Standard Cell Like Via-Configurable Logic Block for Structured ASICs

Mei-Chen Li, Hui-Hsiang Tung, Chien-Chung Lai, Rung-Bin Lin*
Computer Science and Engineering, Yuan Ze University
135 Yuan-Tung Road, Chung-L, Taiwan
*csrlin@cs.yzu.edu.tw**

Abstract

A structured ASIC has some arrays of pre-fabricated yet configurable logic blocks (CLBs) with/without a regular routing fabric. In this paper, we propose a standard cell like via-configurable logic block (VCLB). We design a 0.18um standard cell library based on our VCLB and establish a design flow using as many commercial tools as possible. We also propose a method to evaluate the viability of a structured ASIC fabric. Our structured ASIC fabric with programmable metals for routing achieves a delay of 2.7 times, an area of 3 times, and a power of 1.5 times that attained by the designs using a commercial cell library.

1. Introduction

A structured ASIC has some pre-fabricated yet configurable logic block (CLB) arrays with/without a regular routing fabric [1-5] and perhaps some pre-diffused or customized IPs and programmable I/Os. A CLB can be via- or metal-configurable. A via-configurable CLB is less flexible but uses fewer customizable mask layers. A regular routing fabric has repetitive patterns pre-defined on the higher metal layers. It is normally via-configurable and thus has fewer customizable mask layers [6-9]. Some companies even advocate using only one via layer to customize both CLB and routing fabric [10]. As shown in Fig. 1, structured ASIC embraces a large mid-section of programmability spectrum. One end of the spectrum is cell-based ASIC with non-programmable cells but programmable routing metals. The other end is FPGA with SRAM-based programmable cells and routing fabrics. The gaps between these two ends are addressed in [2]. Structured ASIC can achieve a timing performance comparable to that of a cell-based ASIC while using much less power than that of an FPGA. It pays only the cost of customizing mask layers for cells and routing and an amortized mask cost for the remaining layers. It enables a fast manufacturing turn-around time. Its regular routing fabric is instrumental in improving manufacturability [11].

As an emerging design option, structured ASIC has many problems related to designing of CLBs and routing fabrics as well as tool development [12,13]. Especially, lack of tool supports for cell library development, placement and routing, logic synthesis, etc. has prevented structured ASICs from being widely adopted. Tool development is closely related to how CLBs are designed. Basically, a CLB should be designed to leverage the existing tools as much as possible and to reduce the numbers of customizable mask layers. Cell and routing fabric programmability has a great influence on tool development. Table 1 shows the programmability choices made by the major vendors [1,3,10,12,14,15] and research groups (the last four rows) [4,5,7]. This is by no means an exhaustive list. The last column indicates whether the same transistors in CLBs can be used to implement combinational logic and sequential elements. As one can see, most vendors choose metal programmability for cells and routing fabrics. Such a choice incurs a least amount of efforts for tool development. For example, a placer and a router for standard cell designs can be customized for this kind of structured ASICs. Conversely, a new router should be developed if a via-programmable routing fabric is used. Moreover, CLB layout should not complicate power/ground network distribution. As for cell granularity, a new logic synthesizer or a logic packer should be developed if a coarse grained cell (CLB) is adopted. Besides, designing of CLBs should also consider the following issues.

- Basic logic functions realized by a CLB.
- CLB composition capability.
- Transistor utilization of coarse grained CLBs.
- Library development efforts.

The above is by no means an exhaustive list but it presents the issues closely related to our work. These issues are co-related to each other. The basic logic functions realized by a CLB depend on CLB granularity. What we mean by a basic logic function is a logic function that has a template in a cell library employed by a logic synthesis tool. A coarse grained CLB, especially for a look-up table based CLB, can be configured to implement many basic logic functions. A fine grained CLB has a limited number of realizable functions, but its *composition capability* enables us to form more complex logic functions using more than one CLB. CLB composition only uses the programmable layers dedicated to CLBs for implementing a basic logic function. Metal programmable CLBs have the highest composition capability whereas via-programmable CLBs may have or not have composition capability. A coarse grained CLB may pay a large area penalty due to poor transistor utilization of CLBs. Moreover, coarse grained CLB complicates the library development task since library characterization should be done for a huge number of basic logic functions that can be realized by a CLB. Transistor utilization of coarse grained CLBs also depends on whether the transistors not used for combinational logic can be used to implement sequential elements in a CLB. Poor utilization could occur if each coarse grained CLB consists of a flip-flop and combinational logic blocks but the flip-flop is not used in a design. Conversely, fine grained CLB has better transistor utilization. However, it pays more area penalty for isolating CLBs from each other because more fine grained CLBs will be used to implement the same design. Library development effort is smaller for fine grained CLBs. Besides the above issues, designing of a programmable fabric using CLBs should also consider the relocating problem of an IP [14].

