Architecture and Design Flow for a Highly Efficient Structured ASIC

Man-Ho Ho, *Member, IEEE*, Yan-Qing Ai, *Student Member, IEEE*, Thomas Chun-Pong Chau, *Student Member, IEEE*, Steve C. L. Yuen, Chiu-Sing Choy, *Senior Member, IEEE*, Philip H. W. Leong, *Senior Member, IEEE*, and Kong-Pang Pun, *Senior Member, IEEE*

***Abstract*— As fabrication process technology continues to advance, mask set costs have become prohibitively expensive. Structured application specific integrated circuits (sASICs) offer a middle ground in price and performance between ASICs and field-programmable gate arrays (FPGAs) by sharing masks across different designs. In this paper, two sASIC architectures are proposed, the first being based on three-input lookup-tables, and the second on AOI22 gates. The sASICs are programmed using a standard-cell compatible design flow. They are customized using a minimum of three masks, i.e., two metals and one via. The area and delay of the sASIC are compared with ASICs and FPGAs. Results over a set of benchmark circuits show that our AOI22-based sASIC had an average of 1.76x/1.41x increase in area/delay compared to ASICs, a considerable improvement compared with the 26.56x/5.09x increase for FPGAs. This is, to the best of our knowledge, the best performance reported in the literature for a practical sASIC. A prototype using the sASIC was fabricated using a universal machine control 0.13-μm mixed-mode/RF process. It was fully verified using scan and functional tests, and used in a demonstration system.**

***Index Terms*— Application-specific integrated circuit (ASIC), area-delay comparison, field-programmable gate array (FPGA), structured-ASIC (sASIC), via-programmable.**

I. Introduction

SINCE the invention of the semiconductor fabrication process over 40 years ago, the feature size of the photolithography process has shrunk from over 10 μm to the current 22 nm. The photolithographic mask layers required have also grown from about 25 for a 0.25-μm process, to approximately 36 for a 40-nm process [1]. As reported in [2], a 45-nm mask set costs about \$900K, and a 32/28-nm mask set approximately \$2M. IBM reports that the number of spins needed to produce an operational-integrated circuit has also increased to approximately 2.5 [3]. Increased mask costs translate to a corresponding rise in the NRE cost, making low to midvolume production unaffordable. This has reduced the number of design starts in the industry [4].

Field-programmable gate array (FPGA) devices provide an alternative solution to the NRE cost problem. These devices allow circuit functionality to be reconfigured after fabrication, and the mask cost is amortized over all FPGA users. However, FPGAs are not suitable for all mass production applications since they suffer from poor area, power and speed compared with application-specific integrated circuits (ASICs) [5]–[7]. This performance gap is quantified in [8], which found that, over a particular set of benchmarks, FPGAs require on average 35 times more area, 14 times more power and have 4 times longer critical path delay compared with ASICs.

Structured ASICs (sASICs) were introduced to bridge the expanding gap between ASICs and FPGAs. They consist of a repeating, regular logic fabric. A design is implemented through customization of one or more metal/via layers, while the remaining layers are shared among different designs. It has been reported that sASICs can cut down NRE cost by as much as 90% and time to market by 4–6 months [6]. The layout of the fabric can also be fine-tuned to better exploit design rule limits or to optimize for manufacturability [9], [10]. It was estimated that metal-programmable sASICs can be 2x–10x cheaper than cell-based ICs [11].

Commercial vendors have diverse approaches for sASIC design [7], [12]–[15]. Unfortunately, implementation details for these commercial solutions are proprietary and performance comparisons are limited in scope. Even when comparisons are available [7], they are over a single circuit and many details are not disclosed.

Several academic groups have proposed and studied different lookup table (LUT) circuits and their sizing for use in sASIC logic block [16]–[18]. Heterogeneous logic blocks [5], [19]–[21], dynamic PLA [22], pass-transistor style [23], and static CMOS styles of different patterns were also proposed [10], [24]–[27]. Few of these works have taken into account practical implementation issues, such as antenna design rules, ESD protection, etc. Moreover, most of them used custom EDA tools rather than industry-standard tools, limiting their appeal for practical applications.

Manuscript received August 13, 2011; revised January 11, 2012; accepted February 20, 2012. Date of publication April 3, 2012; date of current version February 20, 2013. This work was supported in part by the Innovation and Technology Fund, the HKSAR under Project GHP/028/07SZ, and the Hong Kong Applied Science and Technology Research Institute Team, Hong Kong.

M.-H. Ho, Y.-Q. Ai, C.-S. Choy, and K.-P. Pun are with the Department of Electronic Engineering, Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: mhho@ee.cuhk.edu.hk; cschoy@ee.cuhk.edu.hk; yqai@ee.cuhk.edu.hk; kppun@ee.cuhk.edu.hk).

T. C.-P. Chau is with the Department of Computing, Imperial College London, London SW7 2BT, U.K. (e-mail: cpc10@imperial.ac.uk).

S. C. L. Yuen is with the Electrical and Mechanical Services Department, Government of the Hong Kong Special Administrative Region, Hong Kong (e-mail: clyuen@gmail.com).

P. H. W. Leong is with the School of Electrical and Information Engineering, University of Sydney, Sydney 2071, Australia (e-mail: philip.leong@sydney.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2012.2190478

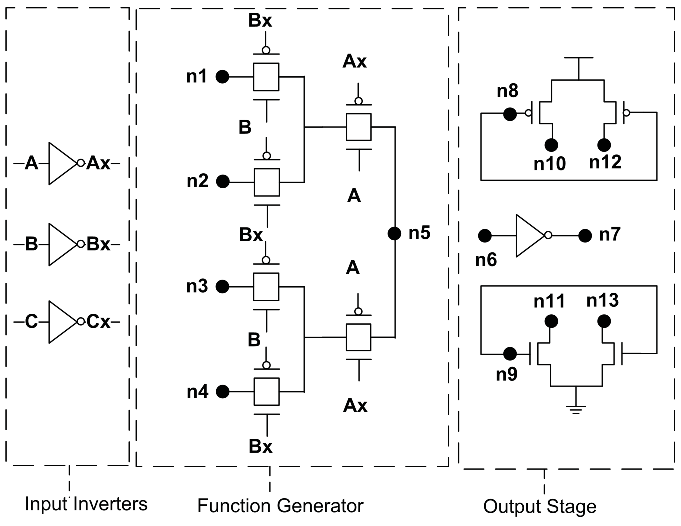

Fig. 1. Schematic of the 3-LUT-based logic block.

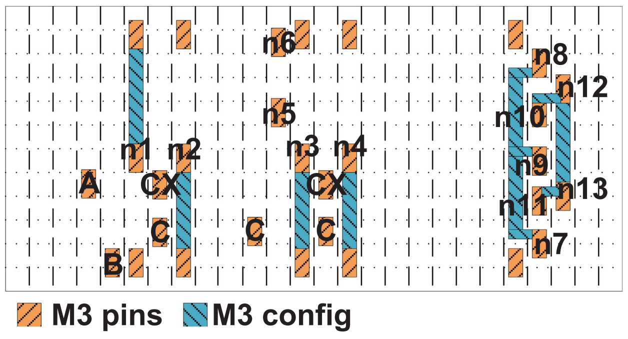

Fig. 2. Occupied tracks and pins of 3-LUT-based logic block.

