Performance-Driven Placement for Design of Rotation and Right Arithmetic Shifters in Monolithic 3D ICs

Hao Zhuang, Jingwei Lu, Kambiz Samadi†, Yang Du† and Chung-Kuan Cheng
Department of Computer Science & Engineering, University of California, San Diego, CA, 92093, USA
†Qualcomm Research, San Diego, CA, 92121, USA
{hao.zhuang, jlu}@cs.ucsd.edu, {ksamadi, ydu}@qti.qualcomm.com, ckcheng@ucsd.edu

***Abstract*—Recent advances in three-dimensional integrated circuits (3D-ICs) offer a new dimension of design exploration at traditional physical architecture of datapath components. The emerging monolithic inter-tier vias (MIVs) provides more advantages over through-silicon vias (TSVs) in terms of higher integration density and lower design overhead. In this work, we develop a performance-driven framework which uses simulated annealing to produce gate-level 3D placement layout for rotation shifter and right arithmetic shifter design. Compared to the optimum 2D layout, the critical path of our solution is much shorter with limited overhead on total wirelength. Our work indicates that by gate-level 3D-IC integration, the new physical dimension can be well leveraged with improvement on both performance and power of shifter design.**

I. Introduction

SHIFTERS are indispensable datapath components in the MPU and ASIC, e.g., floating-point units and encryption units due to its efficient bitstream operation and rotation. In contrast to the works with various physical optimization methods for adders, multipliers, and dividers, there are relatively less prior research focuses on layout optimization of shifters, mainly due to its simple logic functionality and regular structure. Nevertheless, as a basic datapath component in digital logic design, shifters has a broad spectrum of application and could impact the system performance in a larger scale. Moreover, the wiring inside each shifter module is quite dense. In order to resolve such bottleneck on the overall design performance, improvement on timing and power behaviors of shifters becomes an important subject.

During the past decade, the development of 3D-ICs has offered practical solutions [1]–[3] to many existing problems in the current IC designs. Insertion of vertical connections, such as the through-silicon vias (TSVs) proposed at early years, greatly shortens the distance between connected modules thus reduce the total wirelength and dynamic power, improve the routability and timing behavior. However, the huge dimensions of TSVs also induce area overhead [4]. Recently, emerging advances of monolithic inter-tier vias (MIVs, Fig. 1) largely reduce the physical dimensions of the vertical connections to be of only metal-via sizes. The high-density integration makes monolithic 3D-IC a promising solution [5], [6] to cope with interconnect-limited 2D-ICs, where most of the problems are essentially caused by the high interconnect density at gate level.

Previous researchers mainly focus on optimizing either the logic architecture or the physical layout of the shifters to improve the performance and power. Hillebrand *et al.* [7] proposed an approach to half the wirelength of a barrel shifter using permutation of cell positions. Zhu *et al.* [8] discussed the opportunities of architecture enhancements by logic synthesis and integer linear programming (ILP) based synthesis, where both the wire load on the critical path and the switching probability at inter-stage wires are reduced to improve timing and power. All the above approaches are based on the traditional 2D physical structure, where the narrow design space is mostly boxed by the dense wires and long critical path.

In this paper, we propose to optimize the timing and power behaviors of shifter layout under a monolithic 3D-IC physical structure.

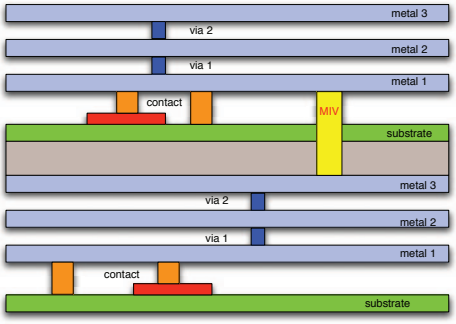

Fig. 1. A sample monolithic 3D with MIV and three layers per tier

Similar to [9], our 3D stochastic approach is based on a simulated-annealing framework, which efficiently generates placement solutions with comparable quality to that of the optimal solution. We develop a stochastic software tool for placing modules of rotators and right arithmetic shifters. We validate our approach by experiments of placing the circuits of shifters of different scales. Moreover, we provide timing and power predictions based on the propagation path length of each shifter testcase as well as a well-defined RC model for monolithic 3D-ICs at 65nm technology node. We also validate our optimized shifter designs in SPICE using $45nm$ NANGATE open source library.