978-0-7695-3170-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ISVLSI.2008.50

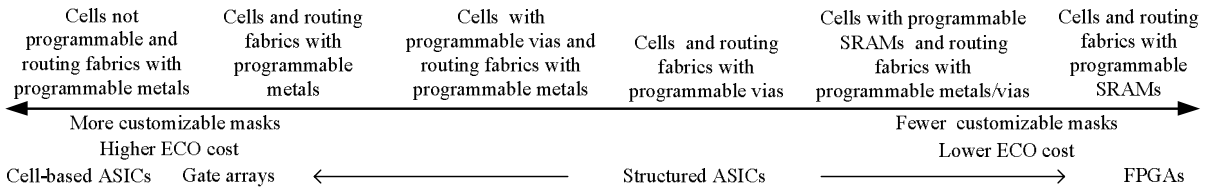

Figure 1. Programmability spectrum of cell and routing fabric.

Table 1. Programmability choices.

Vendors/ Researchers	Grain size	Programmability		Unified c&s?
		Cell	Routing fabric	
LSI	Fine	Metal	Metal	?
eASIC	Coarse	SRAM-based LUT	Via	No
Virage Logic	Fine	Metal	Metal	?
NEC	Coarse	Metal	Metal	No
AMI	?	Metal	Metal	?
ChipX	?	Metal	Metal	?
Fujitsu	?	Metal	Metal	?
Faraday	?	Metal	Metal	?
Altera	Fine	Metal	Metal	Yes
ViASIC	Coarse	Via	Via	No
Triad Semi.	?	Via	Via	?
Synopsys [5]	Coarse	Via	Via	No
Marek- Sadowska et al.	Coarse	Via	Via	Yes
Pilleggi et al.	Coarse	Via	Metal/Via	No
Ours	Medium	Via	Metal	Yes

In this paper, we propose a medium grained via-programmable CLB (VCLB). Our VCLBs along with either a metal or a via-programmable routing fabric can be used to construct a structured ASIC. Each VCLB has five pairs of P-N transistors. These transistors can be configured into a logic function with five or fewer inputs using M1-M2 vias. VCLBs can be abutted to form a more complex function using the jumpers at their both ends. The same VCLB can be used to implement either combinational or sequential elements. Power/ground (P/G) buses are placed on M2 to facilitate the deployment of M3 power straps. They are located at the top and bottom boundaries so that they can be made wider by cell row abutment. We establish a design flow primarily using commercial standard cell design tools. Our own tools include a logic packer and a placement legalizer. We create a 0.18um standard cell library based on our VCLB to test our design flow and to evaluate the viability of our VCLB. Experimental results show that our structured ASIC fabric with programmable metals for routing achieves a delay of 2.7 times, an area of 3 times, and a power of 1.5 times that attained by the designs using a commercial standard cell library. The method we propose here to evaluate the viability of a structured ASIC fabric is very important. It is based on the concept of pushing the delay envelope to see how small a delay can be achieved using a programmable fabric.

The rest of this paper is organized as follows. Section 2 discusses some VCLB design problems. Section 3 presents our programmable fabric. Section 4 describes our design methodology. Section 5 presents some experimental results. The last section draws conclusions and discusses future work.

2. Existing VCLBs

There are three types of VCLB. First, a VCLB can be an FPGA-like look-up table (LUT). LUT-based VCLBs proposed in [4] can be laid out in a way similar to standard cells. Standard cell design style enables simple P/G distribution. P/G buses between adjacent cells can be abutted and P/G straps using higher metal layers can be easily deployed to form a P/G network. The major problem of LUT-based VCLBs is that each LUT must be accompanied by a flip-flop. Since flip-flops are not used as frequently as LUTs, the utilization of transistors in VCLBs is low. The second type of VCLB is based on PLA structure [16]. PLA enables implementation of two-level logic functions, but it still needs flip-flops for implementing a sequential design. The third type of VCLBs [7] uses series-parallel layout structures to implement either combinational logic gates or flip-flops. Since our work is closely related to this type of VCLB, we will detail the work proposed in [7]. Two via-configurable base-cells are proposed in [7]. One is called *n*-ViaCC (or *n*-VCC for short) that consists of *n* pairs of P-N transistors. These transistors can be configured to implement *n* or fewer input functions. A 5-VCC in Fig. 2 is specially proposed. One can refer [7] to see which logic functions can be realized by a 5-VCC. A via in a 5-VCC can be installed at the place where an M1 line and an M2 line intersect. All the layers below M1 are pre-fabricated and M2 layer is with pre-defined mask patterns. Only the M1-M2 vias can be used to set the logic functions of a 5-VCC. The M2 P/G lines run horizontally in parallel with diffusions and M1 wires run vertically in orthogonal to M2 wires. The other base-cell is called *inverter array*(INVA for short) which can be used to implement inverters, XOR/XNORs, multiplexers, etc. An INVA has three diffusion strips for each type of transistors. A layout of INVA is shown in Fig. 2 where P/G lines are run horizontally in M1. A 5-VCC and an INVA can be combined to implement a flip-flop. An even larger programmable block called VCGA as shown in Fig. 3 is also presented. A VCGA has four basic logic elements (BLE). Each BLE has a 5-VCC and two INVAs. Note that an INVA takes a larger area than a 5-VCC. There are some problems with this VCGA.

