PAPER Special Section on Reconfigurable Systems

Region-Oriented Placement Algorithm for Coarse-Grained Power-Gating FPGA Architecture

Ce LI[†a)], ***Student Member***, **Yiping DONG**[†], ***and*** **Takahiro WATANABE**[†], ***Members***

SUMMARY An FPGA plays an essential role in industrial products due to its fast, stable and flexible features. But the power consumption of FPGAs used in portable devices is one of critical issues. Top-down hierarchical design method is commonly used in both ASIC and FPGA design. But, in the case where plural modules are integrated in an FPGA and some of them might be in sleep-mode, current FPGA architecture cannot be fully effective. In this paper, coarse-grained power gating FPGA architecture is proposed where a whole area of an FPGA is partitioned into several regions and power supply is controlled for each region, so that modules in sleep mode can be effectively power-off. We also propose a region oriented FPGA placement algorithm fitted to this user's hierarchical design based on VPR [1]. Simulation results show that this proposed method could reduce power consumption of FPGA by 38% on average by setting unused modules or regions in sleep mode.

key words: *FPGA, low power, region, hierarchical design, power consumption*

1. Introduction

Field-Programmable Gate Array (FPGA) has many advantages such as short development time and flexibility for commercial design. But the disadvantage is power consumption that limits its applications in mobile devices. Many commercial FPGA companies pay more attention to minimize process scaling to get a lower supply power [2], [3].

Also, many ASIC companies use FPGA to emulate their designs to check the design quality and debugs before tape-out. It can reduce the non-recurring engineering (NRE) [1]. Nevertheless, more and more chips in the portable devices or notebook computers, such as ARM, x86 CPU and CHIPSET are designed by using hieratical design method. Power consumption can be saved by dynamically power off some modules which are in sleep mode. In the ASIC floorplan, some Sleep Regions (SRs) are used for sleep modules. For FPGA, fine-grained power gating has been discussed [4]–[6]. It provides flexibility for power gating and is almost independent of placement. But, the drawback is area increasing, because each Clustered Logic Block (CLB) needs one related sleep transistor and related control logic in the FPGA chip. Therefore, most of current commercial FPGA chip design does not adopt power gating.

This paper presents a coarse-grained power gating FPGA architecture instead of traditional fine-grained, which supports hieratical design with sleep modules. It is used not only to fill the SR with CLBs that come from the same module but also to power off unused SR. In this architecture, CLB of the sleep modules cannot be placed with the one which is in always power-on module in the same region. Besides, the fewer the used SR is, the less power consumption is. So, placement algorithm plays an essential role on coarse-grained power gating FPGA architecture. From this consideration, we focus our effort on module placement algorithm.

The remainder of this paper is organized as follows. In the next section, related work is described. Section 3 introduces low power design background and SR based FPGA architecture. Placement algorithm based on SR is discussed in Sect. 4. The CAD framework which supports this new FPGA architecture and region oriented placement algorithm is shown in Sect. 5. Finally, experimental results and the conclusion are presented in Sects. 6 and 7,respectively.

2. Related Work

Former researches [4], [7], [8] focused on power reduction methods such as power-gating, clock gating, dual-VTH/VDD, micro-VDD-hopping and so on. Paper [5] introduced a field programmability of dual supply voltages for FPGA power reduction. High VDD was applied for critical path logic, and low VDD for non-critical path to save power consumption. This is a good way for low power consumption, but it does not take user's top-down design method into consideration. Usually, logic in the same modules has the same power voltage and state in ASIC design. So, our paper pays more attention on region placement of module level by using single supply voltage. What's more, dual-VDD can be added for different regions based on our FPGA architecture if needed.

An asynchronous FPGA architecture based on autonomous fine grain power gating was proposed in [6]. It is more efficient in power than synchronous FPGA at less than 30% utilization. However, most current FPGA architectures are synchronous, so, an asynchronous approach cannot be adopted.

In ref. [9], power gating of logic fabrics was investigated and region-constrained placement was applied to reduce leakage power of unused logic blocks on Xilinx FPGA. Their placement algorithm placed a designed circuit into contiguous regions by utilizing two different styles: hori-

Manuscript received April 22, 2011.
Manuscript revised July 29, 2011.

[†]The authors are with the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu-shi, 808–0135 Japan.
a) E-mail: rice@fuji.waseda.jp

DOI: 10.1587/transinf.E95.D.314

zontal and vertical placement. One limitation of this idea is that the parietal row can be used only when the lower rows are fully filled in horizontal placement. So, circuit for Input Output (IO) PAD on FPGA top edge may be placed in bottom row when FPGA size is large. It may increase the wire length and decrease FPGA performance.

In [10], we proposed a circuit named a 'power control hard macro' (PCHM) at the cost of increasing area, by which synchronous FPGA logic blocks are autonomously powered-off by the IO PAD or internal logic signal. We did fine-grained power gating by using a sleep enable (SLPEN) signal, but power efficiency is decreased for small CLB.

This paper proposes coarse-grained power gating instead of fine-grained in [10]. Only one PCHM is used for each SR which is composed of several CLBs. PCHM controls the power of the related SR. We could get both high efficiency of power and area by reducing PCHM count. Former study [1] paid more attention on the critical path affected by the placement. The placement will affect routing resource and power consumption. But this kind of placement could not support the low power FPGA design with sleep module. Based on the FPGA architecture with SR, placement algorithm for user design with sleep module is also explored.

3. Background

3.1 Sleep Module

ASIC chips are composed of several modules, each of which uses its own supply, because different modules have different performance objective and constraints. An example of mobile SoC (System on Chip) is shown in Fig. 1. Processor module and memory module use un-gated power. USB and WIFI module use gated power. A Power Management Unit (PMU) is used to control power supplies for USB and WIFI modules by the command of the processor. The power can be saved by shutting down the modules in sleep or idle state.

Current commercial FPGA architecture has the limitation in supporting above design with sleep modules. Former researches also pay less attention on design hierarchy. The basic units in a conventional FPGA [2] are CLB, IO PAD, Connection Box (CB), Wire Channel (WC) and SB. FPGA implements user design by dividing it into small pieces which can be achieved by CLBs. CB, WC and SB are used as wires to connect the input and output among the CLBs. Many multiplexers (MUX), buffers and transmission gates are used in FPGA basic unit to achieve flexibility and reprogram ability of FPGA. FPGA users generally pay more attention on optimizing their circuits, but can do little improvement based on the fixed FPGA architecture to reduce power. So, power gating methods of sleep module based on their circuit could not be implemented on FPGA chip.

