Three-dimensional (3D) field-programmable gate arrays (FPGAs) are expected to offer higher logic density as well as improved delay and power performance by utilizing 3D integrated circuit technology. However, because through-silicon vias (TSVs) for conventional 3D FPGA interlayer connections have a large area overhead, there is an inherent trade-off between connectivity and small size. To find a balance between cost and performance, and to explore 3D FPGAs with realistic 3D integration processes, we propose two types of 3D FPGA and construct design tool sets for architecture exploration. In previous research, we created a TSV-free 3D FPGA with a face-down integration method; however, this was limited to two layers. In this paper, we discuss the face-up stacking of several face-down stacked FPGAs. To minimize the number of TSVs, we placed TSVs peripheral to the FPGAs for 3D-FPGA with 4 layers. According to our results, a 2-layer 3D FPGA has reasonable performance when limiting the design to two layers, but a 4-layer 3D FPGA is a better choice when area is emphasized.

**key words:** three dimensional IC, 3D-FPGA, power estimation, face-down face-up stacking

1. Introduction

Field-programmable gate arrays (FPGAs) are widely used for various applications, owing to their low development cost and the low risk of programming defects after fabrication. When compared with application-specific computing architectures, however, the area, delay, and power performance of FPGAs require further improvement[1]. In FPGAs, routing resources occupy a large proportion of the available area and have a large impact on delay performance. Vertically stacking FPGA devices by employing three-dimensional (3D) integrated circuit (IC) technologies is one way to solve this problem. Designing FPGAs with 3D IC technologies is expected to have several benefits. First, 3D IC technologies provide higher logic density, which expands the available silicon area on a single FPGA package without increasing the chip size. Die size has a strong inverse correlation with yield rate, which is very important to the cost of an IC product. Second, arranging logics in 3D space can also shorten FPGA logics distances, which reduces interconnection delays. However, because of the limitation of current 3D IC processes, vertical interlayer connections have a much larger area than horizontal wires. The through-silicon via (TSV), as the main component of a vertical connection, also occupies a large silicon area. On the other hand, implementing vertical routing channels with the same connectivity as horizontal ones in a 3D FPGA would require hundreds of vertical connections in each switch block (SB) within a tile, which would result in a large area overhead. Therefore, how to utilize vertical connections efficiently is the key problem of 3D FPGA architecture. Most conventional 3D FPGAs improve delay performance by designing SBs with vertical connections, which has benefit of creating shorter connections between logic gates[2]–[4]. However, the chip area cannot be improved much because of the overhead from the large number of vertical connections.

In this paper, in order to balance cost and performance, and to explore 3D FPGA architectures with realistic 3D integration processes, we present an area-compacted 3D FPGA architecture. The contributions of this research are summarized as follows: (i) a scalable small-area 3D FPGA architecture with fewer TSVs; (ii) a complete 3D FPGA design flow based on the latest VTR7.0[5] and METIS [6] and a new power estimation tool are introduced; (iii) evaluations are performed with an available 3D IC process[7] and large-scale benchmarks from the MCNC and the VTR benchmark sets. A part of this paper was published in [8], but we have redesigned the CAD flow and improved the performance of the proposed 3D FPGA significantly.

The remainder of this paper is organized as follows. Related work is summarized in Sect. 2. Section 3 describes the novel 3D FPGA architecture. The detail of the proposed CAD flow is introduced in Sect. 4. Section 5 describes the evaluation methods and results. Conclusions are given in Sect. 6.

2. Related Works

The majority of 3D FPGAs are spatially distributed, and have the same structure of tiles arranged in the 3D space. Such 3D FPGAs commonly form interlayer connections by building 3D SBs. However, a 3D SB needs a huge number of TSVs. Since the area of a TSV is much larger than that of a horizontal metal wire, such architectures are still infeasible because of the area overhead. To reduce the number of interlayer connections, Siozios et al. designed a 3D FPGA that combines 2D and 3D SBs[3]. This approach achieved remarkable area, delay, and power improvements.
when compared with a 3D FPGA based entirely on 3D SBs, but designing an FPGA in which the tiles have large differences in structure and size is not easy. The 3D SB structure proposed by Gayasen et al. reduced interlayer connections by 51% [4]. However, the best area reduction is only 16% for a five-layer FPGA because of the area overhead of the TSVs. Xilinx released a commercial FPGA product with several FPGA dies arranged upon and connected by an interposer layer [9]. However, because the FPGA slices are not vertically stacked, the package size cannot be reduced and communication between dies is much slower than that on a single die. Therefore, this technology is considered as being 2.5D.