- The short P/G lines in a VCGA are laid out on two different layers. Connecting them together, we must use a large number of vias and M1/M2 wires. This drastically increases the resistance of P/G network. Moreover, we need to add extra vias to connect the P/G networks among VCGAs, this further increases the P/G network resistance. Note that the P/G lines in a VCC and an INVA are not located at the boundaries so that row abutting to make wider P/G lines is not possible.
- We need to implement a complicate P/G network using higher metal layers (M3 or above) no matter whether the P/G lines in VCCs and INVAs are configured to be connected or not. Moreover, we can not freely drop a via from an M3 P/G strap to an M1 P/G line because a short to M2 signal wires in INVAs may occur.
- The number of functions that can be implemented by a VCGA is large and some of them can be very complex. Creating a library that consists of these functions for logic synthesis, physical design, and timing analysis tools is very difficult. No discussion of such efforts is made in [7].

In summary, the above drawbacks prevent VCGAs from being used as standard cells. Yet another problem with VCGA is that the fourth quadrant of a BLE does not have any transistors. Our standard cell like VCLB does not have the aforementioned problems.

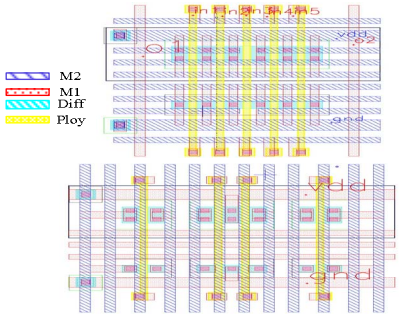

Figure 2. Layouts of 5-VCC (top) and INVA.

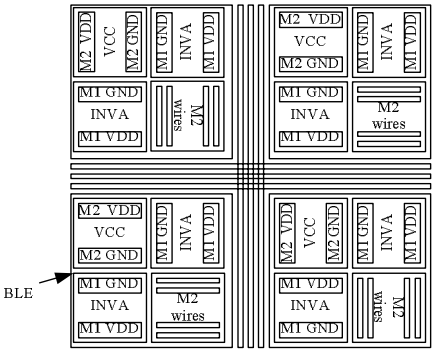

Figure 3. A VCGA proposed in [7].

3. Proposed programmable fabric

Our programmable fabric has a two dimensional array of VCLBs and a metal programmable routing fabric. We will first describe our VCLB and then demonstrate the configurability and composition capability of our VCLBs.

3.1. Layout of our VCLB

Fig. 4 shows the stick diagram and layout of our VCLB. Our VCLB has five transistor pairs. The three transistor pairs on the left are used to form CMOS logic and the two transistor pairs on the right are used to form pass transistor logic. CMOS logic is implemented using serial-parallel structures. Pass transistors can be better used to implement multiplexers and XOR/XNOR. Vias can be selectively inserted at the junctions of M1 and M2 wires to implement a logic function. Almost all the logic functions realized by a 5-VCC can be implemented by our VCLB.

Despite of a resemblance between our VCLB and the 5-VCC and INVA, our VCLB has the following features.

- Our VCLB is a monolithic block. It is asymmetric and can not be flipped w. r. t. Y-axis for structured ASIC designs.
- The preferable routing directions of M1 and M2 are vertical and horizontal, respectively.
- M2 P/G lines are located at the top and bottom boundaries. Our VCLBs can be abutted seamlessly to form a cell row. A cell row can be flipped w. r. t. X-axis so that it can abut the rows above and below it to form wider P/G lines.
- M3 (or high layer) P/G straps can be freely deployed above the core area because they can be connected to the M2 P/G lines in the cells without any restriction. Basically, what can be done for a traditional standard cell design can also be done for a design based on our VCLBs.
- The I/O pins are the short vertical M2 wire segments located at the central strap of the VCLB. They are on-grid and have enough space for via doubling. The unused inputs need not be tied to P/G because any floating input does not have a chance to create a short circuit from power to ground. However, a choice of connecting a pin to P/G can always be made.
- A VCLB has five long M2 wires for intra VCLB connections and three M1 jumpers at both ends for inter VCLB connections. The jumpers enable the composition of our VCLBs so that a more complex logic function can be implemented using several VCLBs.
- Creating a cell library using our VCLB is considerably easier than that using VCGA given in Fig. 3 because the number of logic functions that can be implemented by our VCLB is much smaller.

Our VCLB with respect to power planning is similar to that presented in [5]. However, there are two important differences. One is that each CLB in [5] must contain a fixed ratio of combinational logic elements to sequential elements. The other is that the CLB in [5] lacks composition capability.