Our work shows the feasibilities and benefits of improving the physical layouts of shifter architecture using monolithic 3D-ICs, while it further indicates the potential improvement on other larger datapath components by stochastic optimization. The rest of the paper is organized as follows. In Section II, we introduce the background knowledge and related works in literature regarding optimization of shifter layout at 2D- and 3D-scope. In Section III, we demonstrate our approach of stochastic placement on rotation and arithmetic shifters, respectively. Timing and power analysis of the produced layouts are discussed in Section IV. We validate the algorithm in Section V with promising experimental results. Finally, we conclude and discuss future works in Section VI.

II. Problem Statement

A. Rotator and Arithmetic Shifter

We focus on rotation shifter (rotator) and right arithmetic shifter to demonstrate our method. A shifter takes N-bit binary data word $D[N-1:0]$ with n-bit control word $S[n-1:0]$, then produces an output word $Z[N-1:0]$, where $n = log_2 N$. The binary value of control signal is $|S| = \sum_{i=0}^{n-1} 2^i S[i]$. Fig. 2 shows an example of a 8-bit MUX-based rotator with a linear ordered indices of MUXes in each logic level (linear order design). There is a very long path for

978-1-4799-3051-7/13/$31.00 ©2013 IEEE

rotation operation Rot,

$$\begin{aligned} Z[N-1:0] &= Rot(D[N-1:0], S[n-1]) \\ &= (D[|S|-1:0, D[N-1:|S|]) \end{aligned} \quad (1)$$

In arithmetic shifter (Fig. 4, which contains some dummy cells for regularity) with linear order design, there is no wrap-around long wire as shown Fig. 2. The functionality here is right arithmetic shifting while extending most significant bits (MSB).

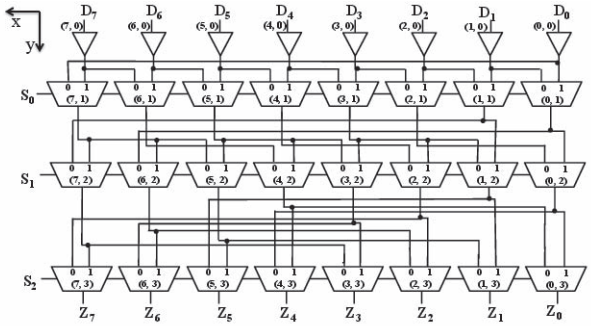

Fig. 2. 8 bits MUX-based rotator 2D ICs with linear order design

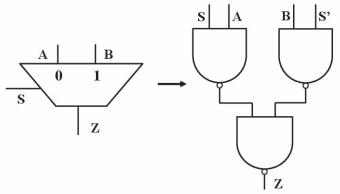

Fig. 3. The gate level implementation of MUX in Fig. 2

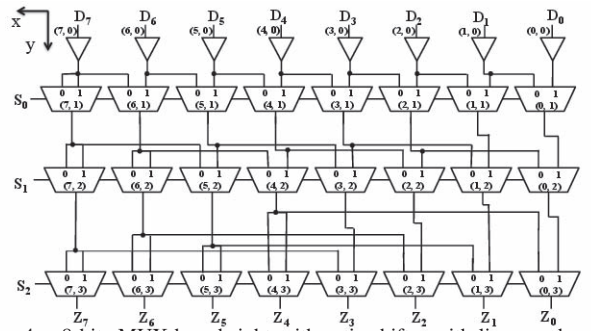

Fig. 4. 8 bits MUX-based right arithmetic shifter with linear order design

B. 2D ICs, Monolithic 3D ICs and Shifter Folding

In 2D ICs, the wires of shifter need to go across many cell widths to reach the MUX cells in the corresponding position at the following logic level. With the help of MIVs in 3D ICs, the wires can utilize the vertical "short cuts" and reach the target cell instead. Fig. 5 shows that the shifter is folded into 3D by cutting the bits in each level equally with N/L where L is the number of layer, and assigned them in multiple layers. Fig. 5 illustrates the bit slice folding strategy similar to the Figure 3(c) of [10].