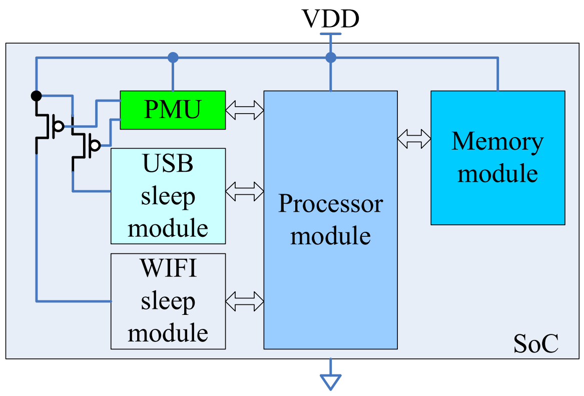


Fig. 1 Example of SoC power gating.

3.2 Proposed FPGA Architecture and Sleep Region

We focus on island-style FPGA architectures in this paper, shown in Fig. 2. In our proposed architecture, a power control hard macro (PCHM) is used as a low power controller as the PMU in Fig. 1. The PCHM could power off the 2 (column)*2 (row) CLBs, totally called an SR. The power of each SR can be gated by a sleep transistor which is controlled via the corresponding PCHM. A MUX is used for PCHM to select the SLPEN signal from internal connection. A MUX is used for PCHM to select the SLPEN signal from internal connection. In another word, we could assume that PCHM gets the SLPEN signal from one edge of the SB which is on the top-left of the SR. The MUX count equals the SR count on FPGA chip. To avoid leakage, isolate cells (ISOC) are added to CLB outputs which are not shown in this figure. Because when the circuit grows even larger, the size of SB, CB and WC are bigger, adding ISOC at the output of CLB need less area than at the CB or SB output. In Fig. 2, when CLBs in SR (1,2) and SR (2,2) are used for same module or have same power behavior, these two PCHMs have the same power state. SR (1,1) can be

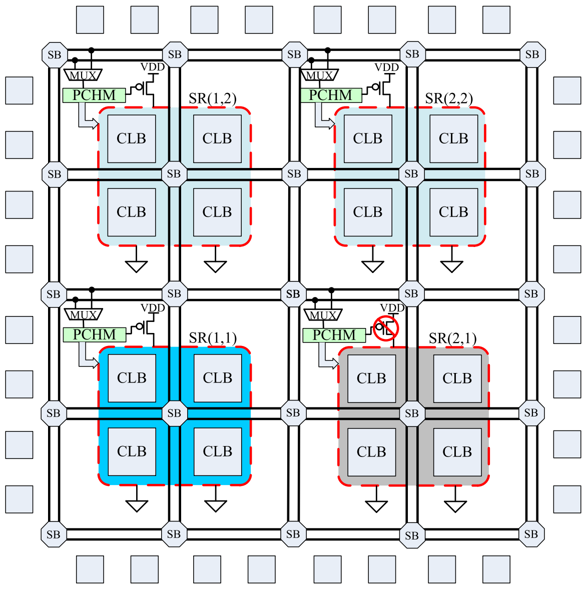


Fig. 2 A new FPGA architecture based on SR.

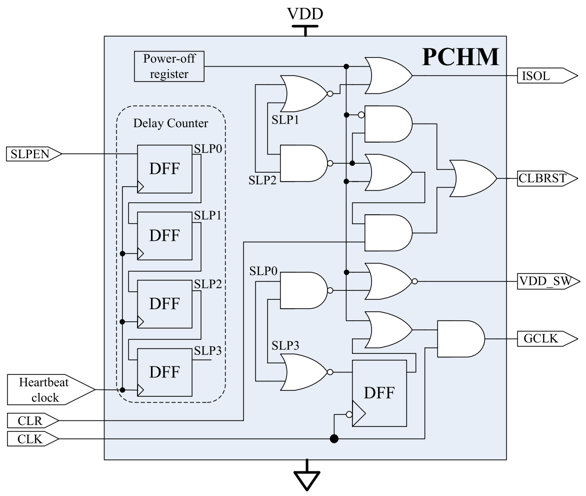


Fig. 3 PCHM block diagram.

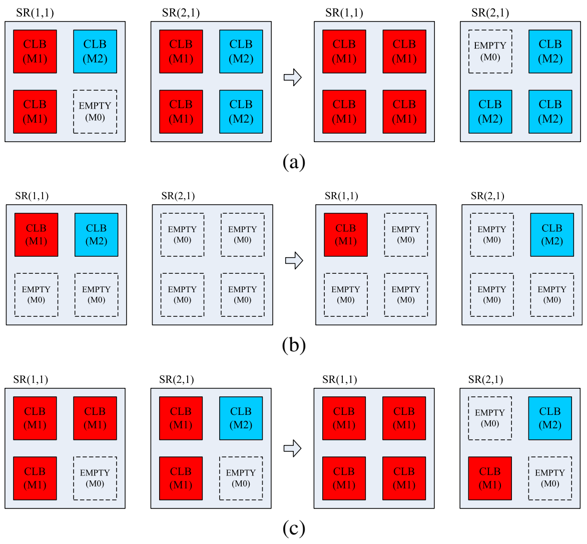


Fig. 4 CLB swapping results.

used for logic in another module. If no logic is mapped into SR (2,1), it could be powered off.

3.3 Power Control Hard Macro

It controls the power down and power up sequencing [10]. It also gates the clock of SR. Figure 3 shows the PCHM block diagram. A power-off register in top-left of this figure is a key register to gate power of connected SR in the highest priority. SLPEN comes from FPGA IO PAD or another CLB output after routing. It controls the power state of SR dynamically when the value of power-off register is “0”. Delay counter is used to meet signal phase delay by a group of D flip-flops (DFF). The count of DFF can be changed basing on power sequence. To reduce power consumption, we use a heartbeat clock (32 KHz) for the delay counter. SLP0, SLP1, SLP2 and SLP3 are outputs of the delay counter. They are used by combinational logic to generate the SR control signals such as ISOL, CLBRST, VDD_SW, and GCLK. ISOL is used by ISOC to isolate the power off domains and power on domain. The VDD_SW and GCLK are power gating control signal and gated clock sent to the SR, respectively. CLBRST is CLB reset signal based on the global clear signal (CLR) and power on/off sequence.

4. Placement Algorithm

FPGA placement algorithm determines CLB location of user circuit on chip. The goals are to minimize total area, wire connections, delay of critical path and so on. We focus our effort on finding a fast method which can separate the CLB of different modules into different SRs.