The development of computer-aided design (CAD) tools for 3D FPGAs is also crucial. The design flow differences between 2D and 3D FPGAs occur in processes after logic clustering. First, logic blocks (LBs) are distributed into several layers. Then, intra- and inter-layer placement and routing are performed. TPR [10], an open-source 3D FPGA placement and routing tool, was developed to handle second-step processes. However, this tool was published over a decade ago and is based on VPR 4.0. Therefore, it does not support some recent FPGA architectures. A design flow to explore minimal vertical connections has been proposed by Ababei et al. [4], but the details of their novel tools have not been described. Therefore, although 3D FPGAs are attractive, effective methods for utilizing 3D IC processes still need to be researched.

3. Proposed 3D FPGAs

3.1 3D Integration and Basic Tile Structure

Figure 1 shows the cross-sectional structure of a 3D LSI circuit fabricated using Koyanagi’s 3D integration technology [11]. Thinned upper layers are stacked face-up onto a face-up thick LSI wafer (Fig. 1 (a)). Figure 1 (b) shows the arrangement when the thinned upper layers are stacked face-down onto the thick LSI wafer. In face-down stacking, TSVs are not needed until more than two layers are used.

It is important to balance cost and performance when deciding on a 3D FPGA architecture. To treat various array sizes in a similar manner, it is important to simplify the logic tile structure. Thus, we consider a homogeneous (uniform) tile structure [12]. In this paper, the proposed 3D FPGAs are based on the homogeneous logic tile structure shown in Fig. 2.

3.2 Face-Down Stacked FPGA Architecture (Two Layer)

Figure 3 shows the details of 3D-FPGA structures using face-down stacking technology, which we call face-down FPGAs [13]. There are two layers in the architecture examined in the current research: a logic layer and a routing layer. The tiles on the logic layer have an LB and a small part of the routing resources; the tiles on the routing layer have only routing resources. The difference between conventional 3D SB and the proposed architecture is that the 3D connections are made at the LB input and output pins in the logic tile. The number of inter-layer connections within
one tile is equal to the total number of LB input and output pins. The number of LB inputs $I$ is determined by the following formula [14].

$$I = \frac{K}{2} \times (N + 1)$$  \hspace{1cm} (1)

Here, $K$ is the number of logic cell inputs and $N$ is the cluster size. The number of vertical wires per tile is $I + N$. For example, when $K = 6$ and $N = 4$, each LB has 15 inputs and 4 outputs. Therefore, the number of vertical connections per tile is 19.

We next introduce a method for determining the minimum channel width (CW) of the routing track channel for the two layers. To determine optimal routing tracks, we first set the initial CW of the logic layer to 1.5 times the number of LB input pins. Then, the areas used by the connection blocks (CBs) and SBs on the logic layer can be regarded as the routing area of the logic layer; the tile area of the logic layer is the sum of the logic and routing resources used to connect neighboring LBs. The CW of the routing layer is calculated by allocating the size of the routing area as the size of the logic layer tile area. We next perform routing. If the routing is successful, then the next trial logic layer will have its CW set to half the current one. If routing fails, the CW of the next trial logic layer will be set to twice the current one. This process is repeated until the minimum CW that can lead to successful routing is found.

By dividing routing resources into two layers, we can achieve a smaller tile. Smaller tiles allow shorter routing wires and thereby enable faster signal transport, which improves the routing performance. With this structure, the router can choose a network route on either the logic layer or the routing layer. Although this approach cannot be used with more than two layers, no TSVs are necessary because face-down stacking can be used.

3.3 Four-Layer Extension with Face-Up Stacking

By using multiple face-down FPGAs, it is possible to increase the integration scale. Figure 4 (a) shows a 3D FPGA overview using two face-down FPGAs. The routing layers of face-down FPGA1 and face-down FPGA2 are face-up stacked. Vertical connection is realized by TSV. As shown in Fig. 4 (b), TSVs exist between peripheral logic tiles and input/output blocks (IOBs). $N_{TP}$ represents the number of TSVs per IOB that share external input and output between face-down FPGA1 and face-down FPGA2. $N_{TP}$ is the same number as the architectural parameter $I_{O\text{capacity}}$ in VPR. Also, $N_{TP}$ is defined to be the number of TSVs per logic tile required for signal propagation between face-down FPGAs. However, if all wire tracks are connected with a TSV, the TSV area overhead becomes large. Therefore, $N_{TP}$ should be less than the actual channel width. In our 3D-FPGA, since the TSVs are at the periphery of the FPGA, the FPGA consists of normal logic tiles. The total number of TSVs per peripheral logic tile is

$$N_{TSV} = N_{TP} + N_{TP}.$$  \hspace{1cm} (2)

Thus, although the number of TSVs in traditional 3D-SB type FPGA is of order $O(M^2)$, that for our 3D FPGA is of order $O(M)$.