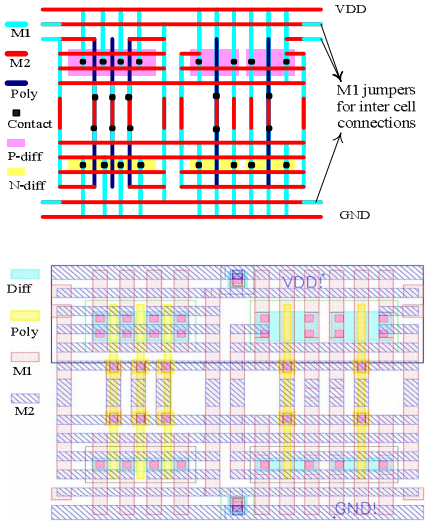

Figure 4. Proposed VCLB.

3.2. Configurability and composition capability

Here, we will show the configurability and composition capability of our VCLBs. The layout in Fig. 5 (left) is a 2-to-1 multiplexer. The third pair of P-N transistors is used to create the complement of *S0* and the two pairs of P-N transistors on the right are used to form two transmission gates. Two pairs of transistors are unused. Fig. 5 (center) shows a layout for AOI221X4. The layout consists of two VCLBs. A buffer of 4X driving capability is implemented in one VCLB. Fig. 5 (right) shows a layout of a positive edge triggered D flip-flop with active low reset. The layout consists of three VCLBs. From the above two layouts, one may find that the jumpers between two VCLBs have been employed to send signals from one VCLB to another. Note that more complex functions can be realized using a various number of VCLBs. Large drive inverters and buffers can be realized in a similar way. The VCLB can be modified to provide more jumpers and long wires for inter cell connections. Our library will have logic functions that uses two or more VCLBs. A multi-VCLB logic block can be freely placed at any legal position as long as it is confined within a cell row. That is to say, it is relocatable. Besides, more than one basic logic function can be packed into the same VCLB. Later, we will present a logic packer that carries out the packing task for a synthesized design.

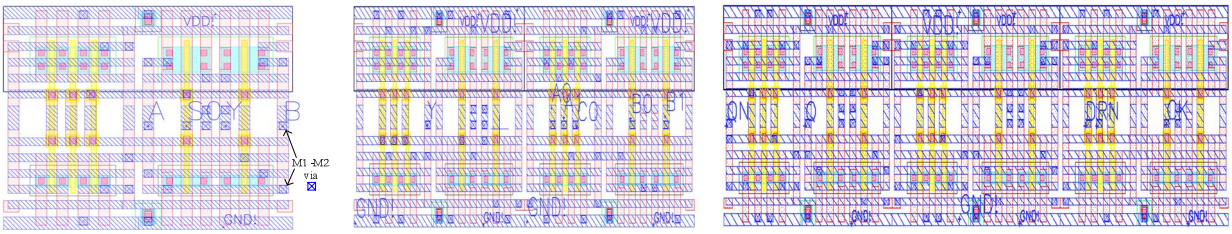

Figure 5. Layout of 2-to-1 multiplexer (left), layout of AOI221X4 (center), and layout of a D flip-flop.

4. Design methodology

Here, we present a design methodology for structured ASICs. Fig. 6 shows a simple flow of how to construct a standard cell library (the shaded part) and perform chip designs using a cell library based on our VCLB. To create a cell library, we first define a via map for each logic function. We install the so-defined vias on a VCLB to complete the layout of a logic function. We use Mentor's Calibre to extract post-layout parasitics and the tool in [17] to perform post layout timing characterization. Once this is done, we create a library called *DC cell library* for Synopsys Design Compiler. In DC cell library, since the transistors in the VCLBs for simple logic functions are not fully utilized, we hand-pack two logic functions into a single VCLB to form a multi-function packed block (MFPB). Determining which basic logic functions should be put into an MFPB is very complicate. We only combine the most frequently used logic functions into one MFPB. Accordingly, we have to create a layout for each MFPB. Once this is done, we create a library called *SOC-E cell library* for Cadence SOC Encounter. SOC-E cell library has packed versus non-packed logic blocks.

A design flow mostly using commercial tools such as logic synthesizer, placer, router, timing analyzer, etc. is also shown in Fig. 6. The tools developed by ourselves are a logic packer and a placement legalizer.

4.1. Logic packer

Our logic packer places two logic functions into a VCLB. It has the following steps.

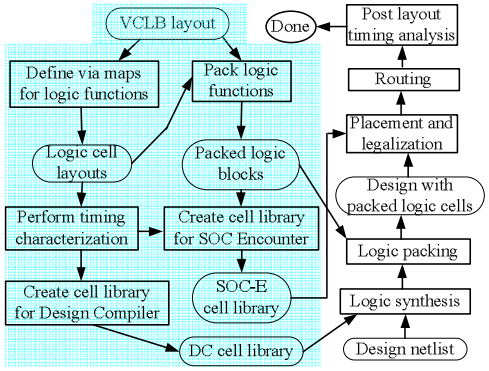

Figure 6. Library and chip design methodology.