4.1 Linger Congestion Cost and Timing Driven Cost

Based on the adaptive annealing schedule, an FPGA CAD tool called Versatile Place and Route (VPR [1]) could get a better placement result within a short time by using Simulated Annealing (SA) method. By using the linear congestion (LC) cost function below, VPR can model the difficulty of routing connections in areas with different channel widths.

$$Cost_{LC} = \sum_{i=1}^{N_{net}} q(i)\left(\frac{bb_x(i)}{C_{av,x}(i)^{\beta}} + \frac{bb_y(i)}{C_{av,y}(i)^{\beta}}\right) \quad (1)$$

The summation [1] is over the N_{net} in the circuit. For each net i, $bb_x(i)$ and $bb_y(i)$ denote x (horizontal) and y (vertical) spans of its bounding box respectively. The factor $q(i)$ compensates for the fact that the bounding box wire length model underestimates wiring necessary to connect net with more than three terminals [11]. Its value depends on the number of terminals of net i. $C_{av,x}(i)$ and $C_{av,y}(i)$ are the average channel capacities (in tracks) in x and y directions respectively, over the bounding box of net i [1].The exponent, β, allows relative cost of using narrow and wide channels to be adjusted.

To get higher FPGA performance, path timing driven cost, $Cost_T$, is used in [1]. With $Cost_T$, CLBs connected by the critical path can be placed closely to reduce path delay of whole FPGA chip. So, critical path delay is smaller.

4.2 Sleep Region Cost

None of the above cost function can support the SR placement. A new cost, $Cost_{SR}$, is introduced to indicate the sleep region cost. To make this idea clearly, CLB placement cases which could occur after initial placement or during CLBs swapping are shown in Fig. 4 [12]. Each SR shown on the left side has CLBs in two modules (M1 and M2) colored by red and blue. M0 means the current CLB slot is EMPTY. Two SRs are enough for the two modules placement in Fig. 4 (a) and (b). The desired placement is shown on the right side. CLBs in different modules are separated

Table 1 SR parameter value.

		Before Swap		After Swap	
		SR (1,1)	SR (2,1)	SR (1,1)	SR (2,1)
Case (a)	$N_{CLB}(m,n,0)$	1	0	0	1
	$N_{CLB}(m,n,1)$	2	2	4	0
	$N_{CLB}(m,n,2)$	1	2	0	3
	$N_{mc}(m,n)$	2	2	1	1
Case (b)	$N_{CLB}(m,n,0)$	2	4	3	3
	$N_{CLB}(m,n,1)$	1	0	1	0
	$N_{CLB}(m,n,2)$	1	0	0	1
	$N_{mc}(m,n)$	2	0	1	1
Case (c)	$N_{CLB}(m,n,0)$	1	1	0	2
	$N_{CLB}(m,n,1)$	3	2	4	1
	$N_{CLB}(m,n,2)$	0	1	0	1
	$N_{mc}(m,n)$	1	2	1	2

and placed in two SRs.

Two parameters are used for module information for each SR. One is module count in each SR, called $N_{mc}(m,n)$; the other is CLB count, $N_{CLB}(m,n,p)$, in each module (from 0 to p) in SR. m and n are SR coordinates on FPGA chip. p is the index in total module numbers (N_{mc}) of FPGA. No logic is assigned to CLB slot when p is zero. The values of N_{mc}(m, n) and N_{CLB}(m, n, p) are shown in Table 1. The placement goal for the SR architecture FPGA is to make sure each N_{mc}(m, n) is less than 2, and make N_{CLB}(m, n, p) as big as possible for each SR.

A cost function, $Cost_{SR}$(m, n, p), is introduced to indicate a module cost of SR located in physical coordinate (m, n) of FPGA as shown in Eq. (2). It is award swapping for SR density increasing. $N_{SR-size}$ means the maxim CLB count in one SR. If SR size is 2*2, $N_{SR-size}$ is 4. $Cost_{SR}$(m, n, p) turns to zero when SR is fully filled with CLBs in the same module or without any CLBs. The cost function is shown as follow,

$$Cost_{SR}(m,n,p) = \begin{cases} 1 - (\frac{N_{CLB}(m,n,p)}{N_{SR-size}})^2 & N_{CLB}(m,n,p) \neq 0 \\ 0 & N_{CLB}(m,n,p) = 0 \end{cases} \tag{2}$$

The $Cost_{SR}(m,n)$ of one SR is cost summation of each power domain shown in Eq. (3). Module index 0 is used for "EMPTY" slots. Their cost does not need to be added into $Cost_{SR}$. But if we only do the addition of $Cost_{SR}(m,n,p)$ in each power domain, we cannot handle the case shown in Fig. 4 (b). Total cost of these two SRs will not be changed after swapping. So, we use $N_{mc}(m,n)$ to penalize the case that one SR has CLBs that come from different modules. Total SR cost is shown in Eq. (3). After swapping, $N_{mc}(1,1)$ and $N_{mc}(2,1)$ is 1 in Fig. 4 (b). $Cost_{SR}$ for the two SRs is reduced. Total SR cost is the summation of all $Cost_{SR}$ on the FPGA shown in Eq. (4). So in Fig. 4 (a), $Cost_{SR}$ before swapping is 7.375, and it is 1.4375 after swapping.

$$Cost_{SR}(m,n) = N_{mc}(m,n) * \sum_{p=1}^{N_{mc}} Cost_{SR}(m,n,p) \tag{3}$$

$$Cost_{SR} = 1 + \sum_{m=1}^{Width} \sum_{n=1}^{Length} Cost_{SR}(m,n) \tag{4}$$

Algorithm 1 Pseudo code of trying swap.

```
$grid\_from$ = start coordinate in CLB grid
$grid\_to$ = destination coordinate in CLB grid
$grid\_form.p$ = current module index of start position
$grid\_to.p$ = current module index of destination
$sr\_from$ = start coordinate in SR grid
$sr\_to$ = destination coordinate in SR grid

swap2clb{
random_pick_one_CLB(grid_from);
random_pick_destination(grid_to, $R_{limit}$);
if $grid\_form.p \neq grid\_to.p$ and $sr\_from \neq sr\_to$ then
    $\Delta Cost_{SR} \leftarrow calculate\_SR\_cost\_change(grid\_from, grid\_to)$;
else
    $\Delta Cost_{SR} \leftarrow 0$;
end if
$\Delta Cost_{LCTSR} \leftarrow calculate\_cost\_change(\Delta Cost_{LC}, \Delta Cost_T, \Delta Cost_{SR})$;
if $swap\_accept(\Delta Cost_{LCTSR}, T)$ then
    $Cost_{LCTSR\_newT} \leftarrow Cost_{LCTSR\_oldT} + \Delta\ Cost_{LCTSR}$;
    if $grid\_form.p \neq grid\_to.p$ and $sr\_from \neq sr\_to$ then
        $N_{CLB}(grid\_from, grid\_from.p) - 1$;
        $N_{CLB}(grid\_to, grid\_from.p) + 1$;
        $N_{CLB}(grid\_to, grid\_to.p) - 1$;
        $N_{CLB}(grid\_from, grid\_to.p) + 1$;
        swap(grid_from, grid_to);
    end if
end if
}
```

Figure 4 (c) shows an unwilling case. In this case, after swapping, SR (1,1) is fully filled with CLBs in M1. $N_{mc}(2,1)$ is 2 after swapping. But we could not allow SR (2,1) has the CLBs which come from different modules when these two modules do not have same power state. If there are more SRs in FPGA chip, swapping will be continued. So, we must calculated the minimal SR count for all modules, and then, use an FPGA with enough space to do placement. More conditions are described in Sect. 5.