4. Design Flow and CAD tools

4.1 Design Flow

In this section, we introduce a design flow and CAD tools that can be used for designing our 3D FPGA architectures. As is done with 2D FPGAs, the circuit (in ‘BLIF’ format) is first technology-mapped with ABC [15] and then clustered with AAPack, which is included as part of VPR 7.0 [5]. Next, circuit partitioning is performed. For partitioning and placement, we modify METIS and VTR 7.0, respectively. In the routing process we use EasyRouter [16], which can deal with various routing topologies, and obtain final area and delay information. We have also developed a new power analysis tool that can analyze the power consumption of a 3D FPGA. Activity information is calculated
4.2 Partitioning

The partition processes translate the original mapped netlist into a graph description and then divides the circuit with METIS, which can efficiently group logic elements into a specified number of layers with a minimal number of interconnections. Since METIS is a graph-based partitioning tool, we generate a netlist representing a set of logical cones that have a flip flop (FF). As shown in Fig. 6, the total numbers of actual look-up tables (LUTs) on the left and right sides are 3 and 2, respectively. However, when simply representing a netlist as a set of logical cones, METIS counts the left and right LUT sizes as 4 and 3, respectively. This is because the number of logical cones handled by METIS is different from the number of LUTs actually in the netlist. To handle LUTs correctly, the weight of each logical cone is calculated as follows.

\[ W_i = \frac{1}{TN_{LUT_i}} \]

Here, \( TN_{LUT_i} \) denotes the total number of LUTs of type \( i \). By this formula, the number of LUTs in METIS matches the actual number of LUTs in the netlist. To handle LUTs correctly, the weight of each logical cone is calculated as follows.

\[ W_{l}\text{out} = 1 \times W_{LUT_1} + 3 \times W_{LUT_2} + 3 \times W_{LUT_3} = 1 \times 1 + 3 \times \frac{1}{3} + 3 \times \frac{1}{3} = 3 \]

\[ W_{r}\text{out} = 1 \times W_{LUT_1} + 2 \times W_{LUT_2} + 1 \times W_{LUT_3} = 2 \]

\[ W_{\text{total}} = 1 \times W_{LUT_1} + 2 \times W_{LUT_2} = 1 \]

**Fig. 6**  Example of how to calculate weights of logic cones.

4.3 Placement and Routing

We utilize a 3D placement tool that is based on the VPR 7.0 placer. This tool operates as follows. First, the layer allocation information is read from the clustered and partitioned netlist. Then, a conventional placement by simulated annealing is performed. During the placement process, logic modules are freely swapped within each layer. The algorithm used is the bounding box (BB) method, which focuses on minimizing the bounding-box wire length of the circuit. In order to evaluate the BB cost of a net across multiple layers, we calculate \( bb_x \) (BB distance in the \( x \) direction) and \( bb_y \) (BB distance in the \( y \) direction) for each layer, and then use the maximum values of \( bb_x \) and \( bb_y \) across all layers in the calculation of the final BB cost. The vertical connection cost is not considered at this point in the placement, for two reasons: (1) A typical 3D-FPGA CAD flow processes the vertical allocation of logic modules in the partitioning step. During the partitioning step, inter-layer connections are minimized by algorithms such as hMETIS [18]. For the placement step, logic modules are placed within each layer, and so vertical connection costs do not need to be considered. (2) When the vertical connection cost is comparable with the horizontal connection cost, we have to build a cost model for the partitioning algorithm. However, in our approach, the vertical connection cost is negligible during both the partitioning and placement steps. Of course, the final delay derived from the router certainly includes the TSV delay. In the future, in order to handle 3D-FPGAs with different 3D-VLSI processes and more layers, we will improve 3D partitioning to include the consideration of vertical cost.

Finally, routing is performed with our novel tool, EasyRouter. EasyRouter implements a pathfinder routing algorithm similar to that used in VPR; however, EasyRouter simplifies the implementation of new FPGA architectures with various routing topologies. In addition, EasyRouter combined with VLSI CAD can provide highly accurate reports on area and critical path delays for FPGA designs that are based on standard cells.