C. Optimization via Cell Permutations and ILP

Zhu *et al* [8] showed cell permutations can reduce long wire length in rotator. The method for 2D ICs cell permutation is detailed in [8], which permutes the MUX cell in the same level $y = 0 \rightarrow n-2$. We extend the ILP from [8] to 3D scenario, and use *Gurobi* Package [11] to solve the ILP problem.

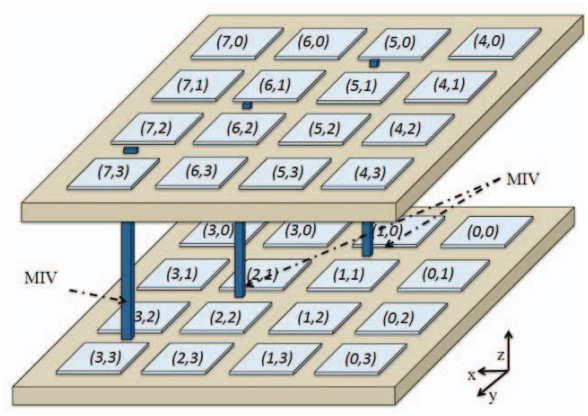

Fig. 5. 8 bit MUX-based shifter placement in 3D ICs by folding linear order shifter design

1) Extension of ILP [8] to M3D ICs : First, we model MIVs as vertical interconnects. W_{MUX} is the unit width of a MUX cell. αW_{MUX} is the wire length of MIV connecting two adjacent layers, where α is a ratio for MIV over the width of MUX cell.

The formulation is described as follows. $s^l_{i,j,k} \in \{0,1\}$: 1 if and only if logic cell i is placed at physical position of $x = j$ and $z = k$ at logic level $y = l$. The physical cell is occupied by one logic cell, so the following constraints should be met, $\sum_{i=0}^{N-1} s^l_{i,j,k} = 1,\ (0 \le j \le N/L-1,\ 0 \le k \le L-1,\ 0 \le l \le n-1)$; $\sum_{j=0}^{j=N/L-1}(\sum_{k=0}^{k=L-1} s^l_{i,j,k}) = 1,\ (0 \le i \le N-1,\ 0 \le l \le n-1)$. The span length of the path from (i_1, l) to $(i_2, l+1)$, where l is the logic level index in shifter, is calculated as follows,

$$d^{(i_1,l)\rightarrow(i_2,l+1)} = d_x^{(i_1,l)\rightarrow(i_2,l+1)} + \alpha d_z^{(i_1,l)\rightarrow(i_2,l+1)} \quad (2)$$

where cell i_2 at logic level $l+1$ is one fan-out node of cell i_1 at logic level i. d_x is the number of MUX cells a wire goes across in x-direction. d_z is the number of MUX cells (scaled by α) a wire goes across in z-direction. Then, d is the number of MUX cells a wire go across in the 3D layout from (i_1, l) to $(i_2, l+1)$.

$$d_x^{(i_1,l)\rightarrow(i_2,l+1)} = |\sum_{k=0}^{L-1}\sum_{j=0}^{N/L-1} j \cdot s^l_{i_1,j,k} - \sum_{k=0}^{L-1}\sum_{j=0}^{N/L-1} j \cdot s^{l+1}_{i_2,j,k}| \quad (3)$$

$$d_z^{(i_1,l)\rightarrow(i_2,l+1)} = |\sum_{j=0}^{N/L-1}\sum_{k=0}^{L-1} k \cdot s^l_{i_1,j,k} - \sum_{j=0}^{N/L-1}\sum_{k=0}^{L-1} k \cdot s^{l+1}_{i_2,j,k}| \quad (4)$$

Along a path from input bit a at level 0 to output bit b at level $n-1$, we sum up $d^{(i_1,l)\rightarrow(i_2,l+1)}$ and obtain $T_{path_{a\rightarrow b}}$. T_{path} is the set that contains $T_{path_{a\rightarrow b}}$ for all possible combinations (a, b), which form paths in a shifter. The goal of ILP here is to minimize the maximum value of these T_{path},

2) Timing Driven Placement and Simulated Annealing Methods: Compared to ILP, simulated annealing method is more scalable. To deal with large placement problem, Sechen *et. al.* introduced *Timberwolf* [9] based on such stochastic method. Also, for the purpose of timing improvement, several timing driven simulated annealing-based placement are investigated, e.g., [12], [13]. In the section III, we provide details of our scalable simulated annealing-based method.