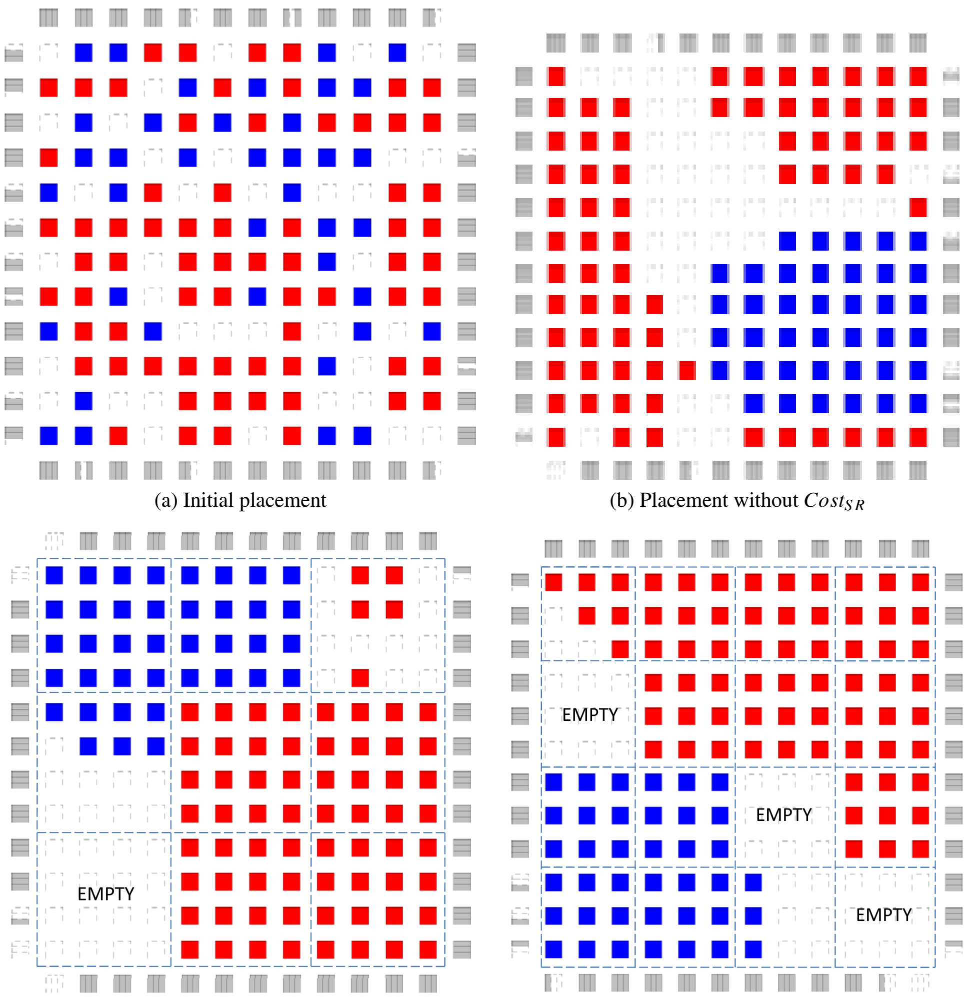
4.3 FPGA Total Cost

We update FPGA full chip cost function, $Cost_{LCTSR}$, based on the cost of $Cost_{LC}$, $Cost_T$ and $Cost_{SR}$. To do the tradeoff between each cost, factors γ and τ are introduced shown in Eq. (5). Each old temperature ($oldT$) cost is added to balance different cost. It makes the tradeoff factor more accurate.

$$Cost_{LCTSR} = (1-\gamma)((1-\tau)\frac{Cost_{LC}}{Cost_{LC\_oldT}} + \tau\frac{Cost_T}{Cost_{T\_oldT}}) + \gamma\frac{Cost_{SR}}{Cost_{SR\_oldT}} \tag{5}$$

In original VPR environment, τ is set to 0.5 to balance $Cost_{LC}$ and $Cost_T$ [13]. After we enhanced VPR, the minimal γ could be auto determined by user's design. It can make sure the circuit has a better performance based on $Cost_T$ and $Cost_{LC}$ when supporting SR placement. Of course, it supports fixed γ provided by user. More information is given in Sect. 5 about CAD flow.

Millions of potential block swaps will be evaluated in a typical placement even with a good annealing schedule.



(a) Initial placement (b) Placement without $Cost_{SR}$

(c) Placement with $Cost_{SR}$ ($N_{SR-size}$ = 4*4) (d) Placement with $Cost_{SR}$ ($N_{SR-size}$ = 3*3)

Fig. 5 Placement result by different cost.

Making computation as fast as possible is crucial. To reduce CPU time during placement, we do not re-compute $Cost_{LCTSR}$ during each swap. The swapped CLBs come from either the same or different modules, or even swap to a empty slot. We just calculate cost change based on affected nets and SRs.

The change of $Cost_{LC}$ and $Cost_T$ due to affected nets by two swapped CLBs (or by moving a CLB to an empty slot) is calculated. $Cost_{SR}$ change is affected by SRs whose $N_{CLB}(m, n, p)$ are changed. Based on Eq. (2) and Eq. (3), only swapping two CLBs in different modules between different SRs can affect the cost. No consideration should be taken when the swapping happens inside the same SR. Algorithm 1 shows the pseudo code when swapping two CLBs.

During each swap, we pick one block of user's design randomly and get its CLB information, such as coordinate in grid scale and module index. Then, we choose destination within the R_{limit} (the swapping range of two CLBs). The $Cost_{SR}$ is calculated when the start point and the destination point are not in the same module and SR.

If the swap is accepted after assessing based on $\Delta Cost_{LCTSR}$ (total cost change) and new temperature ($newT$), module information of the affected SRs and grids are preprocessed. The cost in $newT$ is changed by adding cost change as follow,

$$Cost_{LCTSR\_newT} = Cost_{LCTSR\_oldT} + \Delta Cost_{LCTSR} \qquad (6)$$

Similar to the former work in [1], conditions to exit SA and to decrease T are not changed. When swapping range is small in low temperature, $Cost_{SR}$ is hardly changed. Only $Cost_{LC}$ and $Cost_T$ optimize local placement to reduce wire cost.