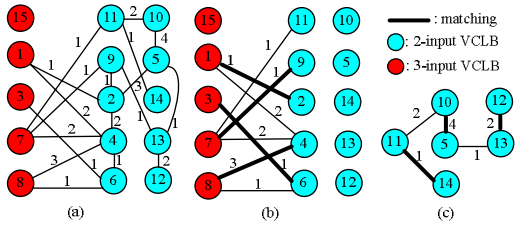

Figure 7. An example of logic packing.

Step 1: A graph is created. A non-fully used VCLB is modeled as a vertex. An edge between two vertices is created if the two corresponding VCLBs can be combined into one VCLB. Any two VCLBs can be combined into one VCLB if there is a corresponding MFPB in the SOC-E cell library. Each edge is assigned a weight using the approach proposed in [18]. The edge weight represents how strongly the two related VCLBs should be connected.

Step 2: We first perform a maximum weight bipartite matching on a subgraph formed by VCLBs, each of which implements either a 4-input or 1-input basic logic function. Once this is done, any two matched VCLBs are combined into one VCLB and the two vertices corresponding to the two original VCLBs are removed from the graph. Any 4-input VCLB that can not find a matching should be removed from the original graph. This step is repeated for a subgraph comprised of 3-input and 2-input VCLBs and a subgraph comprised of 3-input and 1-input VCLBs.

Step 3: We perform a general maximum weight matching on a subgraph comprised of only 2-input VCLBs, 2-input and 1-input VCLBs, and only 1-input VCLBs, respectively.

Fig. 7 shows a logic packing example. Fig. 7(a) is the original graph. Fig. 7(b) is the matchings for the subgraph comprised of 3-input VCLBs and 2-input VCLBs. The graph in Fig. 7(c) is obtained by deleting the vertices with matchings and the un-matched vertices that correspond to 3-input VCLBs. The matchings in Fig. 7(c) are for 2-input VCLBs only. We obtain an optimal solution for the example in terms of the number of matchings and total edge weights.

4.2. Placement legalizer

Our placement legalizer takes the result from a standard cell placer as input and determines a legal position for each cell such that the total displacement of all cells from their original positions is minimized. We assume that the order of the cells in a cell row is preserved. The legalizer is run for the cells in each row. A legal position for a cell in a row should be an integral multiple of VCLB width. This problem is formulated as a minimum weight bipartite matching problem. We create a bipartite graph as follows. A cell is modeled as a vertex in one group. The vertices in this group is denoted as C_i, i=0, n-1. Each of the legal positions on a row is modeled as a vertex in the other group. The vertices in this group is denoted as R_j, j=0, m-1. An edge between C_i and R_j is created if R_j is a permissible position for vertex C_i. A permissible position for C_i is a legal position so that there is at least a placement solution for all the cells in a row if C_i is placed at that position. Let the width of C_i be w_i, the total width of the cells to the left of C_i be ℓ_i, the total width of all the cells to the right of C_i be $\ell_{i'}$, and the length of a row be ℓ_r, all in terms of the VCLB width. Then, the permissible position for C_i starts from ℓ_i and terminates at $\ell_r - (\ell_{i'} + w_i)$. A weight is assigned to each edge between C_i and its permissible position R_j. The weight is set to the distance between the original position of C_i and R_j. Fig. 8(a) shows a cell row with

cells C_0 and C_1 whose positions are originally at 7.25 and 21. The width of a VCLB is 10. The permissible positions are 0, 10, 20 for C_0 and 10, 20, 30 for C_1, respectively. Fig. 8(c) shows a minimum weight bipartite matching which places C_0 at 10 and C_1 at 20, with a minimum weight of 3.75.

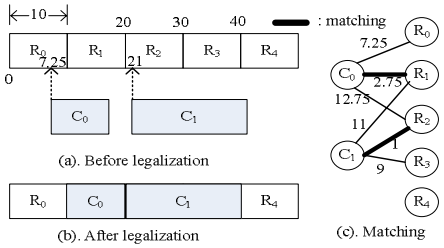

Figure 8. An example of legalizing cell positions.

5. Experimental results

We have implemented a cell library using our VCLB for evaluating the design methodology described in Fig. 6. When implemented using 0.18um process technology, our VCLB has an area of 78 um^2. The library consists of 105 unpacked cells and 73 packed cells. We selectively implement some functions that are found frequently used by a logic synthesis tool. Most of the logic functions except inverter, buffer, and flip-flop have driving capability of 1X, 2X, 4X. Buffer has driving capability from 1X up to 16X and inverter has driving capability from 1X up to 20X. D flip-flop with reset has driving capability of 1X and 2X. Power and timing data of each cell are characterized using HSPICE.