In this paper, two sASIC architectures, one based on a three-input LUT, and the other on an AOI22-based (AND-OR-INVERT) logic block are proposed. Based on the former, a fully standard-cell compatible design flow was developed. The latter achieved significant improvements on area and delay by exploring a finer granularity in the logic block design. The proposed sASICs are programmed by customizing a minimum of one via mask and two metal masks for both logic configuration and routing. Additional layers can be used for designs with tough routing and/or performance requirements.

The rest of this paper is organized as follows. Section II details our proposed sASIC architectures and logic block designs. The standard-cell ASIC compatible design flow is explained in Section III. Section IV gives a comprehensive comparison on the performance of the proposed sASICs against standard-cell ASIC and FPGA. A prototype of a real-world sample design working in a running system is demonstrated in Section V, and finally, we conclude our works in Section VI.

II. Architecture

For this paper, the universal machine control (UMC) 0.13-μm 1P8M2T mixed-mode/RF process was used. The FSC0H_D standard-cell library, from Faraday, was used for parts of our sASIC design as well as ASIC comparison purposes. A total of 415 cells are available in the library, and the cell height is 3.2 μm, equal to eight horizontal routing tracks.

A. LUT-Based Logic Block

Our initial logic block design was LUT-based, since they have been extensively studied and are functionally complete. We studied three via-programmable LUT circuits for use in an sASIC and concluded that the transmission-gate (TG) style LUT provides a good balance between area and performance [17].

The three-input LUT employed in our sASIC is illustrated in Fig. 1. It can be divided into three parts, i.e., the input inverters, the function generator, and the output stage. The three input inverters provide the optional logic inversion for input pins.

The function generator is based on a 4-to-1 Mux. First, consider such a Mux with the four inputs (n1-n4 in Fig. 1) each tied to logic-high or logic-low, and the two select pins (A, B in Fig. 1) used as inputs. This implements a 2-LUT. A 3-LUT can be implemented by allowing n1–n4 to be configured as either high, low, "C," or its complement "CX." It can perform all three-input binary functions.

This function generator reduces the number of transistors for a 3-LUT from 28 for the standard implementation using an 8-to-1 Mux, to 12, a 57% reduction in transistor count. Increasing the LUT size to more than three inputs impacts delay since signals would need to pass through extra transmission gates. Other research has reached the same conclusion that a 3-LUT [18] implemented using TGs [19] provide a good balance in terms of area, delay and power.

An "output stage" is included in the logic block to provide the gate sizing capability. Drive strengths of 0.6x, 1x, and 2x are implemented.

Apart from three input combinational logic functions, numerous other circuit elements can also be implemented using our 3-LUT logic block in different configurations. These include tri-state buffers, tri-state inverters, filler cells, D-latch, clock-gating cells, and D-flip-flops. However, as flip-flops are time critical and require additional functionality, dedicated cells were used and are described in Section II-C.

The layout of the block measures 10.4 μm by 4.8 μm. As the wire pitch for the process is 0.4 μm, the logic block spans 26 vertical routing tracks, and is 12 tracks high. Fig. 2 shows the pins and metal configuration sites of the logic block. Metal-3 and metal-4 are run in vertical and horizontal directions, respectively. The configuration sites are all in metal-3, leaving all of metal-4 for routing. In metal-3, 13 out of the 26 routing tracks, i.e., 50%, are occupied by IO pins or configuration sites.

B. AOI22 Gate-Based Logic Block

A phenomenon we observed with the 3-LUT library is that the standard-cell synthesis tool does not utilize all of the logic functions that a 3-LUT can provide. FPGA tools may be able

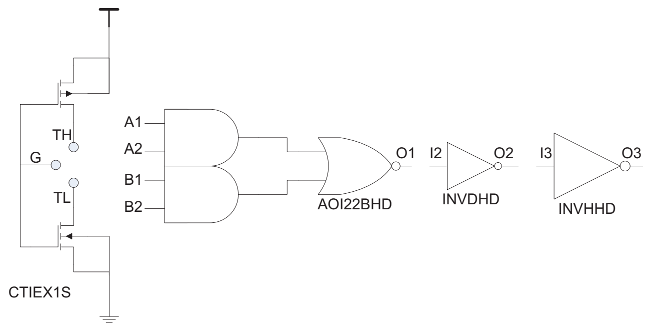

Fig. 3. Schematic of the AOI22-based logic block.

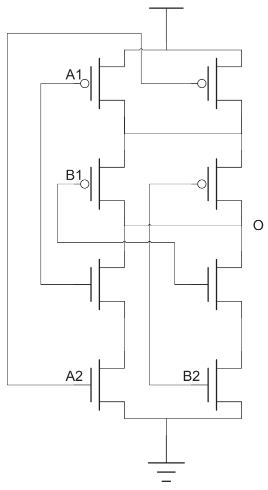

Fig. 4. Schematic of the AOI22 cell.

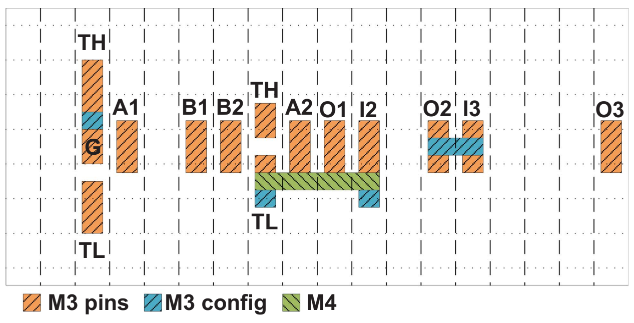

Fig. 5. Occupied tracks and pins of AOI22-based logic block.

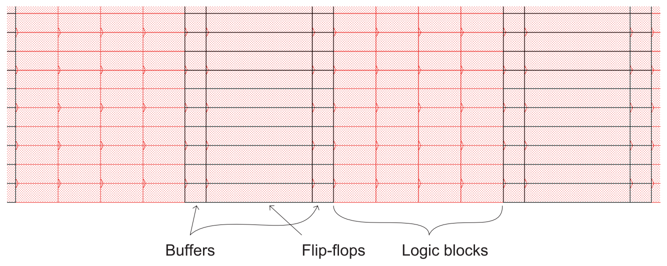

Fig. 6. Fabric with logic/buffer/flip-flop ratio 4:2:1.

to better utilize the logic flexibility but cannot be used for other tasks, such as gate sizing and buffer insertion.

With these considerations, we designed a logic block based on the AOI22 logic gate, which implements a four-input "AND-OR-INVERT" function. The schematic of a single logic block is shown in Fig. 3. The logic block contains a custom-layout configurable tie-hi/lo cell, an AOI22 gate, and two inverters of different driving strengths.

The tie cell is modified from the standard-cell library version so that either logic-high or logic-low can be provided with different connections of the three nodes. When the gate node is connected to the drain of the pMOS, the drain of nMOS provides a tie-lo output, and vice versa. The inverters INVDHD and INVHHD are of 0.8x and 2x driving strength, while the AOI22BHD cell is of 0.4x driving strength. These were chosen so that, for all logic functions that require at most one inverter in the input or output of the AOI22 gate, two driving strengths can be provided. Standard-cell library components were used to implement the gates.