III. Our Approach

In this section, we propose an approach for efficient shifter placement. Our objective is to reduce a mixed of timing and total wire length with a weighing factor γ. Marquardt *et al.* [12] and Eguro *et al.* [14] use the auto-normalizing strategy to calculate cost of a move during simulated annealing iterations. Before incorporating

such strategy in our shifter placement, we first introduce our *timing cost function* and *total wire length cost function*.

A. Timing Cost Function

The shifter has a simple structure, so that timing analysis is straightforward. We use static timing analysis (STA) during the SA iteration and obtain the slacks for each edge e. Then the summation of edges, which belong to one net net_i, form the weight of net_i. For timing driven design, we emphasize the critical path and nets, the exponential parameter θ is set to a number larger than 1. Then, the edge e's weight is

$$w_e = (1 - \frac{slack(e)}{D})^{\theta} \tag{5}$$

where $slack(e)$ in the shifter is updated via STA, the procedure is similar in [12]. Therefore, the weight of n_i is the summation of its edges' weight,

$$w_{n_i} = \sum_{e \in n_i} (1 - \frac{slack(e)}{D})^{\theta} \tag{6}$$

where e is in the set of n_i. For wire set $network$ of shifter, the total timing cost function is obtained by summation of all nets' weight,

$$W^{slack} = \sum_{n_i \in network} w_{n_i} \tag{7}$$

B. Total Wire Length Cost Function

Suppose $L(n_i)$ is the length of net n_i. The cost function of total wire length is the summation of $L(n_i)$.

$$W^{TWL} = \sum_{n_i \in network} L(n_i) \tag{8}$$

C. Auto-Normalizing Cost Function

$$\Delta_{cost} = \gamma \frac{\Delta W^{slack}}{W^{slack}_{prev}} + (1 - \gamma) \frac{\Delta W^{TWL}}{W^{TWL}_{prev}} \tag{9}$$

where W^{*}_{prev} is the value obtained from previous iteration, which is used to normalize the weights from timing and total wire length. $\Delta W^{slack} = W^{slack}_{temp} - W^{slack}$ and $\Delta W^{TWL} = W^{TWL}_{temp} - W^{TWL}$.

D. Stochastic Optimization Algorithms

We integrate the simulated annealing (SA) framework to form our 3D shifter placer. The pseudo code of our algorithm is illustrated in Algorithm 1. $Evaluation()$ uses STA to update slacks, wire lengths, calculate net weights and summation for W^{slack}, W^{TWL} and W^{slack}_{temp}, W^{TWL}_{temp}. Based on Eqn. (9), we can determine the cost of movement. In SA, we accept every improved movement greedily ($\Delta_{cost} \leq 0$). For "bad" movement ($\Delta_{cost} > 0$), the acceptance of one movement is based on probability, which has relation with current temperature T (line 16 of Algorithm 1). In addition, $\Delta C^{slack} = C^{slack}_{now} - C^{slack}_{prev}$, $\Delta C^{TWL} = C^{TWL}_{now} - C^{TWL}_{prev}$ are the weight differences between the weight after and before one inner loop, we use these to be a metric of determining the frozen status of solution. T_{start} is the temperature at the beginning, it is set as a high value and then reduced slowly to mimic annealing process. T updated in the outer while loop. The algorithm is terminated when the condition $OuterLoopExit == TRUE$, which is the solution approaching "frozen" and no further improvement happens from current status (line 27 to 34).

During the inner while loop, we permute the cells as many times as possible in the inner while-loop, of which the complexity is positive proportional to the number of cells, $O(nlog(n))$. When the "$trials$" or "$changes$" becomes too large, it triggers the termination of movement in current temperature, which means either too cold or too hot for the solution in SA (line 20-22). Set $trials$ as number of swapping attempts. $changes$ is increased after the acceptance of a tentative swap. If the ratio of $\frac{changes}{trials}$ is too small, i.e 1%, or the difference of the weights before and after the inner loop is slight different (under a threshold t), we regard it as approaching frozen status and trigger the frozen counter (line 27). When the frozen counter is larger than $Threshold$, we terminate the algorithm.