Two kinds of experiments are performed. One uses our library (YZUL for short) for doing chip design. The other uses a commercial standard cell library (STDL for short). Both libraries are based on the same 0.18um process technology. The same parameter settings such as power ring width, core utilization, etc. are used to design chips regardless of which library is used. The designs using YZUL are routed without using M1 and M2 layers whereas those using STDL are routed with M1 and M2 layers. The large benchmark circuits from ISCAS89 and ITC99 are used in our experiments. Our experiments are carried out in the following manner. First, we use Synopsys Design Compiler to synthesize each circuit without any timing requirements but aiming only to minimize the area. We then find the longest path delay for each circuit after the circuit is routed. We call such a delay *base_delay*. We then perform logic synthesis for each circuit with clock period specified as 0.9*base_delay, 0.8*base_delay, 0.65*base_delay, 0.5*base_delay, 0.35*base_delay, 0.2*based_delay, and 0.1*based_delay, respecyively. The reason for carrying out experiments in this way is to see how far we can push the delay envelope of the designs using each library.

Tables 2 and 3 show the longest path delays of the post-routed circuits respectively designed with both libraries. The numbers of cells and nets of the circuits synthesized without timing requirements using STDL are also given in Table 2. The columns denoted by "**1**" give the base_delays. The columns denoted by "0.9" give the longest path delay for the cases with clock period specified as 0.9*base_delay. Tables 4 and 5 show the cell areas of the designs. Tables 6 and 7 present the total power dissipation. Table 8 summarizes the delay, area, and power data. The column "Delay ratio" gives the ratio of the smallest longest path delay of a circuit using YZUL to that of the same circuit using STDL. For example, the small longest path delay of b14 using STDL is 4.9ns and it is 10.5ns for the case using YZUL. Thus, we have a ratio of 2.1. The area and power ratios are calculated accordingly. For example, the total area of b14 using STDL corresponding to the case with the smallest longest path delay is 283000 um^2 and it is 679000 um^2 for the case using YZUL. Thus, we have an area ratio of 2.4. Power ratio is similarly calculated. The power ratios of some circuits are smaller than 1. This does not mean that the circuits using YZUL consume less power because the circuits using YZUL have their delays several times more than that of the circuits using STDL. If we push the delay envelope, the delay of our structured ASIC designs is on average 2.7 times that of their standard cell counterparts at the expense of 3 times area and 1.5 times power. These data serve as achievable lower bounds for the case of using a via-programmable routing fabric.

The last two columns in Table 8 present some interesting data about achievable delay envelope. A value in the last column is the ratio of the largest longest path delay of a circuit using YZUL to the smallest longest path delay of the same circuit also using YZUL. A value in the column "STDL max/min delay ratio" is calculated similarly. We find that the circuits using STDL have much larger ratios. This implies that a standard cell library can enable more aggressive delay optimization than our YZUL. This is an important observation which gives a clear direction for the future research on CLBs. Part of the reason why YZUL achieves a smaller ratio is that each cell in YZUL has much more parasitic capacitance than that in STDL and thus has a larger cell delay. As a result, the total cell delay takes a dominant part of critical path delay of a circuit using YZUL so that its critical path can not be effectively optimized.

Transistor utilization of the VCLBs in the circuits using YZUL is on average 73%. This is smaller than that reported in [7]. Part of the reason for this is that we do not pack the un-related VCLBs.

Making a comparison of our results to the previous works [5,7] is difficult because they do not perform experiments like ours, i.e., not performing experiments to push delay envelope. The comparison depends on which point on the delay curve is used, whether layout parasitics are considered, what kinds of tools are used, etc. However, if a comparison must be made, our YZUL on average achieves a delay-area product of 8.0 and a delay-power product of 3.9 whereas the work in [5] from an EDA company on average achieves a delay-area product of 14.8 and a delay-power product of 5.4. Note that the work in [5] has a via-programmable routing fabric rather than a metal programmable routing fabric like ours. The work in [7], having a metal programmable routing fabric, achieves a delay-area product of 2.87 and a power-delay product of 1.56. However, these data are obtained without performing detailed routing.

Table 2. Longest path delay of designs using STDL (ns).

	Cells	Nets	1	0.9	0.8	0.65	0.5	0.35	0.2	0.1
s35932	3929	4553	11.2	12.3	13.0	11.4	12.4	9.4	13.2	2.0*
s38417	5033	5189	4.9	3.5	3.4	3.2	2.9	2.8	2.8	2.7
s38584	6067	6474	36.7	4.7	5.0	5.0	5.4	4.9	3.9	3.1
b14	2217	2666	19.7	14.3	13.9	11.4	6.8	6.5	4.9	5.6
b15	3127	3448	35.3	14.6	12.9	14.6	12.2	10.1	8.3	13.0
b17	10290	10509	28.7	12.8	12.7	13.4	11.6	16.0	19.1	14.3
b18	28002	30647	29.2	17.7	16.9	15.3	15.0	10.3	8.6	13.2
b19	58545	60412	36.6	20.0	18.1	16.5	16.0	13.6	17.2	13.6
b20	4674	5389	19.8	15.3	14.9	12.4	7.0	9.8	6.1	5.8
b21	4561	5263	20.0	15.0	15.7	12.1	7.6	7.0	5.4	5.5
b22	6894	7966	21.0	17.2	17.4	16.0	10.3	9.0	5.3	6.3
Ave			23.9	13.4	13.1	12.0	9.7	9.0	8.6	7.7

* This is not an error. This delay value is achieved due to having proper fanout optimization.