The AOI22 gate is logically equivalent to two AND gates connected to a single NOR gate as shown in Fig. 3. Its implementation is a four-input logic gate with eight transistors, and its schematic is given in Fig. 4. We chose this gate since two-input NAND and NOR gates are extensively used in our benchmark circuits, even when 3-LUTs are available. If we tie the A1, A2 inputs of the AOI22 cell to logic-lo, the cell is a two-input NAND gate, with B1, B2 as inputs. If we tie the A2, B2 inputs to logic-hi, we get a two-input NOR gate. Under these configurations, the AOI22 gate has slightly improved the speed and significantly less area than a 3-LUT. Complex logic functions such as 2/4-input MUXes, can be implemented with less than four AOI22 logic blocks. A total of 17 logic functions were implemented in the cell library, including half-adder, full-adder, MUX2, MUX4, etc. Other special cells like delay cells and filler cells are also provided.

Layout of the block measures 7.2 μm by 3.2 μm, which is 30% narrower than the 10.4-μm wide 3-LUT block. The logic block is 18 routing tracks wide, with 11 of them assigned to the four-input pins, the output pin, and the configuration sites, as shown in Fig. 5. This leaves 39% of the vertical routing tracks on metal-3 free for routing. Depending on the function to be implemented, up to three out of the eight horizontal tracks on metal-4 can be used.

C. Fabric Organization

Similar to standard-cell ASICs, the sASIC fabric in our paper is row based, Fig. 6 shows the organization of the fabric. Each row consists of a repeating pattern of a logic block, followed by a D-flip-flop, and a number of inverters/buffers. Rows are flipped and abutted to form the entire programmable fabric, resulting in columns of flip-flops and buffers between a sea of logic blocks. In addition, an arbitrary number of hard-macro blocks, placed in any location, are supported.

The DFZRSBHHD D-flip-flop from the standard-cell library was used. This has a drive strength of two and incorporates asynchronous set and reset, scan functionalities, and has dual outputs. Each of these flip-flops has an additional tie-high, tie-low, and antenna cell, which can be connected to unused flip-flop pins. Antenna cells are placed in advance because inserting them after routing would require changes to lower layer masks.

For our initial 3-LUT fabrics, buffers were implemented using the logic blocks. This was later improved by including

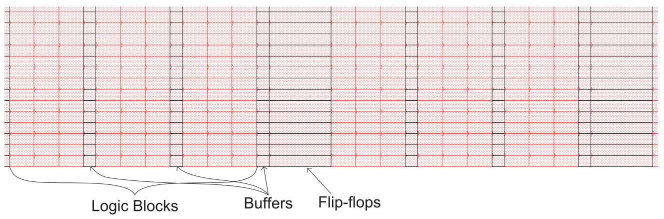

Fig. 7. Fabric with logic/buffer/flip-flop ratio 9:3:1.

dedicated buffers for the better timing performance. Dedicated buffers were used in all AOI22-based fabrics. The buffer cell chosen was BUFKHD from the standard-cell library, which is of driving strength four.

The ratio between number of logic blocks, flip-flops, and dedicated buffers is adjustable in the backend steps, allowing the fabric to be tuned for different classes of designs. This is demonstrated in Figs. 6 and 7. In Fig. 6, the ratio between logic blocks, dedicated buffers, and flip-flops was set to 4:2:1, and it is suitable for register-intensive designs. Buffer columns were placed directly adjacent to flip-flop columns. In Fig. 7, the ratio was set to 9:3:1, for logic-intensive designs. Buffer columns were evenly distributed across the sea of logic blocks, to minimize the performance degradation associated with legalizing the inserted buffers to their feasible sites.

III. EDA Design Flow

A. Library Preparation and Design Synthesis

For the 3-LUT cell library, all 256 possible functions were enumerated. The layout of each was generated by scripts written in the Cadence SKILL language, under the ICFB environment. Each function uses the same layout, with different configuration patterns for the metal-3 programming layer. Parasitic extraction was done using Cadence QRC and Cadence encounter library characterizer was used to produce timing libraries for the 3-LUT cells in the Liberty library format.

For the AOI22-based cell library, we treated each logic block cell as a hard-macro. Layouts of standard-cells were instantiated using Cadence Virtuoso, where fixed and programmable interconnects were manually drawn. Parasitic extraction was set to only consider interconnects between standard-cells. Synopsys PrimeTime was used to produce timing libraries at different process corners.

Design synthesis can be done using any synthesis tool supporting the Liberty library format. In our case, Synopsys design compiler was used.

B. Fabric Creation and Design Mapping Flows

The backend flow is a modified ASIC flow. We divided the backend flow into two sub-flows, one for creating a reusable fabric and one for mapping specific designs onto a fabric. In Fig. 8, the left is used for building a fabric, predefining legal sites for flip-flops and hard-macros, such as block memories. Power planning for the fabric is also done in this step.

The input netlist contains only the hard macros to be prebuilt on the fabric. The fabric designer must provide the size of the fabric and manually define the legal sites for hard-macros. Our script also takes as input, the ratio for the number of different types of "constrained elements" in this step. These refer to cells in our library, which have restricted placement sites, and their placement/legalization needs to be done with our custom tool scripts, e.g., flip-flops. Based on the category of circuits to be mapped on the fabric, the ratio between logic blocks and flip-flops can be customized. The flow also supports a second type of constrained element, which could, for example, be a dedicated cell or buffer.

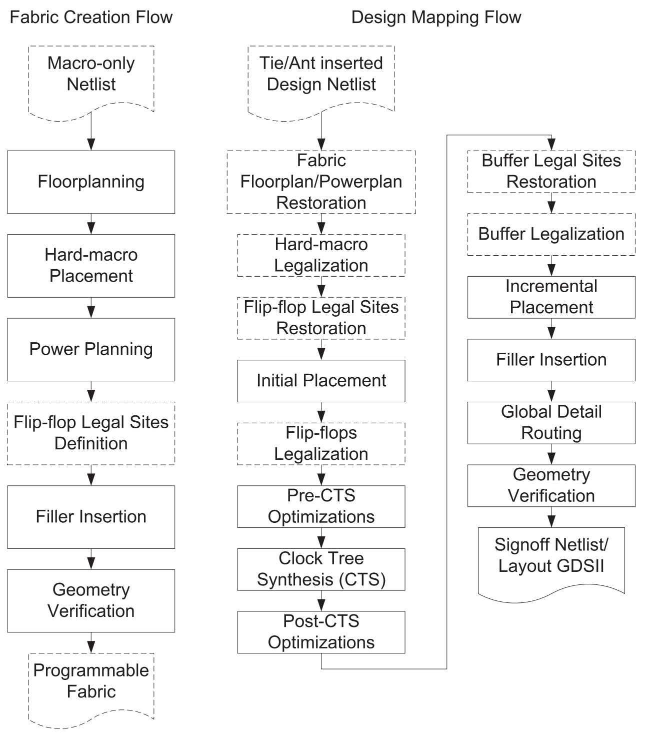

Fig. 8. Fabric creation and design mapping flows.