Algorithm 1 Stochastic Optimization of Shifter Placement

Require: Conventional Shifter Placement solution x
Ensure: x_{opt}

1: $T = T_{start}$; $frozen\_count = 0$
2: $R = max\{\#MUXCells \times R', 5\}$
3: $C = max\{\#MUXCells \times C', 500\}$
4: $C^{slack}_{prev} = C^{TWL}_{prev} = C^{slack}_{now} = C^{TWL}_{now} = 0$
5: $InnerLoopExit = OuterLoopExit = FALSE$
6: $[W^{slack}, W^{TWL}] = Evaluation(x)$
7: **while** $OuterLoopExit \neq TRUE$ **do**
8: $trials = 0$; $changes = 0$
9: $[C^{slack}_{prev}, C^{TWL}_{prev}] = [C^{slack}_{now}, C^{TWL}_{now}]$
10: **while** $InnerLoopExit \neq TRUE$ **do**
11: $trials = trials + 1$
12: $x_{temp} = Swap(x_i, x_j)$, where $x_i, x_j \in$ are random picked cells to swap in current placement solution space
13: $[W^{slack}_{temp}, W^{TWL}_{temp}] = Evaluation(x_{temp})$
14: Calculate Δ_{cost} using Eqn. (9)
15: $r = Random(0.0,\ 1.0)$
16: **if** $r < min(1, e^{-\frac{\Delta_{cost}}{T}})$ **then**
17: $changes = changes + 1$
18: $x = x_{temp}$
19: $[W^{slack}, W^{TWL}] = [W^{slack}_{temp}, W^{TWL}_{temp}]$
20: **if** $tirals > R\ \&\ changes > C$ **then**
21: $InnerLoopExit = TRUE$
22: **end if**
23: **end if**
24: **end while**
25: $T = \beta T$
26: $[C^{slack}_{now}, C^{TWL}_{now}] = [W^{slack}, W^{TWL}]$
27: **if** $(|\frac{\Delta C^{TWL}}{C^{TWL}_{prev}}| \leq t$ and $|\frac{\Delta C^{slack}}{C^{slack}_{prev}}| \leq t)$ or $(\frac{changes}{trials} < 1\%)$ **then**
28: $frozen\_count = frozen\_count + 1$
29: **else**
30: $frozen\_count = 0$
31: **end if**
32: **if** $frozen\_count > Threshold$ **then**
33: $OuterLoopExit = TRUE$
34: **end if**
35: **end while**
36: $x_{opt} = x$

IV. Delay and Power Analysis

In order to make high-level evaluation, we create our 3D ICs model based on [8] MIVs in monolithic 3D ICs are treated as special vertical interconnects.

A. Layout Model

We use the logical effort method [15] for delay/power estimation of MUX-based shifter. Logical effort measures the delay of single

TABLE I
DEVICE/WIRE PARAMETERS

$2\lambda(\mu m)$: technology feature size	0.0700
$C_g(fF)$: input capacitance of minimum size inverter	0.0660
$C_a(fF/\mu m^2)$: unit area capacitance of wire	0.0611
$C_f(fF/\mu m)$: unit fringing capacitance of wire	0.0148
$C_x(fF/\mu m)$: unit coupling capacitance of wire	0.0416
h_w: electrical effort contributed by wire per column spanned	1.0300

stage in unit of τ, the delay of an ideal inverter driving an identical inverter. $D = \frac{D_{abs}}{\tau} = gh + p$, where g is the logical effort of the gate, which is the ratio of the input gate capacitance to the input capacitance of an inverter with the same unit effective resistance; h is the electrical effort of the gate, which is the ratio of load capacitance to input capacitance; p is the intrinsic(parasitic) delay of gate. For a static CMOS NAND gate, $g = \frac{4}{3}$ and $p = 2$. Therefore, the electrical effort of a NAND gate is $h = h_{\text{fanout}} + h_w l_w$. where h_{fanout} is the number of load gates and l_w is the length of the driven net which is normalized to the width of MUX cell. h_w is the electrical effort contributed by the wire per width of MUX cell spanned. Assuming the MUX cell width W_{mux} is 80λ and NAND gates in the shifter are 2X of the minimum size uniformly. In addition, based on the wire model of [16], $C_w = (C_a W_w + C_f + C_x)L_w$ where W_w, L_w is the width and length of wire, respectively. Derived from this work, suppose the wire unit of width 4λ, where λ is defined as half of the technology feature size. $h_w = \frac{(4\lambda C_a + C_f + C_x)\times 80\lambda}{\frac{4}{3}C_g \times 2}$. We treat MIV like a normal interconnect, equivalent to the $\alpha \times W_{mux}$. Therefore, using a $80\alpha\lambda$ MIV, the $h = h_{fanout} + h_w(l_w^x + l_w^y + \alpha l_w^z)$ where l_w^z is the number of layer the vertical MIV interconnect goes across.