6. Conclusions and future work

In this paper, we have proposed a standard cell like VCLB such that existing tools for standard cell designs can be leveraged to design a structured ASIC based on our VCLB. Our approach achieves a delay of 2.7 times, an area of 3 times, and a power of 1.5 times that attained by the designs using a commercial standard cell library if we push the delay envelope of the designs. Currently, we are working on a via-programmable routing fabric and developing a router for it. A clock tree synthesizer and some timing and power optimization methods are yet to be developed.

Table 3. Longest path delay of designs using YZUL (ns).

	1	0.9	0.8	0.65	0.5	0.35	0.2	0.1
s35932	31.7	25.8	35.8	22.2	32.8	21.7	32.6	11.4
s38417	8.0	7.3	7.7	6.8	6.3	6.5	6.3	6.9
s38584	21.1	8.3	7.1	8.8	7.5	6.6	5.8	6.7
b14	21.6	16.0	13.6	13.4	10.5	11.1	11.6	12.4
b15	66.9	20.6	20.7	20.5	19.2	16.5	13.2	14.4
b17	44.7	19.6	21.0	18.7	17.7	15.5	17.0	16.2
b18	53.7	24.8	29.2	28.0	28.1	26.0	36.7	37.6
b19	55.4	27.1	31.4	30.9	28.2	20.9	22.1	29.4
b20	22.2	15.6	15.8	17.2	17.0	21.1	20.2	18.9
b21	28.3	23.4	15.3	15.4	16.0	16.4	19.1	17.5
b22	31.9	30.3	23.4	21.3	24.1	26.7	31.9	30.7
Ave	35.0	19.9	20.1	18.5	18.8	17.2	19.7	18.4

Table 4. Total cell area of designs using STDL (x1000um^2).

	1	0.9	0.8	0.65	0.5	0.35	0.2	0.1
s35932	180	182	182	182	182	182	191	268
s38417	180	196	205	232	273	293	309	308
s38584	173	169	169	169	169	169	170	183
b14	88	123	143	169	102	130	283	276
b15	116	121	122	121	121	128	166	206
b17	337	360	360	360	368	415	522	634
b18	953	996	1003	1007	1099	1079	1595	1694
b19	1880	1938	1944	1970	1996	2350	2640	3022
b20	170	244	291	353	214	331	535	560
b21	166	237	268	439	209	316	566	595
b22	248	343	363	473	322	438	810	803
Ave	408	446	459	498	459	530	708	777

Table 5. Total cell area of designs using YZUL (x1000um^2).

	1	0.9	0.8	0.65	0.5	0.35	0.2	0.10
s35932	783	787	787	787	787	787	788	820
s38417	715	780	809	915	1038	1094	1153	1115
s38584	706	699	698	698	701	713	812	915
b14	274	468	499	549	679	871	1044	951
b15	434	493	492	493	493	493	596	762
b17	1237	1466	1467	1474	1488	1712	2102	2189
b18	3236	4032	4077	4152	4341	4456	5801	6089
b19	6336	8024	8104	8251	8655	8659	11210	11905
b20	544	996	989	1291	1594	1927	2040	1989
b21	531	1413	929	993	1366	1911	2012	2049
b22	801	1746	1772	1502	1815	2488	2917	2935
Ave	1418	1900	1875	1919	2087	2283	2771	2884

Table 6. Power dissipation of the designs using STDL (mw, at 100MHZ, 20% toggling rate).

	1	0.9	0.8	0.65	0.5	0.35	0.2	0.1
s35932	3.6	3.3	3.4	3.4	3.3	3.4	3.4	9.4
s38417	5.6	6.0	6.5	7.6	9.2	9.9	10.8	10.8
s38584	5.1	5.5	5.5	5.5	5.5	5.4	5.6	6.2
b14	2.9	4.1	5.0	5.8	3.5	4.6	12.2	11.9
b15	3.7	3.8	3.8	3.8	3.8	4.0	5.3	7.6
b17	12.6	12.8	12.8	12.8	13.0	14.2	19.9	25.3
b18	32.6	36.9	36.9	36.9	39.7	38.8	61.5	66.3
b19	70.0	73.1	73.1	74.3	74.4	82.6	98.3	118.8
b20	5.9	8.7	10.8	12.5	7.6	12.8	22.9	24.8
b21	5.8	8.1	9.6	12.7	7.4	12.1	24.4	26.7
b22	8.7	11.8	13.3	17.9	12.1	16.9	33.9	36.0
Ave	14.2	15.8	16.4	17.6	16.3	18.6	27.1	31.3

Table 7. Power dissipation of the designs using YZUL (mw, at 100MHZ, 20% toggling rate).