Although, the current flow only supports two kinds of constrained elements, it is possible to support an arbitrary number of them with small modifications to the current tool.

Following fabric creation, a design mapping flow is run to implement specific designs on the fabric, as shown on the right in Fig. 8. Steps in dashed boxes are unique to our sASIC, while the remaining steps are standard ASIC steps. The first step is to read in the power connections and hard-macro sites defined for the fabric. During implementation, hard-macros are manually placed near the desired sites, and our tool will legalize them to the exact co-ordinate required. In many cases, the design to be implemented may not fully utilize all these legal sites and, as such, the legalize script automatically fills the unused sites with "macro fillers," where the input pins of the macros are tied to logic-hi/lo.

Fig. 9 shows the layout of a sASIC with mapped sample design. Only one of the five memory blocks is in use and extra rows of flip-flops were placed directly above and below the hard-macros, since more are required near the inputs and outputs of hard-macros.

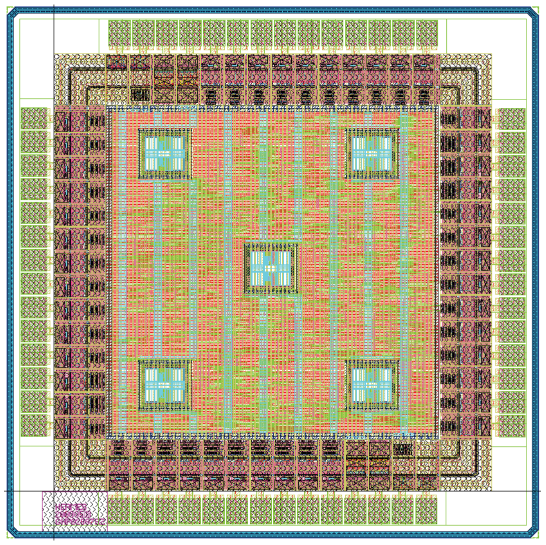

Fig. 9. Design mapped on a fabric with spare hard-macro sites.

Before placement, the legal sites for flip-flops are restored. Logic cell placement is similar to that for standard-cell designs, but the floorplan now includes predefined sites for the constrained elements. These are implemented by adding obstructs on the user-defined legal sites, blocking logic cells from being placed on them. Constrained elements are made floating during logic cell placement so that they do not occupy area and block logic cells. The constrained elements are later legalized back into these sites by our custom tool script. We found that restoring the legal sites for dedicated buffers altogether in this step would greatly reduce the available free space for buffer insertion during the optimization steps. Thus, this is handled in later steps.

After the logic cell placement, we legalize floating flip-flops into their predefined feasible sites. A greedy algorithm is applied to move the flip-flop under consideration to the nearest legal site. After our algorithm places all flip-flops, optimization steps are executed to counter the effects of flip-flop legalization, as this can move the cells in-front or behind the flip-flop on its timing path. Flip-flops are fixed after this to prevent optimization steps from moving them.

After these processes, pre-clock tree synthesis (CTS) optimization and in-placement optimization are performed, followed by CTS and post-CTS optimizations. Since buffer insertions have all been done by now, legal sites for dedicated buffers can be restored. A custom script is used to legalize buffers as was the case for flip-flops. The script does not consider logic cells that possibly overlap these sites since logic cells were placed earlier. As such, we need to run an incremental placement to insure that all logic cells are moved out of any illegal sites, e.g., buffer sites. Afterwards, a custom "cell legalize" is run to move cells onto the correct placement grid.

After placement, we can proceed with global detailed routing. At a minimum, the router requires that metal-3 and metal-4 layers are available. Additional layers can also be used, when routing requirements cannot be met and a compromise between NRE cost and design area must be made.

By this step, the design mapping flow can be considered finished. We then verify the design against the design rules and report the timing results. The flow just described is highly compatible with a typical ASIC's backend design flow, and can be operated at the users' site. This allows proven, industrial quality tools to be used and removes the need for dedicated sASIC CAD tools. Moreover, compatibility with standard tools allow users to do further verification after physical design. Although all the scripts developed were for the Cadence encounter platform, our tools could be ported to other platforms.

IV. Experimental Results

To evaluate the performance of our sASIC architectures, a total of nine benchmark circuits were implemented, respectively, on ASIC, FPGA, and the two proposed sASIC architectures. These chosen circuits are the larger ones from the IWLS 2005 benchmarks [28] collection.

As mentioned earlier, standard cells used within the sASIC and for comparison purposes were taken from Faraday's FSC0H_D generic-core cell library and the UMC 0.13 μm 1P8M2T mixed-mode/RF fabrication process was targeted. For designs requiring block memories, e.g., vga_lcd and ethernet, Faraday's memory compiler was used to generate libraries of block memory IP cores for both ASIC and sASIC designs. The block memory occupies 0.042 mm^2 of the core area in the "ethernet" design, and 0.096 mm^2 of the "vga_lcd" design.

Synopsys design compiler D-2010.03-SP1 was used for design synthesis. We adopted the method of Kuon *et al.* [8]. to compare FPGA and ASIC performance. The desired clock rate was set to an unattainable 2 GHz during a first round synthesis, and the resulting frequency obtained was used during a second round of compilation from which a maximum clock frequency was recorded. Typical case libraries were used during synthesis.

For FPGAs, the flow targeted a Xilinx Virtex-II XC2V3000 device [29], speed grade -6, which is fabricated in a 0.12-μm transistor/0.15 μm 8-metal-layer process. This device is selected since it was manufactured in a comparable process node, allowing a fair area comparison to be made. The Xilinx ISE 10.1 tool was used with no clock speed constraints provided. On-chip block memory was instantiated as appropriate.

Cadence EDI System suite v9.11 was used for placement and routing of both the standard-cell and sASIC flows. For standard-cell ASIC designs, an initial utilization of 0.75 was set for floorplanning all the benchmark designs, to ensure that all designs can finish without design rule violations. All metal layers were set to be usable in the ASIC flow. For the sASIC flow, instead of building one universal fabric for all designs, we built a customized fabric for each design. The layers of routing were also allowed to change with the design, depending on the design's routing congestion status. The worst case timing libraries were used during placement and routing. I/O pins were placed if possible entirely on the bottom, otherwise on both the top and the bottom of the design. This was done to mimic a real-world situation where the placement of the I/O pins of a block in a SoC is usually predefined and fixed, rather

