Norm Jouppi, Tim Sherwood, Susan Eggers, Rakesh Kumar, and Steve Swanson for their helpful feedback.

10. REFERENCES

[1] H. O. Alan Barber, Ken Lee. A bare-chip probe for high I/O, high speed testing. 1994.
[2] Alien technology website. www.alientechnology.com.
[3] Altera website. www.altera.com.
[4] Ambric, Inc. website. www.ambric.com.
[5] AMI semiconductor website. www.amis.com.
[6] Amkor technology website. www.amkor.com.
[7] Artisan website. www.artisan.org.
[8] R. Ball. The promise of structured ASIC. Oct 2004. www.electronicsweekly.com.
[9] K. Brown. Economic challenges on the path to 22 nm. June 2004. www.future-fab.com.
[10] R. R. B.S. Landman. On a pin versus block relationship for partitions of logic graphs. *Transactions on Computers*, 1971.
[11] D. Bursky. Arrays narrow platform ASIC, FPGA gap. July 2006. www.eetimes.com.
[12] ChipX website. www.chipx.com.
[13] T. Clark, R. Ferrigno, and et al. Template-directed self-assembly of 10-micron-sized hexagonal plates. 124, 2002.
[14] Cradle technologies website. www.cradle.com.
[15] Cswitch website. www.cswitch.com.
[16] R. Drost, R. Hopkins, and I. Sutherland. Proximity communication. 2003.
[17] eASIC website. www.easic.com.
[18] C. Ebeling, D. C. Cronquist, and P. Franklin. RaPiD - reconfigurable pipelined datapath. In *Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers*, 1996.
[19] J. U. K. et. al. Development of next-generation system-on-package (SOP) technology based on silicon carriers with fine-pitch chip interconnection. 49, 2005.
[20] K. K. et al. 1.27gb/s/pin 3mw/pin wireless superconnect (wsc) interface scheme. 2003.
[21] R. R. F. Agahdel, C. Ho. Known good die: A practical solution. Apr 1993.
[22] J. Fang and K. F. Böhringer. Wafer level packaging based on uniquely orienting self-assembly (the DUO-SPASS processes). 15, 2006.
[23] Faraday electronics website. www.faradayelectronics.com.
[24] Fujitsu website. www.fujitsu.com.
[25] M. Heskins and J. Guillet. Solution properties of poly(N-isopropylacrylamide). A2:1441, 1968.
[26] D. Huber, R. Manginell, M. Samara, B. Kim, and B. Bunker. Programmed adsorption and release of proteins in a microfluidic device. 301, 2003.
[27] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. In *International Symposium on Field Programmable Gate Arrays*, pages 21–30, 2006.
[28] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *Computer*, 35, 2002.
[29] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *Computer Architecture News*, 33, 2005.
[30] Mathstar, Inc. website. www.mathstar.com.
[31] S. Mick, J. Wilson, and P. Franzon. 4Gbps high-density AC coupled interconnection. 2002.
[32] Nec website. www.nec.com.
[33] T. Okamoto, T. Kimoto, and N. Maeda. Design methodology and tools for NEC Electronics' structured ASIC ISSP. In *International Symposium on Physical Design*, 2004.
[34] Opencores.org website. www.opencores.org.
[35] Y. Pan, R. Wesley, R. Luginbuhl, D. Denton, and B. Ratner. Plasma polymerized N-Isopropylacrylamide: Synthesis and characterization of a smart thermally responsive coating. 2, 2001.
[36] J. M. Perkins. Magnetically assisted statistical assembly of III-V heterostructures on silicon: Initial process and technology development. 2002.
[37] picochip website. www.picochip.com.
[38] Quicksilver technology website. www.qstech.com.
[39] J. T. R. Kalla, Sinharoy Balaram. IBM Power5 chip: a dual-core multithreaded processor. *Micro*, Mar-Apr 2004.
[40] J. Rumpler. Optoelectronic integration using the magnetically assisted statistical assembly technique: Initial magnetic characterization and process development. 2002.
[41] J. Rumpler, M. Perkins, and et al. Optoelectronic integration using statistical assembly and magnetic retention of heterostructure pills. 2, 2004.
[42] D. Saltzman and J. Knight, T. Capacitive coupling solves the known good die problem. 1994.
[43] E. Schuman. Rfid bellwether alien again postpones its ipo. July 2006. www.eweek.com.
[44] J. P. Singh, W.-D. Weber, and A. Gupta. SPLASH: Stanford Parallel Applications for Shared-Memory. *Computer Architecture News*, 1992.
[45] Software/hardware generation for dsp algorithms. ttp://www.spiral.net.
[46] http://www-vlsi.stanford.edu/ee272/proj99/babyviterbi/verilogcode.html.
[47] Sun ultrasparc-t1. http://www.sun.com/processors/UltraSPARC-T1/.
[48] R. R. A. Syms, E. M. Yeatman, and et al. Surface tension-powered self-assembly of microstrutures: The state-of-the-art. 12, 2003.
[49] Synopsys website. http://www.synopsys.com.
[50] TSMC 90nm technology platform. http://www.tsmc.com/ download/english/a05_literature/90nm_Brochure.pdf.
[51] N. Tuck, T. Sherwood, B. Calder, and G. Varghese. Deterministic memory-efficient string matching algorithms for intrusion detection. In *Infocom Conference*, 2004.
[52] K.-C. Wu and Y.-W. Tsai. Structured ASIC: evolution or revolution? In *International Symposium on Physical Design*, pages 103–106, 2004.
[53] Xilinx website. www.xilinx.com.
[54] I. Xilinx. Virtex-II Pro and Virtex-II Pro X FPGA user guide. www.xilinx.com.
[55] X. Xiong, Y. Hanein, J. Fang, Y. Wang, W. Wang, D. Schwartz, and K. Böhringer. Controlled multibatch self-assembly of microdevices. 12, 2003.
[56] Z. Yang, K.-T. Cheng, and K. Tai. A new bare die test methodology. *VLSI Test Symposium*, 00, 1999.
[57] H.-J. J. Yeh and J. S. Smith. Fluidic assembly for the integration of GaAs light-emitting diodes on Si substrates. 6, 1994.
[58] B. Zahiri. Structured ASICs: Opportunities and challenges. *International Conference on Computer Design*, 2003.
[59] Zyvex nanotechnology website. http://www.zyvex.com.