4.4 Cost Function Comparison

Different cost functions are compared based on the same FPGA architecture in Fig. 5. It shows placement results by VPR using different cost function. Since there is no benchmark to evaluate sleep module feature on FPGA, we use two MCNC [14] benchmarks instead and assume they are in different power states. The difference between two benchmarks and user's designt which has sleep module in it is that the interconnection between different modules. But it is impartial for different placers based on two benchmarks.

Figure 5 (a) shows initial placement after VPR is enhanced to get two circuits. CLBs in different color come from different modules. After initial placement, CLBs are placed randomly in FPGA. Placement result with SR cost is shown in Fig. 5 (b). Blank grids located between two modules are disordered. Although blank space is big enough for a 4*4 sleep region, but it is impossible to use power gating for unfix position. More efforts should be taken if we want to use some low power design methods. But, in Fig. 5 (c), based on SR cost, CLBs are placed into the minimum count of 4*4 size SRs. One region should be powered off in this case. If we change $N_{SR-size}$ to 3*3, we could power off three SRs shown in Fig. 5 (d).

5. CAD Framework

There are lots of commercial FPGA design tools. But, most of them could only support the FPGAs by the same provider, especially for the placement and routing tools which could not support other architecture. So, a CAD framework should be explored to support and evaluate our proposed architecture, that is, special placement for SR based architecture.

We use two benchmarks as two modules in user's hierarchical design. Fig. 6 illustrates CAD software flow that supports two modules. This flow composes a CAD framework that can be used for packing, placement, routing, and power simulation for non-commercial FPGA.

The Berkeley Logic Interchange Format (.blif) [15] files are used to describe MCNC benchmark circuits. We could get the .blif file from verilog RTL file by ODIN_II [16] and ABC [17]. The T-VPACK [13] program packs LUTs and flip-flops into CLB which contains one or more Logic Elements (LEs) in .net format for each module.

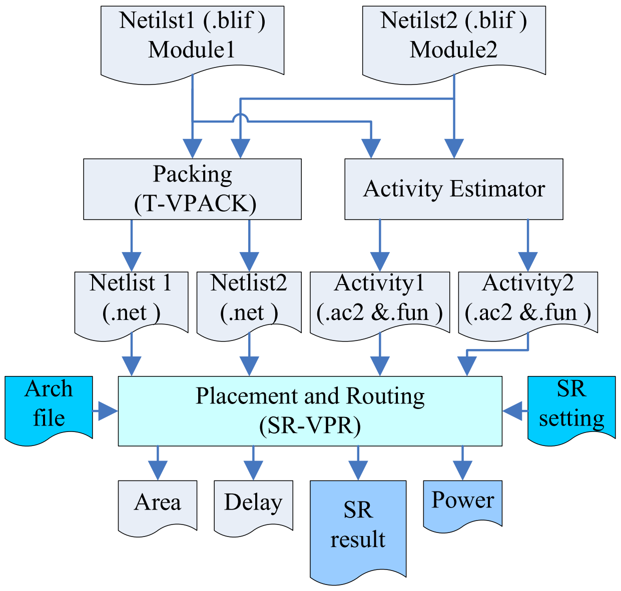


Fig. 6 Software flow for the SR based FPGA.

VPR supports kinds of FPGA architecture with different LE count, CLB count or channel width. It can place the design with SA algorithm [1] and route it for the normal FPGA architecture.

A power model is mentioned in [18] in terms of dynamic power, short-circuit power and leakage power. The activity estimator determines the switching activities inside the design. The transition density model of probabilistic techniques is used in the activity generation step. [19] merges power model and the latest version VPR. We develop enhancement of this CAD software, named Sleep Region VPR (SR-VPR), to support FPGA design with sleep modules.

5.1 SR-VPR

SR-VPR can treat multiple circuits as different modules. We also modify VPR basing on our SR place algorithm. It can get SR setting parameters, such as $N_{SR-size}$. After placement, SR-VPR routes internal connection. The output of SR-VPR describes area, critical path delay, circuit placement, routing information and power consumption. Power-off register setting in each SR can also be generated in the output file.

VPR can auto-size the FPGA basing on benchmark circuits. But to support the SR based FPGA architecture, we enhance this feature to set FPGA size as the minimal times of the SR size. In SR-VPR, for example, if one benchmark contains 7 CLBs, when $N_{SR-size}$ is 3*3, the FPGA will be sized to 3*3 for this benchmark. But if $N_{SR-size}$ is 2*2, the size of the FPGA should be 4*4. It contains 4 SRs.

VPR can also find a better solution to place and route user's design based on different architecture. But to import SR, a new challenge is needed for placement. Not only the net delay, but also the module information should be considered. If an user does not care the parameters of $Cost_{SR}$,

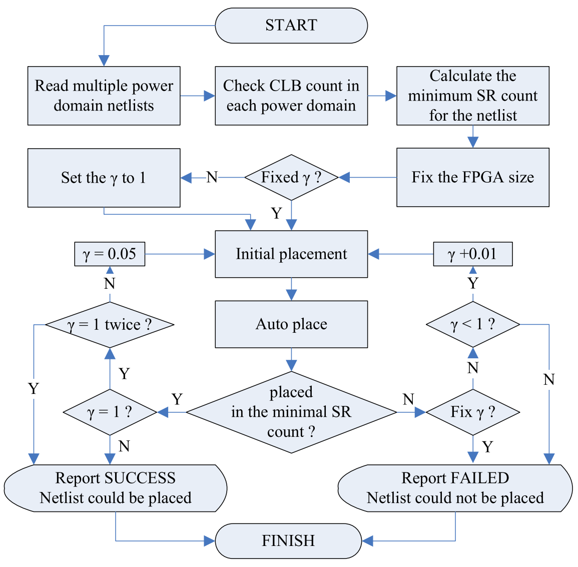

START
Read multiple power domain netlists
Check CLB count in each power domain
Calculate the minimum SR count for the netlist
Fix the FPGA size
Fixed γ ?
N
Set the γ to 1
Y
Initial placement
γ = 0.05
γ +0.01
Auto place
N
γ = 1 twice ?
Y
γ < 1 ?
Y
N
placed in the minimal SR count ?
Y
γ = 1 ?
Y
N
Fix γ ?
N
N
Y
Report SUCCESS Netlist could be placed
Report FAILED Netlist could not be placed
FINISH

Fig. 7 SR-VPR placement flow.