TABLE I
AREA AND CLOCK PERIOD OF BENCHMARK CIRCUITS ON DIFFERENT PLATFORMS

	P&R area (mm^2)				Ratio			P&R clock period (ns)				Ratio		
Designs	STD	3-LUT	AOI22	FPGA	L/S	A/S	F/S	STD	3-LUT	AOI22	FPGA	L/S	A/S	F/S
s35932	0.152	0.844	0.277	4.705	5.54	1.82	30.87	0.810	2.420	1.140	4.151	2.99	1.41	5.12
s38417	0.155	0.743	0.280	4.436	4.78	1.80	28.58	1.460	4.480	2.200	9.503	3.07	1.51	6.51
s38584	0.140	0.627	0.267	4.178	4.48	1.91	29.86	1.030	3.060	1.400	6.472	2.97	1.36	6.28
b14_1	0.105	0.422	0.189	3.372	4.02	1.80	32.12	2.530	5.940	3.570	17.492	2.35	1.41	6.91
b15_1	0.100	0.448	0.193	3.876	4.48	1.93	38.77	2.140	5.030	3.150	11.398	2.35	1.47	5.33
des_area	0.049	0.414	0.102	1.651	8.47	2.10	33.79	1.270	3.420	1.810	6.240	2.69	1.43	4.91
des_perf	1.639	4.918	2.416	19.220	3.00	1.47	11.73	3.060	6.230	5.870	5.373	2.04	1.92	1.76
vga_lcd	0.222	1.047	0.336	2.207	4.72	1.51	9.95	1.670	4.930	1.790	6.400	2.95	1.07	3.83
ethernet	0.263	1.246	0.400	6.152	4.73	1.52	23.36	1.700	4.900	1.900	8.826	2.88	1.12	5.19
avg.					4.91	1.76	26.56					2.70	1.41	5.09

than being freely placed by the EDA tool. Static timing analysis was done with Synopsys PrimeTime C-2009.06-SP3-2.

Area measurements of standard-cell and sASICs only include the logic cells and hard-macros. The area of the core power rings, macro halos and filler cells were omitted. This facilitates comparisons with FPGAs, which are based solely on logic utilization. The comparison of area between standard-cell or sASICs against FPGAs is not straightforward, as little information regarding die area and the distribution of transistors for logic or routing is available. As the actual area of a CLB on the Virtex-II device is reported to be 10240 μm^2 in [30], we multiply this value by the number of CLBs used for individual designs, giving an estimate of the actual occupied die area on the FPGA. The area occupied by hard block memories instantiated in designs is not included in the estimation. It should be noted that in real designs a substantial number of "hard blocks" on FPGAs, such as digital signal processing block or memories, may be included, reducing the area overhead for FPGAs versus ASICs. Since in our benchmarks only two designs used hard blocks, our average area result for FPGAs could be pessimistic depending on the type of designs implemented. This does not affect the comparison with sASICs.

A. Area and Critical Path Delay Comparisons

Table I lists the area and clock period of designs implemented on standard-cell ASIC (STD), three-input LUT-based sASIC (3-LUT), AOI22-based sASIC (AOI22), and the Xilinx Virtex-II FPGA, respectively. Columns titled "L/S," "A/S," and "F/S" refer to results for 3-LUT/AOI22-based sASICs and FPGA implementations normalized against the corresponding standard-cell ASIC implementation.

From Table I, we can see that the area occupied by the two sASICs are between those of ASICs and FPGAs. The AOI22 sASIC design has improved area density over the 3-LUT design. The 3-LUT and AOI22 sASIC implementations have an average area overhead of 4.91 and 1.76 compared with ASICs. FPGAs, which use SRAM-based logic and routing configuration have a 26.6x area overhead over ASICs. This figure is consistent with that found in other works [8].

TABLE II
TOTAL WIRE LENGTH

	Wire length (μm)			WL ratio	
Designs	STD	LUT	AOI	L/S	A/S
s35932	0.401	1.201	0.783	3.00	1.95
s38417	0.268	0.713	0.693	2.66	2.59
s38584	0.370	0.523	0.589	1.42	1.59
b14_1	0.343	0.710	0.431	2.07	1.26
b15_1	0.264	0.623	0.430	2.36	1.63
des_area	0.150	0.575	0.310	3.83	2.06
des_perf	3.182	6.024	7.481	1.89	2.35
vga_lcd	0.367	1.283	0.611	3.49	1.66
ethernet	0.491	1.615	0.973	3.29	1.98
avg.				2.67	1.90

Both the 3-LUT-based and AOI22-based sASICs performed best in terms of area for "des_perf," which is the largest among the nine benchmark circuits. The overheads were 3x and 1.47x that of the ASIC, respectively.

On average, the 3-LUT-based sASIC has 2.7x the critical path delay of the ASIC. The AOI22-based sASIC had the best performance, with an average delay of 1.41x that of the ASIC. The FPGA was the slowest with a speed degradation of 5.09x.

The AOI22-based sASIC performed remarkably well in terms of speed, achieving 1.07x and 1.12x the delay of ASICs for the "vga_lcd" and "ethernet" designs. It performed worst for "des_perf," with a delay of 1.92x that of the ASIC. In contrast, both the 3-LUT-based sASIC and the FPGA performed best for "des_perf," at 2.04x and 1.76x delay. The excellent speed of the 3-LUT sASIC and FPGA for "des_perf" were because the circuit is efficiently constructed using complex LUTs. The AOI22-based sASIC, on the other hand, needs extra levels of logic gates to implement the same complex logic, impacting on its performance. Also, the FPGAs routing were fully buffered, and features direct chains between neighboring slices, whereas the highest buffer strength available on our sASICs were limited by the dedicated buffer cells selected.

Table II shows the total wire lengths for each design on the three platforms. The 3-LUT-based sASIC requires on average

TABLE III
AREA DELAY PRODUCT OF BENCHMARK CIRCUITS ON DIFFERENT PLATFORMS

	Area delay product				Ratio		
Designs	STD	LUT	AOI	FPGA	L/S	A/S	F/S
s35932	0.123	2.04	0.316	19.5	16.5	2.56	158.2
s38417	0.227	3.33	0.616	42.2	14.7	2.72	186.0
s38584	0.144	1.92	0.374	27.0	13.3	2.60	187.6
b14_1	0.266	2.51	0.675	59.0	9.4	2.54	222.1
b15_1	0.214	2.25	0.607	44.2	10.5	2.84	206.5
des_area	0.062	1.42	0.185	10.3	22.8	2.99	166.0
des_perf	5.014	30.64	14.181	103.3	6.1	2.83	20.6
vga_lcd	0.210	4.69	0.429	14.1	22.3	2.04	67.3
ethernet	0.376	5.90	0.679	54.3	15.7	1.81	144.4
avg.					14.6	2.55	151.0

2.67x that of an ASIC, whereas the AOI22-based sASIC is 1.9x.

We also calculated the area-delay product of each design for the three platforms. The results are shown in Table III. The AOI22-based sASIC achieved an average area-delay product of 2.55x that of the ASIC. The 3-LUT-based sASIC had an average of 14.6x, while the FPGA has the worse area-delay product average of 151.0x. These results showed that in terms of area-delay product, our AOI22-based sASIC is about 59.2x better than the FPGA, and a 5.73x improvement over the 3-LUT sASIC. The 3-LUT-based sASIC is still 10.3x better than the FPGA in terms of area-delay product.

Our AOI22-based sASIC achieved an average of 1.76x/1.41x area/delay ratio against ASICs. These figures are comparable with those obtained by Gopalani *et al.* [10], [27] who reported an average area/delay ratio of 1.12x/1.4x. It should be noted that Gopalani used small benchmark circuits, approximately 0.1 mm^2 in area, and if we remove the largest design, "des_perf," from our benchmarks, a delay ratio of 1.35x is achieved. Compared with their work, the main differences are: 1) our buffer sizing scheme that we believe is an improvement over connecting multiple NAND2 gates in parallel; 2) our approach of using standard D-flip-flops rather than constructing them from NAND2 gates reduces vulnerability to timing hazards and offers full scan capabilities; and 3) their design used metal-1 to metal-4 so only masks from substrate to poly can be shared, adding to the mask costs.