	1.0	0.9	0.8	0.65	0.5	0.35	0.2	0.1
s35932	15.8	17.4	17.4	17.5	17.5	17.8	17.4	19.6
s38417	15.7	15.5	15.7	16.7	18.0	18.5	19.3	19.6
s38584	14.6	15.8	15.9	15.7	15.7	15.9	17.1	17.9
b14	4.8	6.7	7.3	8.4	10.0	12.7	15.2	13.8
b15	8.3	9.0	9.0	8.8	8.9	9.3	10.2	11.7
b17	25.1	28.4	28.2	29.1	28.5	31.8	35.7	35.1
b18	65.8	76.9	78.1	79.1	80.9	79.5	96.9	96.7
b19	128.1	154.3	157.6	158.6	161.5	156.8	188.5	193.0
b20	9.9	15.3	15.1	19.5	23.9	27.0	29.6	28.8
b21	9.5	21.3	14.2	15.0	20.8	27.8	29.0	29.4
b22	14.9	27.6	26.6	23.5	28.3	38.1	42.8	44.5
Ave	28.4	35.3	35.0	35.6	37.6	39.6	45.6	46.4

Table 8. Delay, area, and power ratio

	Delay ratio	Area ratio	Power ratio	Delay *power	Delay * area	STDL max/min delay ratio	YZUL max/min delay ratio
s35932	5.8	3.1	2.1	12.2	17.9	6.8	2.8
s38417	2.4	3.7	1.8	4.2	8.8	1.8	1.3
s38584	1.9	4.4	2.8	5.2	8.2	11.9	3.7
b14	2.1	2.4	0.8	1.8	5.2	4.0	2.0
b15	1.6	3.6	1.9	3.1	5.7	4.2	5.1
b17	1.3	4.6	2.4	3.3	6.2	2.5	2.9
b18	2.9	2.5	1.3	3.6	7.3	3.4	2.2
b19	1.5	3.7	1.9	2.9	5.7	2.7	2.6
b20	2.7	1.8	0.6	1.7	4.8	3.4	1.4
b21	2.8	1.6	0.6	1.7	4.7	3.7	1.8
b22	4.0	1.9	0.7	2.8	7.5	4.0	1.5
Ave	2.7	3.0	1.5	3.9	8.0	4.4	2.5

7. References

[1] HardCopy Structured ASICs: ASIC gain without the pain. http://www.altera. com/products/software/flows/asic/qts-structured_asic.html

[2] B. Zahiri, "Structured ASICs: opportunities and challenges," ICCD, 2003, pp. 404-409.

[3] K. C. Wu, Y. W. Tsai, "structured ASIC, evolution or revolution?" ISPD, pp.103-106, 2004.

[4] L. Pileggi et al., "Exploring regular fabrics to optimize the performance-cost trade-off," DAC, 2004, 782-787.

[5] N. V. Shenoy, J. Kawa, R. Camposano, "Design automation for mask programmable fabrics," DAC, 2004, pp. 192-197.

[6] C. Patel, A. Cozzie, H. Schmit, and L. Pileggi, "An architectural exploration of via patterned gate arrays," ISPD, 2003, pp. 184–189

[7] Y. Ran and M. Marek-Sadowska, "Designing via-configurable logic blocks for regular fabric", IEEE Trans. on VLSI Systems, Vol. 14, No. 1, Jan. 2006.

[8] A. Koorapaty, L. Pileggi, and H. Schmit, "Heterogeneous logic block architectures for via-patterned programmable fabrics," LNCS 2778, 2003, pp. 426–436.

[9] Y. Ran and M. Marek-Sadowska, "Via-configurable routing architectures and fast design mappability estimation for regular fabrics," IEEE Trans. on VLSI Systems, Vol. 14, Sept. 2006, pp. 998-1009.

[10] A. Levinthal and R. Herveille, "FlexASIC structured array: a solution to the DSM challenge," DesignCon 2005.

[11] V. Kheterpal, et al., "Design methodology for IC manufacturability based on regular logic-bricks," DAC, 2005, pp.192-197.

[12] D. D. Sherlekar, "Design considerations for regular fabrics," ISPD, April 18-21,2004, pp.97-102.

[13] A. Koorapaty, et al. "Exploring logic block granularity for regular fabrics," DATE, 2004, pp. 1468-473.

[14] White paper, "RapidChip technology: fast custom silicon through platform-based design," LSI Logic, 2004.

[15] T. Okamoto, T. Kimoto, N. Maeda, "Design methodology and tools for NEC electronics' structured ASIC ISSP," ISPD, 2004, pp.90-96.

[16] F. Mo, R. Brayton, "PLA-based regular structures and their synthesis," IEEE Trans. on TCAD, Vol. 22, No. 6, June 2003, pp.723 - 729.

[17] Shu-Ren Ker, "An Automatic Library Development System," Master Thesis, Yuan Ze University, Taiwan, June 1997.

[18] B. Hu and M. Marek-Sadowska, "Wire length prediction based clustering and its application in placement," DAC, 2003, pp. 800-805.