SR-VPR can find suitable γ to get minimum factor based on the SR. Figure 7 shows the placement flow with the parameter selecting in SR-VPR. After reading the netlist, it checks the N_{mc} and $N_{CLB}[1..p]$. When minimum SR count for the circuit has been calculated, required FPGA size is fixed. Then, SR-VPR checks whether γ is fixed. If so, it does the placement and gives out the result. If user does not give the desired value, a placement by only using SR cost is processed to check whether current setting could support SR placement successfully. After that, a loop is used for γ increasing from a very small initial value. We set initial value of γ to 0.05 based on former experiment. When γ grows bigger enough to place successfully, SR-VPR starts routing.

6. Experimental Results

6.1 Conditions for Experiments

To compare the performance between different cost functions, 15 MCNC benchmarks are placed and globally routed. In .blif netlist, CLB consists of 10 LEs, and each LE has one 4-LUT and one flip-flop. The intelligent FPGA Architecture Repository (iFAR) [20] contains accurate area and timing estimates architecture file for the logic and routing of varied island-style FPGA architectures. We use 45 nm technology file in our experiment. The routing parameters are set as Fc_{in}=0.25, Fc_{out}=0.1, Fs=3, and segments of length L=4 in this file. Based on our experiments, best result can be get when $N_{SR-size}$ is 4*4 for less area and higher performance. So, following results of coarse-grained are based on this $N_{SR-size}$. The same as VPR, minimal transistor count is used for area comparison. The routing channel width is set to 1.2 times of the minimum channel width required to route each of the benchmark circuits. IO PAD capacity is 2. To get accurate result of power consumption, we use HSPICE to generate power parameters in FPGA architecture file. Compatible with the current popular process technology, 45 nm Predictive Technology Model (PTM) [21] is used. We set VDD to 1.0 V.

To check the performance of our coarse-grained architecture and placement algorithm, we compare the results both on single module and double modules in this section by using 15 MCNC benchmarks. By comparing the results of using different FPGA architectures, Sect. 6.2 analyzes the advantages of coarse-grained power gating. Based on this architecture, VPR and SR-VPR are compared to find the impact of the new placement algorithm in Sect. 6.3.

6.2 Architecture Comparison

All FPGA architectures discussed here are based on island style [1]. The normal FPGA architecture which is used in original VPR environment does not have sleep transistors and PCHMs. Fine-grained and coarse-grained power gating architectures are based on the normal FPGA architecture with sleep transistors and PCHMs. Each CLB has a P-MOS sleep transistor and a PCHM in fine-grained architecture, while all the CLBs in an SR of coarse-grained architecture can share a P-MOS sleep transistor and a PCHM. Before comparing VPR and SR-VPR in detail, we need to find whether coarse-grained power gating architecture has advantages compared with others. A comparison is shown in Table 2 for different FPGA architectures. We apply the suitable placement algorithm for each FPGA architecture. Original VPR placement algorithm is used for the normal and fine-grained FPGA architecture. While, the coarse-grained power gating architecture needs more support of the placement to put CLBs into minimum count of SRs for reducing the power consumption maximally. So, we use SR-VPR for coarse-grained power gating FPGA in this test.

Table 2 The comparison between different FPGA architectures.

	FPGA architecture		
	Normal	Fine-grained	Coarse-grained
power on CLB percent	100%	70.6%	73.6%
area of logic	4458829	4501671	4480993
area of sleep transistor	-	139552 (3.1%)	53772 (1.2%)
critical path delay (s)	4.7E-7	4.75E-7	4.81E-7
placement time (s)	55.69	55.83	55.86 (196.26)
routing time (s)	688.4	699.5	612.3
power (W)	6.29E-2	5.58E-2	5.62E-2

To compare the architectures, we set the CLB count of normal and fine-grained FPGA architecture the same as coarse-grained architecture. Table 2 shows average value of power on CLB percent, CPU time of placement and routing, chip area, critical path delay and power consumption by using different FPGA architecture. FPGA in this table is sized to minimal size of each benchmark automatically. Usually, the count of CLBs which are used for the benchmarks does not equal a multiple of $N_{SR-size}$. When an SR has both used and unused CLBs, the unused CLBs in this SR cannot be powered off. In our experiments, 29.4% unused CLBs can be powered off by fine-grained power gating, while in coarse-grained power gating architecture, this value can reach 26.4%. Therefore, coarse-grained power gating FPGA architecture has a little larger power consumption (0.71% on average) than fine-grained one.

Area results of different FPGA architectures are shown in the count of minimum transistor. Including CLBs, PCHMs and routing area, the logic area increasing of coarse-grained FPGA architecture is less than 0.5% compared to the normal architecture. When we merge the sleep transistors which are used by each CLB in the same SR, the area of sleep transistors can be reduced [22]. This is one of advantages in coarse-grained power gating architecture. Since there is no sleep transistor in a normal FPGA, we skip the area for it. 3.1% area is overhead by using sleep transistors in fine-grained architecture. But in coarse-grained architecture, this area overhead can be reduced to 1.2%. In other words, the area of sleep transistors can be reduced 61.3% by using coarse-grained power gating architecture compared to fine-grained one. When considering the area reduction of less PCHMs, 2.3% FPGA area can be re-

Table 3 The FPGA power consumption result with one module.

	IO PAD usage	CLB usage	SR usage	γ	Placement time (s)		Routing time (s)		Area		Critical path delay (s)		Power (W)	
					VPR	SR-VPR	VPR	SR-VPR	VPR	SR-VPR	VPR	SR-VPR	VPR	SR-VPR
alu4	11%	20%	25%	0.08	23	23	273	308	7.6E+06	7.6E+06	3.7E-09	3.6E-09	0.123	0.091
apex2	21%	24%	31%	0.05	28	30	473	285	8.1E+06	8.2E+06	4.3E-09	4.3E-09	0.116	0.081
apex4	15%	19%	22%	0.11	24	24	335	378	8.0E+06	7.9E+06	4.2E-09	4.2E-09	0.101	0.068
clma	75%	79%	83%	0.08	217	208	1483	1711	9.1E+06	9.3E+06	6.9E-09	6.6E-09	0.102	0.097
diffeq	54%	16%	19%	0.07	24	24	118	117	6.4E+06	6.4E+06	4.3E-09	4.5E-09	0.093	0.058
ex1010	10%	74%	75%	0.07	126	126	5298	3405	1.1E+07	1.1E+07	4.9E-09	5.2E-09	0.111	0.095
ex5p	37%	16%	17%	0.05	22	24	122	126	7.6E+06	7.7E+06	4.2E-09	4.2E-09	0.098	0.061
frisc	71%	42%	44%	0.05	69	73	1098	953	8.3E+06	8.5E+06	8.2E-09	8.5E-09	0.072	0.047
misex3	15%	20%	22%	0.06	23	23	224	269	7.3E+06	7.6E+06	4.1E-09	3.5E-09	0.109	0.086
pdc	29%	59%	61%	0.05	91	91	2429	2034	9.4E+06	9.6E+06	5.5E-09	5.2E-09	0.095	0.081
s298	5%	16%	19%	0.07	19	20	159	205	7.6E+06	7.6E+06	5.2E-09	5.5E-09	0.075	0.039
s38417	70%	60%	64%	0.06	96	97	73	113	6.8E+06	6.7E+06	4.6E-09	4.8E-09	0.099	0.086
seq	40%	24%	25%	0.05	32	32	306	360	8.3E+06	8.1E+06	4.1E-09	3.5E-09	0.117	0.101
spla	32%	46%	47%	0.07	65	66	1122	1025	8.8E+06	8.6E+06	4.6E-09	4.9E-09	0.108	0.079
tseng	91%	14%	14%	0.07	23	22	66	59	6.3E+06	6.3E+06	4.4E-09	4.9E-09	0.095	0.055
average	38%	35%	38%	0.07	58.8	59.0	905	757	8.00E+06	8.05E+06	4.88E-09	4.89E-09	0.101	0.075
percent					100%	100.2%	100%	73.6%	100%	100.6%	100%	100.1%	100%	74.5%