Compared with this paper in [25], our fabric achieves better area with similar speed, leading to an improved area-delay product.

B. Fabric Parameters

In the previous section, the results reported were for custom design-specific fabrics. To create these fabrics, we set an initial fabric size by dividing the cell area in a synthesized netlist by an utilization of 0.65. On the fabrics we do initial placement, after which we adjust the logic/flip-flop/buffer ratio and scale the size of the fabrics as necessary. We iterate through this process until the designs can be successfully fit in their respective fabrics.

TABLE IV
LOGIC/FLIP-FLOP/BUFFER RATIO

Designs	Asg.	Block counts			Ratio	
		AOI	FF	BUF	A/F	B/F
s35932	4/1/2	7389	1728	968	4.28	0.56
s38417	4/1/2	7857	1564	1083	5.02	0.69
s38584	4/1/2	8132	1160	1322	7.01	1.14
b14_1	15/1/3	7261	215	805	33.77	3.74
b15_1	15/1/3	7045	417	624	16.89	1.50
des_area	15/1/3	4598	128	1082	35.92	8.45
des_perf	9/1/3	78238	8808	9971	8.88	1.13
vga_lcd	4/1/2	5687	1681	1287	3.38	0.77
ethernet	4/1/2	8899	2343	1850	3.80	0.79

TABLE V
FABRIC TOPOLOGIES

	Topologies			Ratio	
Properties	CTM	OBS	FIXD	O/C	F/C
Cell area (mm^2)	0.102	0.126	0.145	1.23	1.42
Wire length (μm)	0.310	0.377	0.955	1.22	3.08
Clk period (ns)	1.79	1.94	1.98	1.08	1.11

Table IV shows the assigned logic/flip-flop/buffer ratio (Asg.) for the fabrics, and the actual counts and ratio of them after the backend flow. It can be seen that in designs "s38584," "b14_1," and "des_area," the assigned logic/flip-flop ratio is lower than the actual usage ratio, which implies that is could be implemented on a smaller fabric with less flip-flops. However, increasing the logic/flip-flop ratio also increases the distance between flip-flop columns, which could affect the timing performance. Hence, we believe that perhaps a number of different fabrics are necessary to achieve reasonably good performance over different logic/flip-flop ratios.

Of course, depending on the application, it may be preferable to use a single fabric for all designs. To examine the worst case effect of this approach, we implemented the smallest ("des_area") design, on the fabric for the largest design ("des_perf"), which measures 1762.8 μm by 1765.6 μm. The design occupies 5% of the available area.

Table V shows the comparison over different fabric topologies. These included: 1) a custom built fabric (CTM); 2) using obstructs (OBS) to block out extra area using the "des_perf" fabric; and 3) using the "des_perf" fabric (FIXD), for the "des_area" design. The second topology is a commonly used method in ASIC backend design to control detailed placement location, and is adapted here to reduce the effects of using a large fabric for a small design. Without using OBS, the design is 42% larger in area, and required over 3x the total wire length when implemented on the large fabric. By using OBS, the area increased by 23%, and the total wire length by only 22% compared with on custom fabric, being 13 and 60% reducted, respectively, compared with no OBS used.

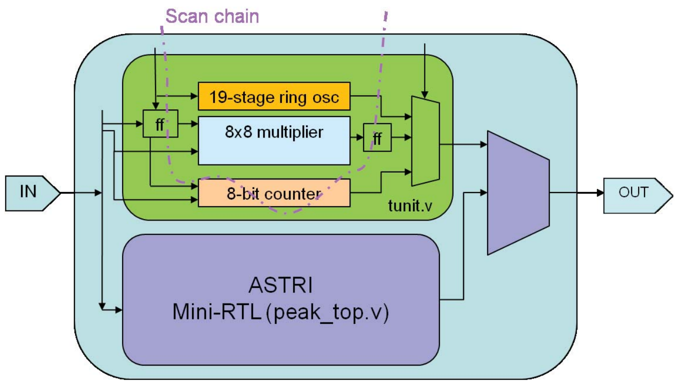

Fig. 10. Abstract for the tape-out design.

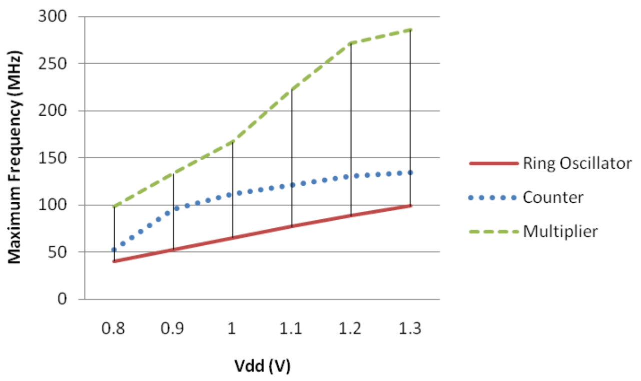

Fig. 11. Maximum frequency for the testing circuits under different voltages.

From our observation, the timing degradation involved is design-specific, and is in this case limited by I/O pin density. In the CTM case, pins can be spread in all four sides as wished, while in the OBS and FIXD cases pins were spread only on at most two sides to reduce the wire length from/to pins.

V. Prototype Chip

To verify our sASIC design methodology and implementation, we fabricated a sample design. The returned chips underwent scan and functional tests, and were used in a PC-based system.

The chip contains several test circuits along with a sample application, which is a subset of a LCD backlit controller design. The design was implemented with the 3-LUT-based sASIC described in Section II, on the UMC 0.13 μm 1P8M2T Mixed-Mode/RF process.

Fig. 10 shows the designs on the tape-out chip. The blocks "tunit" and "peak_top" were all implemented with our 3-LUT sASIC, while the top level MUXes were taken from the Faraday standard-cell library.

The test unit features a 19-stage ring oscillator, a eight-bit counter, and a 8×8 multiplier. The ring oscillator was created by cascading 19 NAND3 logic cells, each formed using a 3-LUT logic block.

Fig. 12. Picture used as sample input. Sample output is shown in the upper left corner.

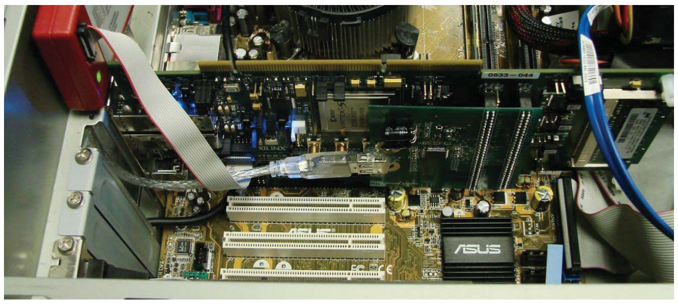

Fig. 13. Tape-out two chip on a custom PCB, plugged onto a Virtex-5 development board.