B. Metrics

1) Delay: The overall path delay is evaluated by summing up the stage delays from input to output stage of shifter: $D_{\text{path}} = \sum_i D_i = \sum_i g_i h_i + \sum_i p_i$, where i is the stage index or logic level index. Then choose the maximum delay value among all the paths.

2) Dynamic Power: The dynamic power is calculated by summation of all the switched capacitance within the shifter, which contains both wire capacitance and gate input capacitance. The resistive RC delay of wires is negligible [16]. The input gate capacitance of NAND gate is $\frac{8C_g}{3}$, also is the capacitance of wire of one column span.

V. EXPERIMENTS AND RESULTS

In this section, two comparisons are made. The first one is to compare permutation-based optimization of 2D/3D designs rotators solved by simulated annealing (SA) and integer linear programming (ILP). The second one is to compare 2D/3D designs of the rotation and right arithmetic shifters and corresponding 2D/3D folding linear ordering shifter designs (LO). The bit widths of shifters are 32, 64 and 128 bits. The metrics are shown as follows.

"LPS ($\mu m/80\lambda$)" is the number of MUX cells that the longest path spans along x- and z-directions. Based on the estimation model of Sec. IV, "Est. Delay (ps/τ)" is the maximum overall path estimation delay D_{path}. "Est. Power (fF)" is the dynamic power and measured by total switched capacitance. In the following analysis of experiment, we call "Est. Delay" and "Est. Power" as *delay* and *power* for short.

The solutions are validated by SPICE using $45nm$ NANGATE open source library. "WD (ns)" is the critical path delay; "DP ($ns \times mW$)" is the delay power product to demonstrate the energy-efficiency after optimization using SA and M3D.

For the parameters of SA, D is the half value of LPS at the initial shifter placement, $\theta = 5$, $\gamma = 0.6$. The decreasing rate of temperature $\beta = 0.999$. The initial temperature $T_{start} = 400$. $R' = 1$, $C' = 0.01$. The device and wire parameters are shown in the Table. I. The ratio α of MIV height over MUX cell is 0.05. The algorithm is implemented in C++. We solve the ILP problem using *Gurobi* Package [11]. All the experiments are performed on a Linux workstation with an Intel Core i7-920 2.67GHz CPU and 12GB memory.

A. Permutation-based Optimizations using SA and ILP

Due to the poor scalable ILP, we simply compare the LPS's solution quality of 16 bits rotator in 2D, and 2, 4 layers of M3D.

TABLE II
LPS & CPU TIME OF 16 BITS ROTATOR OPTIMIZATIONS BY SA VS. ILP

	SA		ILP	
# layer(s)	LPS	CPU (s)	LPS	CPU (s)
1 (2D)	17.0	75.6	17.0	32.7
2 (3D)	8.05	87.1	8.05	90.1
4 (3D)	4.20	83.1	4.15	52.4

It can be found that the LPS of our proposed SA method approximately equals that of the ILP method. In this case, the runtime of ILP is acceptable; However, it does not scale well. For example, to optimize a 32 bits rotator in 2 layer with $\alpha = 0.05$, ILP spends over days to obtain the solution, while SA only take minutes.