Table 4 The FPGA power consumption result with two modules.

		placement time (s)		routing time (s)		Area		Critical path delay (s)		Power (W)			
M1	M2	VPR	SR-VPR	VPR	SR-VPR	VPR	SR-VPR	VPR	SR-VPR	VPR	SR-VPR	M1ON	M2ON
alu4	alu4	52	347	376	353	1.05E+07	1.05E+07	3.61E-09	3.94E-09	0.173	0.125	0.099	0.101
apex2	alu4	60	216	444	442	1.11E+07	1.10E+07	4.01E-09	4.35E-09	0.158	0.122	0.098	0.092
apex4	alu4	53	414	543	619	1.07E+07	1.10E+07	4.10E-09	3.92E-09	0.148	0.114	0.087	0.095
clma	alu4	278	1161	2009	1855	1.26E+07	1.28E+07	7.15E-09	6.90E-09	0.128	0.115	0.103	0.077
diffeq	alu4	64	190	312	186	1.03E+07	1.05E+07	4.26E-09	4.41E-09	0.137	0.093	0.073	0.084
ex1010	alu4	183	894	3759	3400	1.49E+07	1.50E+07	4.91E-09	4.83E-09	0.150	0.134	0.117	0.094
ex5p	alu4	53	175	492	479	1.04E+07	1.04E+07	3.92E-09	4.10E-09	0.147	0.102	0.079	0.088
frisc	alu4	123	271	1249	1076	1.11E+07	1.12E+07	8.47E-09	8.40E-09	0.096	0.067	0.057	0.053
misex3	alu4	52	180	325	356	1.08E+07	1.03E+07	4.01E-09	3.53E-09	0.154	0.132	0.102	0.110
pdc	alu4	146	373	1965	1731	1.30E+07	1.31E+07	5.24E-09	5.16E-09	0.136	0.114	0.097	0.083
s298	alu4	47	311	443	457	1.03E+07	1.05E+07	5.90E-09	6.04E-09	0.105	0.063	0.047	0.060
s38417	alu4	131	748	353	458	1.04E+07	1.04E+07	4.48E-09	4.40E-09	0.145	0.125	0.103	0.107
seq	alu4	64	419	494	424	1.11E+07	1.10E+07	3.69E-09	3.61E-09	0.172	0.143	0.118	0.106
spla	alu4	112	398	1172	979	1.20E+07	1.16E+07	4.74E-09	4.69E-09	0.141	0.117	0.099	0.087
tseng	alu4	63	127	254	248	1.04E+07	1.07E+07	4.38E-09	4.38E-09	0.139	0.097	0.076	0.085
average		99	415	946	871	1.13E+07	1.13E+07	4.86E-09	4.84E-09	0.142	0.111	0.090	0.088
percent		100%	420%	100%	92%	100%	100%	100%	99.7%	100%	78%	64%	62%

duced totally. Our SR-based placement algorithm does not give heavy burden to the CPU during the placement when fix γ, but when it automatically detects γ, CPU time is almost 4 times of VPR. Chip performance is not affected much when using SR-VPR as shown in the row of critical path delay.

From these results, we could see that coarse-grained power gating FPGA architecture has less power consumption than normal architecture. It also has less area (2.3%) and only 0.71% more power consumption than fine-grained power gating architecture by using the proposed SR-VPR which can place the CLB into minimum count SRs. That is the reason why we pay attention to coarse-grained architecture.

6.3 Placement Algorithm Comparison

For detailed evaluation of our proposed SR-VPR, we execute placement by VPR and SR-VPR using coarse-grained power gating FPGA architecture. Table 3 and Table 4 show the results and comparisons when single module and two modules are used respectively. In the experiment for Table 3, we fix the FPGA size for all 15 benchmark in 24*24 which is minimum FPGA size of '*tseng*'. 36 regions are used for benchmark placement. We can check the performance of SR-VPR compared with VPR when the benchmark is treated as one sleep module.

The usage of IOPAD, CLB and SR are list in this table. When the usage of CLB is less than the usage of SR, it means that there are unused CLBs in one SR. Different circuit has different minimum γ, the range is from 0.05 to 0.11, we could find that γ is 0.07 on average based on the last average line. The CPU time increment of placement is 0.2%, but routing time is decreased for 26.4%. Without affecting chip area and performance (critical path delay), 25.5% power consumption can be reduced. Note that the area of sleep transistors is not included in the chip area when compare these two algorithm by using the same FPGA architecture. Compared with the result of Table 2 which use minimum FPGA size based on each benchmark, SR-VPR performs better when FPGA usage is lower.

A most highlight characteristic of SR-VPR is that it can place CLBs of different modules into different SRs. VPR can not separate CLBs in different modules into different regions as shown in Fig. 5 (b). We performed two modules experiments by using fifteen MCNC benchmarks as the first module (M1) and using *alu*4 as the second module (M2). Table 4 shows simulation results based on M1 and M2. CPU time of placement is 4.2 times of VPR for chose the best γ, while routing time is 92%. Of course, the placement time can be reduced when we use a fixed γ. By using coarse-grained power gating method, area of PCHM is almost ignored when FPGA chip grows larger. The power consumption in different states are also compared. Assuming the power of normal un-gated FPGA architecture is 100% when using VPR, by gating the unused SRs, 22% power is reduced during two modules' working. Different modules are powered on during "M1ON" and "M2ON" states. These two states mean M1 or M2 is power on individually. The power consumption is reduced about 36% and 38% during these power states.