A. Prototype Chip Results

The fabricated chip is verified on a HILEVEL Griffin tester using both scan and functional tests. For the scan test, the generated scan pattern was shifted into the chip through the scan chain using the tester to verify its logical correctness against over 99% of stuck-at faults. All the test chips passed the scan tests.

For functional tests, we used a supply voltage over the range of 0.8–1.3 V, and recorded the maximum frequency for each test circuit. For the counter and the multiplier, the maximum frequency refers to the highest frequency for which correct outputs were generated. The 8×8 multiplier had its input shifted-in eight bits a time, and therefore requires two cycles to input data.

Fig. 11 shows the maximum frequency of test circuits against supply voltage. At the default voltage of 1.2 V for the typical case corner, the oscillation frequency of the 19-stage ring oscillator was 88.31 MHz, implying the delay of each 3-LUT-based NAND3 cell to be 0.30 ns. The maximum frequency for the counter recorded was 131 MHz, and that for the multiplier was 285.71 MHz.

B. Functional Test of LCD Backlight Controller

Fig. 12 shows a sample input for the system. The upper left corner shows the low resolution output of the LCD backlit panel. The sample pictures like this one were transformed into vectors and used as inputs for the circuit.

The outputs of circuit during functional test were verified against a software model as well as post-layout simulations.

After verification, the chip was inserted in a custom PCB, connected to an ML555 FPGA development board with a Xilinx Virtex-5 FPGA. The ML555 is interfaced to a Linux PC via PCI-E x8. The remaining parts of the application also reside on the FPGA. Jungo WinDriver [31] was used for the PCI-E software interface. This part replaces the LVDS interface of the original design, to act as input interface from PC memory/hard disk to FPGA. The output is sent back through the driver, and shown as a eight bit-depth frame on the screen.

Fig. 13 is a photograph showing the experimental setup. The USB cable is used solely for providing power to the custom PCB. This system verified correct system-level operation of the LCD backlight controller.

VI. Conclusion

In this paper, we presented 3-LUT and AOI22-based sASIC fabrics that can be customized using three masks. User-defined hard macros can be accommodated and full scan capability available. The associated design tools are based on a standard-cell compatible EDA design flow using industry-standard software. This approach allows the entire flow to be executed at the user's site, avoiding intellectual property disclosure issues, and alleviating the additional cost of purchasing new EDA tools.

Quantitative comparisons between the sASIC fabrics, FPGAs, and ASICs were made. The AOI22 cell was found to be superior to the 3-LUT in terms of area and speed, and both were significant improvements over FPGAs. A prototype chip was fabricated and used to verify the correctness of the fabric and design tools.

A. Future Work

There are several possibilities for improvement of the fabric reported in this paper. Currently, only one type of dedicated buffer is supported in the programmable fabric. The types and distribution of additional buffers could be investigated and may lead to improve the performance. More work could be done in optimizing the number of routing tracks occupied by logic blocks to improve the routability. Along the same lines, additional via-configurable routing structures could be prebuilt in the upper layers to relieve congestion when additional routing resources are required.

References

[1] *TSMC 65 nm Technology Overview (MPW)* [Online]. Available: http://www.europractice-ic.com/technologies_TSMC.php?tech_id=65nm

[2] M. LaPedus. (2011, Jan.). *450-mm Brings Confusion to Supply Chain* [Online]. Available: http://www.eetimes.com/electronics-news/4212786/450-mm-brings-confusion%-to-supply-chain?pageNumber=2

[3] P. Doe. *High Costs of Mask Sets and Design Force Industry Change* [Online]. Available: http://www.electroiq.com/index/display/semiconductors-article-display/2%05808/articles/wafernews/volume-11/issue-23/features/high-costs-of-mask-sets-a%nd-design-force-industry-change.html

[4] M. Horowitz, "Why design must change: Rethinking digital design," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarch.*, Dec. 2009, p. 267.

[5] L. Pileggi, H. Schmit, A. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Tong, "Exploring regular fabrics to optimize the performance-cost trade-off," in *Proc. Design Autom. Conf.*, Jun. 2003, pp. 782–787.

[6] K.-C. Wu and Y.-W. Tsai, "Structured ASIC, evolution or revolution?" in *Proc. Int. Symp. Phys. Design*, 2004, pp. 103–106.

[7] T. Okamoto, T. Kimoto, and N. Maeda, "Design methodology and tools for NEC electronics' structured ASIC ISSP," in *Proc. Int. Symp. Phys. Design*, 2004, pp. 90–96.

[8] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[9] R. Mehrabadi, S. Yuan, and R. Saleh, "Structured logic arrays for future CMOS technologies," in *Proc. Canad. Conf. Elect. Comput. Eng.*, Apr. 2007, pp. 235–238.

[10] S. Gopalani, R. Garg, S. P. Khatri, and M. Cheng, "A lithography-friendly structured ASIC design approach," in *Proc. 18th ACM Great Lakes Symp. VLSI*, 2008, pp. 315–320.

[11] U. Ahmed, G. Lemieux, and S. Wilton, "Performance and cost tradeoffs in metal-programmable structured ASICs (MPSAs)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 12, pp. 2195–2208, Dec. 2011.

[12] D. D. Sherlekar, "Design considerations for regular fabrics," in *Proc. Int. Symp. Phys. Design*, 2004, pp. 97–102.

[13] H. K. Phoon, M. Yap, and C. K. Chai, "A highly compatible architecture design for optimum FPGA to structured-ASIC migration," in *Proc. IEEE Int. Conf. Semicond. Electron.*, Dec. 2006, pp. 506–510.

[14] G. H. Oh, W. T. Lor, H. K. Phoon, and C. P. Lim, "High performance configurable distributed hybrid memory in structured ASIC," in *Proc. IEEE Int. Conf. Semicond. Electron.*, Nov. 2008, pp. 27–32.

[15] Y.-W. Tsai, K.-C. Wu, H.-H. Tung, and R.-B. Lin, "Using structured ASIC to improve design productivity," in *Proc. 12th Int. Symp. Integr. Circuits*, Dec. 2009, pp. 25–28.

[16] C. Patel, A. Cozzie, H. Schmit, and L. Pileggi, "An architectural exploration of via patterned gate arrays," in *Proc. Int. Symp. Phys. Design*, 2003, pp. 184–189.

[17] T. C. Chau, P. H. Leong, S. M. Ho, B. P. Chan, S. C. Yuen, K.-P. Pun, O. C. Choy, and X. Wang, "A comparison of via-programmable gate array logic cell circuits," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2009, pp. 53–62.

[18] Y.-C. Chen, H.-Y. Pang, K.-W. Lin, R.-B. Lin, H.-H. Tung, and S.-C. Su, "Via configurable three-input lookup-tables for structured ASICs," in *Proc. 20th Symp. Great Lakes VLSI*, 2010, pp. 49–54.

[19] K. Tong, V. Kheterpal, V. Rovner, L. Pileggi, and H. Schmit, "Regular logic fabrics for a via patterned gate array (VPGA)," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2003, pp. 53–56.