B. Analysis of LO Design and Permutation-based Optimization (SA)

Table III shows results of 32, 64 and 128 bits rotators with LO designs and optimized solution by SA. Totally, SA vastly improves LPS after permutation solved by SA in 2D. There are 40%, 31%, 23% of LPS reductions from 32, 64, 128 bits rotators, respectively. The corresponding delay improvements are 22%, 29%, and 49%. With the increasing of shifter bits, we observe the larger timing improvements because the total wire loads along critical paths are actually reduced more and more. However, the power improvement is limited, The permutation help reduce the longest path length but increase the length of other previous short wires to balance the fixed logic connection of shifter. Besides 3D folding with MIVs reducing LPS, our SA method improves timing as well as power. Another aspect of power reduction is resulted from the 3D folding, which reduces the total wire length by providing cheap vertical interconnects. Compared to the optimized solution in 2D LO designs of 32, 64 and 128 bits rotators, there are 33%, 39%, 44% improvements in dynamic power by 2 layers M3D; there are 45%, 53% and 60% by 4 layers M3D. Our SA, compared to simple 2D LO and 3D LO folding, improves timing improvement 32% and reduces power 5% on average.

Table IV shows results of 32, 64 and 128 bits right arithmetic shifters. We do not provide the results of 2D SA here because the timing and power improvements are scarce in the 2D cases. It should be noted there are not wrap-around wires as in rotators, hence the space for improvement become smaller. The folding does help reduce LPS, but worsen the timing. It is because some previous short wires in 2D is elongated when folding to another layer, and these wires are often loaded by MUX cells along a path. The naive LO 3D folding is not a wise choice in this scenario. Then, our permutation-based optimization become an important role to compensate such delay penalty, while maintaining the power reduction. Compared to 2D LO, the power reductions by SA are 13%, 18%, and 22% using 2 layers M3D. The delay reductions by SA in 4 layers are around 18% among 32, 64 and 128 bits cases. Totally, our SA has 23% timing improvement and 5% power reduction on average compared to the 2D LO and 3D LO folding strategy. We also run the SPICE simulation and verify the trend of improvement.

TABLE III

RESULTS OF 32, 64 AND 128 BITS ROTATORS PLACEMENTS BY LO DESIGN AND SA IN 2D/3D ICS. **Specifications**: LPS ($\mu m/80\lambda$): THE NUMBER OF MUX CELLS THAT THE LONGEST PATH SPANS ALONG X- AND Z-DIRECTIONS. **Estimation model based on Sec. IV:** EST. DELAY (ps/τ): MAXIMUM ESTIMATION PATH DELAY; EST. POWER (fF): ESTIMATION DYNAMIC POWER ; CPU (S): RUNTIME OF SA OPTIMIZATION. **SPICE simulation using 45nm NANGATE library:** WD (ns): THE CRITICAL PATH DELAY; DP ($ns \times mW$): DELAY $\times$ POWER PRODUCT.

	Dimensions	Method	LPS	Est. Delay	Est. Power	CPU	WD	DP
32 bits	2D	LO	61.00	135.4	213.1	-	0.789	1.2
		SA	36.00	105.2	218.3	208.3	0.548	0.9
	3D (2 layers)	LO	29.10	80.7	153.4	-	0.422	0.5
		SA	21.10	69.0	145.5	88.3	0.351	0.4
	3D (4 layers)	LO	13.30	59.0	123.9	-	0.274	0.3
		SA	9.20	55.4	120.5	128.3	0.262	0.3
64 bits	2D	LO	125.00	276.7	705.9	-	1.711	8.6
		SA	86.00	194.3	694.1	526.8	1.189	6.7
	3D (2 layers)	LO	61.10	145.0	466.3	-	0.832	3.0
		SA	45.10	105.9	423.8	350.2	0.584	2.3
	3D (4 layers)	LO	29.30	90.5	347.0	-	0.461	1.3
		SA	21.35	79.2	328.1	288.1	0.390	1.3
128 bits	2D	LO	253.00	593.7	2451.9	-	3.812	65.3
		SA	194.00	301.1	2395.3	1085.6	2.245	47.7
	3D (2 layers)	LO	125.10	286.3	1491.8	-	1.765	20.7
		SA	103.25	207.2	1344.1	854.2	1.207	16.5
	3D (4 layers)	LO	61.30	154.9	1013.0	-	0.873	7.3
		SA	51.40	127.8	958.0	916.3	0.675	6.2

TABLE IV

RESULTS OF 32, 64 AND 128 BITS RIGHT ARITHMETIC SHIFTERS PLACEMENTS BY LO DESIGN AND SA IN 2D/3D ICS.