Because two placement methods are compared based on coarse-grained power gating architecture, and the size of CLB is pre-defined in the architecture file, the area difference is affected by the channel width and SB size. Only 0.6% area is increased in Table 3 and un-increased in Table 4. That means the placement algorithm in VPR will not increase a huge area of routing resource. The critical path delay is also shown in Table 3 and Table 4 without obvious changing.

7. Conclusion

We have proposed a new low power FPGA architecture and its placement algorithm which is incorporated in top-down design method with sleep modules. The CLBs for the user circuit are stuffed in minimal sleep regions. As a result power consumption is reduced by 22% when circuit is in working states, and it can be saved by 38% when sleep module is in sleep state.

Acknowledgments

This work was partly supported by JSPS KAKENHI 11515400, Program for Fostering Regional Innovation (Global Type), MEXT and the Core Research for Evolutional Science and Technology of Japan Science and Technology Agency (JST) in Japan.

References

[1] V. Betz, J. Rose, and A. Marquardt, Architecture and CAD for Deep-Submicron FPGAs, Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[2] Altera, Stratix IV Device Handbook. Altera., 2011. http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf

[3] Xilinx, Virtex-5 user guid. Xilinx Co., May 2010. http www.xilinx.com/support/documentation/user_guides/ug190.pdf

[4] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, and T. Tuan, "A dual-vdd low power fpga architecture," Proc. International Conference on Field Programmable Logic and Applications, pp.145–157, 2004.

[5] F. Li, Y. Lin, and L. He, "Field programmability of supply voltages for fpga power reduction," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.26, no.4, pp.752–764, April 2007.

[6] S. Ishihara, M. Hariyama, and M. Kameyama, "A low-power fpga based on autonomous fine-grain power-gating," Proc. 2009 Asia and South Pacific Design Automation Conference, ASP-DAC '09, Piscataway, NJ, USA, pp.119–120, 2009.

[7] C.Q. Tran, H. Kawaguchi, and T. Sakurai, "95% leakage-reduced fpga using zigzag power-gating, dual-vth/vdd and micro-vdd-hopping," Asian Solid-State Circuits Conference, 2005, pp.149–152, Nov. 2005.

[8] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, Low Power Methodology Manual: For System-on-Chip Design, Springer Publishing Company, 2007.

[9] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, and T. Tuan, "Reducing leakage energy in fpgas using region-constrained placement," Proc. ACM Intl. Symp. Field-Programmable Gate Arrays, pp.51–58, 2004.

[10] C. Li, Y.P. Dong, and T. Watanabe, "A novel low power fpga architecture," FIT2010 of IPSJ, pp.65–68, Sept. 2010.

[11] C.E. Cheng, "RISA: Accurate and efficient placement routability modeling," 1994 IEEE/ACM International Conference on Computer-Aided Design, pp.690–695, Nov. 1994.

[12] C. Li, Y.P. Dong, and T. Watanabe, "New power-efficient fpga design combining with region-constrained placement and multiple power domains," 9th IEEE International NEWCAS Conference, pp.69–72, June 2011.

[13] V. Betz, T. Campbell, W. Fang, I. Kuon, J. Luu, A. Marquardt, J. Rose, and A. Ye, VPR and T-VPACK User's Manual, July 2009. http://www.eecg.utoronto.ca/vpr/VPR_5.pdf

[14] S. Yang, "Logic synthesis and optimization benchmarks user guide version 3.0," 1991.

[15] U.o.C. Berkeley, Berkeley Logic Interchange Format, Feb. 2005. http://www.cs.uic.edu/~jlillis/courses/cs594/spring05/blif.pdf

[16] P. Jamieson and J. Rose, "A verilog rtl synthesis tool for heterogeneous fpgas," Field Programmable Logic and Applications, 2005. International Conference on, pp.305–310, Aug. 2005.

[17] B.L. Synthesis and V. Group, ABC: A System for Sequential Synthesis and Verification, 2007. http://www.eecs.berkeley.edu/~alanmi/abc/abc.htm/

[18] K.K.W. Poon, S.J.E. Wilton, and A. Yan, "A detailed power model for field-programmable gate arrays," ACM Trans. Des. Autom. Electron. Syst., vol.10, pp.279–302, April 2005.

[19] P. Jamieson, W. Luk, S. Wilton, and G. Constantinides, "An energy and power consumption analysis of fpga routing architectures," Field-Programmable Technology, 2009. FPT 2009. International Conference on, pp.324–327, Dec. 2009.

[20] intelligent FPGA Architecture Repository. http://www.eecg.utoronto.ca/vpr/architectures

[21] http://ptm.asu.edu

[22] A. Bsoul and S. Wilton, "An FPGA architecture supporting dynamically controlled power gating," 2010 International Conference on, Field-Programmable Technology (FPT), pp.1–8, Dec. 2010.

Ce Li was born in Liaoning, China, in 1982. He received the B.E. degree in Electrical Engineering from Dalian University of Technology in 2004. Then, He received the M.E. degree of System LSI in Waseda University in 2007. Before he became a doctor in Waseda University in 2009, he joined VIA Corp. Where he got familiar with X86 computer architecture and many low power technologies in ASIC design. His current research interests are Computer and FPGA Architecture, Computer Added Design Methodology, Processor Design, and FPGA Application.

Yiping Dong was born in Jiangsu pref., China, in 1983. He received the B.E. degree in electronics and engineering from Southeast University, China in 2006 and M.S. degree in Graduate School of Information, Production and System, Waseda University, Japan. Currently, he is a Ph.D. candidate in Graduate School of Information, Production and System, Waseda University, Japan. He is a member of RISP of Japan. His research interesting includes Networks on Chips, Neural network and low power VLSI architecture.

Takahiro Watanabe was born in Ube city, Yamaguchi Pref., Japan, in 1950. He received the B.E. and the M.E. degrees in Electrical Engineering from YAMAGUCHI University in 1974 and 1976, respectively, and the Dr. of Eng. from Tohoku University, in 1982. He joined Research and Development Center of TOSHIBA Corp. in 1979, where he worked in the field of LSI design automation as a Senior Research Scientist, and he was also responsible for the research group of circuit design technology. In August 1990, he joined Yamaguchi University as an Associate Professor of the Depr. of CSSE, and in April 2003 he joined Waseda University as a Professor of Graduate School of Information, Production and Systems. His current research interests are EDA algorithms, Design Methodology, Processor Design, FPGA Application, MPSoC and NoC Design. He is a member of IPSJ, JSAI, RISP and IEEE.