[20] A. Koorapaty, L. Pileggi, and H. Schmit, "Heterogeneous logic block architectures for via-patterned programmable fabrics," in *Field Programmable Logic and Application* (Lecture Notes in Computer Science), vol. 2778, P. Y. K. Cheung and G. Constantinides, Eds. Berlin, Germany: Springer-Verlag, 2003, pp. 426–436, 10.1007/978-3-540-45234-8_42.

[21] A. Koorapaty, V. Kheterpal, P. Gopalakrishnan, M. Fu, and L. Pileggi, "Exploring logic block granularity for regular fabrics," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, vol. 1. Feb. 2004, pp. 468–473.

[22] N. Jayakumar and S. Khatri, "A metal and via maskset programmable VLSI design methodology using PLAs," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, Nov. 2004, pp. 590–594.

[23] K. Gulati, N. Jayakumar, and S. Khatri, "A structured ASIC design approach using pass transistor logic," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1787–1790.

[24] N. Shenoy, J. Kawa, and R. Camposano, "Design automation for mask programmable fabrics," in *Proc. 41st Design Autom. Conf.*, Jul. 2004, pp. 192–197.

[25] Y. Ran and M. Marek-Sadowska, "Designing via-configurable logic blocks for regular fabric," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 1, pp. 1–14, Jan. 2006.

[26] M.-C. Li, H.-H. Tung, C.-C. Lai, and R.-B. Lin, "Standard cell like via-configurable logic block for structured ASICs," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Apr. 2008, pp. 381–386.

[27] S. Gopalani, R. Garg, S. Khatri, and M. Cheng, "DFM-aware structured ASIC design," in *Proc. 12th Int. Symp. Integr. Circuits*, Dec. 2009, pp. 29–32.

[28] *IWLS Benchmarks*. (2005) [Online]. Available: http://www.iwls.org/iwls2005/benchmarks.html

[29] Xilinx Inc. San Jose, CA (2007, Nov.). *Virtex-II Platform FPGAs: Complete Data Sheet*, [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds031.pdf

[30] H. Zarandi, S. Miremadi, C. Argyrides, and D. Pradhan, “Fast SEU detection and correction in LUT configuration bits of SRAM-based FPGAs,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, Mar. 2007, pp. 1–6.

[31] *Windriver PCI for Linux*. JUNGO Ltd., Netanya, Israel [Online]. Available: http://www.jungo.com/st/linux.html

Man-Ho Ho (M’11) received the B.Eng. degree in computer engineering and the M.Phil. degree in electronic engineering from the Chinese University of Hong Kong (CUHK), Hong Kong, in 2008 and 2011, respectively.

He was a Research Assistant with the Department of Electronic Engineering, CUHK, from 2008 to 2009. He is currently an Assistant ASIC Design Engineer with HiSilicon Technologies, Hong Kong. His current research interests include computer architecture and machine learning.

Yan-Qing Ai (S’12) received the B.Eng. degree in mechanical engineering from the Beijing Institute of Technology, Beijing, China, the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, in 2001 and 2004, respectively, and the M.Phil. degree in integrated circuit design from Zhejiang University, Hangzhou, China, in 2009. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, Chinese University of Hong Kong (CUHK), Hong Kong.

He was a Team Leader with National High Performance IC Design Center, Shanghai, China, a Project Leader with HiSilicon Technologies Company Ltd., Hong Kong, from 2004 to 2010, and a Research Assistant with the Application-Specific Integrated Circuit Laboratory, Department of Electronic Engineering, CUHK. His current research interests include sASIC design tools development, channel modeling, and protocols of body area network communication.

Thomas Chun-Pong Chau (S’09) received the B.Eng. and M.Phil. degrees in computer engineering from the Chinese University of Hong Kong, Hong Kong, in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree with the Department of Computing, Imperial College London, London, U.K.

His current research interests include reconfigurable computing, high performance computing, and real-time computing.

Mr. Chau was a recipient of the Croucher Foundation Scholarship.

Steve C. L. Yuen received the B.Eng. and M.Phil. degrees in computer engineering from the Chinese University of Hong Kong (CUHK), Hong Kong, in 2002 and 2004, respectively.

He was a Research Associate at CUHK from 2008 to 2010, where he led the Structured Application Specific Integrated Circuit Team. Currently, he is with the Transport Department of Hong Kong, engaged in the prequalification of traffic signal equipment. His current research interests include sensor and low-power electronics.

Chiu-Sing Choy (SM’96) received the B.Sc., M.Sc., and Ph.D. degrees from the University of Manchester, Manchester, U.K., in 1983, 1984, and 1987, respectively, all in electrical and electronic engineering.

He spent a year with Ferranti Microelectronics, Oldham, U.K., participating in application specific integrated circuit (ASIC) technology research from 1985. In 1986, he joined the Department of Electronic Engineering, Chinese University of Hong Kong, Hong Kong, where he is currently a Professor. His current research interests include network-on-chip, body network, structured ASIC, ultralow supply circuit techniques, multimedia, and baseband processing.

Prof. Choy is a fellow of Hong Kong Institution of Engineers (HKIE) and was a council member of the HKIE.

Philip H. W. Leong (SM’02) received the B.Sc., B.E., and Ph.D. degrees from the University of Sydney, Sydney, Australia.

He was a Consultant with ST Microelectronics, Milan, Italy, working on advanced flash-memory-based integrated circuit design in 1993. From 1997 to 2009, he was with the Chinese University of Hong Kong, Hong Kong. He is currently an Associate Professor with the School of Electrical and Information Engineering, University of Sydney. He is a Visiting Professor with Imperial College, London, U.K., and the Chief Technology Consultant with Cluster Technology. He is the author of more than 100 technical papers and holds four patents.

Dr. Leong was the Co-Founder and the Program Co-Chair of the International Conference on Field Programmable Technology (FPT), the Program Co-Chair of the International Conference on Field Programmable Logic and Applications, and is an Associate Editor for the *ACM Transactions on Reconfigurable Technology and Systems*. He was the recipient of the FPT Conference Best Paper Award in 2005 as well as the FPL Conference Stamatis Vassiliadis Outstanding Paper Awards in 2007 and 2008.

Kong-Pang Pun (S’97–M’01–SM’09) received the B.Eng. and M.Phil. degrees in electronic engineering from the Chinese University of Hong Kong (CUHK), Hong Kong, in 1995 and 1997, respectively, and the Ph.D. degree in electrical and computer engineering from the Institute Superior Técnico, Technical University of Lisbon, Lisbon, Portugal, in 2001.

He is currently an Associate Professor with the Department of Electronic Engineering, CUHK. He was a Visiting Scholar with the Department of Electrical Engineering, Columbia University, New York, NY, and the Institute of Microsystems Engineering, University of Freiburg, Breisgau, Germany, in 2004 and 2011, respectively. His current research interests include the circuits for complex signal processing, continuous-time filters, delta-sigma modulators, and ultralow-voltage circuits.

Prof. Pun was a General Co-Chair of the IEEE International Conference on Electron Devices and Solid-State Circuits in 2008 and a member of the International Technical Program Committee of the IEEE International Solid-State Circuits Conference from 2008 to 2011. He was a recipient of the Faculty Exemplary Teaching Award from the Faculty of Engineering, CUHK, in 2005 and 2010.