	Dimensions	Method	LPS	Est. Delay	Est. Power	CPU	WD	DP
32 bits	2D	LO	31.00	68.2	155.8	-	0.333	0.5
	3D (2 layers)	LO	29.05	80.7	139.6	-	0.433	0.5
		SA	16.05	68.3	135.3	149.0	0.329	0.4
	3D (4 layers)	LO	13.15	58.7	120.5	-	0.298	0.3
		SA	8.30	55.9	119.8	151.1	0.281	0.3
64 bits	2D	LO	63.00	99.5	471.2	-	0.502	2.3
	3D (2 layers)	LO	61.05	145.0	408.7	-	0.845	2.9
		SA	33.10	109.2	387.1	319.0	0.542	2.0
	3D (4 layers)	LO	29.15	90.1	332.7	-	0.467	1.3
		SA	19.10	81.2	320.8	268.9	0.438	1.4
128 bits	2D	LO	127.00	152.8	1501.9	-	0.802	10.6
	3D (2 layers)	LO	125.05	286.3	1256.6	-	1.777	19.0
		SA	74.20	187.3	1175.1	847.6	1.024	12.6
	3D (4 layers)	LO	61.15	154.5	954.3	-	0.879	7.0
		SA	42.35	125.1	905.6	838.4	0.659	6.5

VI. CONCLUSIONS AND FUTURE WORKS

The technology of monolithic 3D IC (M3D) offers VLSI design with a new physical dimension, which can be leveraged for gate-level 3D circuit design. In this paper, we propose a performance-driven placer to produce 3D rotation and arithmetic shifters. They have better timing and power performance than 2D and simple 3D folding linear order designs. We utilize logical effort method and SPICE simulation to validate the performance of these optimized designs and demonstrate the advantages of our method combined with M3D. In the future, we plan to architect other datapath components, such as adders and multipliers, into M3D.

VII. ACKNOWLEDGEMENTS

We acknowledge NSF CCF-1017864.

REFERENCES

[1] L. Cheng, L. Deng, and M. D. F. Wong, Floorplanning for 3-d vlsi design, in *ASPDAC*, pages 405–411, 2005.

[2] J. Cong and G. Luo, A multilevel analytical placement for 3d ics, in *ASPDAC*, pages 361–366, 2009.

[3] J. Cong and Y. Zhang, Thermal-driven multilevel routing for 3-d ics, in *ASPDAC*, pages 121–126, 2005.

[4] K. Yang et al., Design quality tradeoff studies for 3d ics built with nano-scale tsvs and devices, in *ISQED*, pages 1–8, 2012.

[5] S. Bobba et al., Celoncel: Effective design technique for 3-d monolithic integration targeting high performance integrated circuits, in *ASPDAC*, pages 336–343, 2011.

[6] S. Panth et al., High-density integration of functional modules using monolithic 3d-ic technology., in *ASPDAC*, pages 681–686, 2012.

[7] M. A. Hillebrand et al., How to half wire lengths in the layout of cyclic shifters, in *VLSID, 2001.*, pages 339–344, 2001.

[8] H. Zhu et al., An interconnect-centric approach to cyclic shifter design using fanout splitting and cell order optimization, in *ASPDAC*, pages 616–621, 2007.

[9] C. Sechen and A. Sangiovanni-Vincentelli, IEEE JSSC **20**, 510 (1985).

[10] G. Voicu et al., 3d stacked wide-operand adders: A case study, in *ASAP*, To Appear, 2013.

[11] Gurobi optimizer reference manual, http://www.gurobi.com.

[12] A. Marquardt et al., Timing-driven placement for fpgas, in *FPGA*, pages 203–213, 2000.

[13] W. Swartz and C. Sechen, Timing driven placement for large standard cell circuits, in *DAC*, pages 211–215, 1995.

[14] K. Eguro and S. Hauck, Enhancing timing-driven fpga placement for pipelined netlists, in *DAC*, pages 34–37, 2008.

[15] I. E. Sutherland et al., *Logical effort: designing fast CMOS circuits*, Morgan Kaufmann, 1999.

[16] Z. Huang et al., Effect of wire delay on the design of prefix adders in deep-submicron technology, in *ACSSC*, pages 1713–1717, 2000.