4.4 Power Estimation

4.4.1 Overview of Easy Power

We have also developed a new power estimation tool, EasyPower, for analyzing power consumption in 3D FPGAs. EasyPower cooperates with EasyRouter in this analysis. Figure 7 shows the relation between EasyPower and EasyRouter functional blocks. By using the routing resource graph (RR graph) generated by EasyRouter, EasyPower can handle the circuit and the parasitic load of the wiring. Power consumption is also calculated using the power information of the standard cell library.

EasyRouter generates an RR graph from the netlist,
placement information, and architecture file. After routing, EasyRouter outputs area, delay, and bitstream information. EasyPower uses the RR graph after routing, the bitstream, the switching activity, and information from the cell library as input. The multi-input multiplexer in the RR graph is decomposed into a tree of two input multiplexers. EasyPower also inserts a buffer to adjust the drive capability. To obtain the switching activity, the output of ACE 2.0 is used. However, since ACE 2.0 does not calculate switching information inside LUTs, switching information is calculated with EasyPower.

4.4.2 Power Calculation

The power consumption of large-scale integrated (LSI) circuitry is classified into dynamic power and static power. Dynamic power is the power consumed by charging and discharging when the transistor switches. When using the standard cell library, dynamic power consists of switching power and internal power. Switching power is the power consumed when charging and discharging the load capacitance connected to the gate. Switching power is expressed by the following equation.

\[
\text{Switching Power} = 0.5 \cdot C_y \cdot V_{dd}^2 \cdot D_y \cdot f_{clk}
\]  

(5)

In this, \(C_y\) is the load capacitance, which is mainly the sum of the input capacitance of the next-stage cell and the wire capacitance; \(V_{dd}\) is the power supply voltage; \(D_y\) is the switching activity, which is the probability of the output signal transitioning per clock cycle\[17\]; and \(f_{clk}\) is the operating frequency.

Internal power is power consumption due to charging and discharging of capacity inside the cell. This power includes the short circuit power generated momentarily between the source and drain region when the state of the transistor changes. Internal power is expressed by the following equation.

\[
\text{Internal Power} = E_Z \cdot D_y \cdot f_{clk}
\]  

(6)

where \(E_Z\) is a value obtained from the cell library data sheet and is the energy consumed by one switching. In contrast, static power (“leak power”) is the electric power consumed by a slightly flowing current in an off-state transistor. The static power for each gate is available from the data sheet.

To calculate dynamic power, circuit switching information is required. ACE 2.0 adopts a method that takes an intermediate path between simulation and the probability method. In ACE 2.0, an input waveform is automatically generated according to the set value of the input vector of primary inputs (PIs), and pseudorandom simulation is performed. From this, switching information can be estimated with higher accuracy than with the probability based method. In our tool, power is calculated based on the output from ACE 2.0.

4.4.3 Buffer Insertion

The internal power, looked up in the cell library, is a two-dimensional table consisting of signal transition time and load capacitance. Because the transition time is determined by the driving capability of each cell, EasyPower inserts a buffer to adjust the drive capability. The drive capability of the buffer is determined so as to satisfy the following expression.

\[
\frac{C_L}{C_{BUF}} \leq 4
\]  

(7)

Here, \(C_L\) is the load capacitance and \(C_{BUF}\) is output capacitance of the inserted buffer.

4.4.4 Switching Activity Inside LUTs

In EasyPower, the multiplexers of SBs, CBs, and LUTs are each decomposed into two input multiplexer trees to calculate power consumption. The problem is how to calculate the switching probability inside the LUT. ACE 2.0 calculates the switching probability of the output signal from the LUT, but since the inside of the LUT is a black box, there is no information about internal activity. Thus, because the lag-one model\[19\] produces exact switching probabilities, the switching activity of a 2-input multiplexer is obtained by

\[
P_{out} = P_A \cdot (1 - P_S) + P_B \cdot P_S
\]

\[
D_{out} = (D_B - D_A)P_S + D_A
\]  

+ \((1 - 2P_A)P_B - \frac{D_B}{2} + P_A - \frac{D_A}{2}\) \(D_S\)

(9)

where \(P_A, P_B, P_S\) denote the switching probabilities for input signals A, B, S, respectively, and \(D_A, D_B, D_S\) denote the corresponding switching activities.

4.5 Validation of Power Estimation

The accuracy of power consumption depends on the calculation model and switching information. To estimate the accuracy of EasyPower, we compare the output with that of a commercial tool, Synopsys Power Compiler. We evaluate three cases. In case 1, Power Compiler calculates power consumption based on switching activity from Synopsys VCS. In case 2, EasyPower calculates power consumption based on the activity obtained by VCS. In case 3, EasyPower calculates power consumption based on the activity obtained by ACE 2.0. The evaluation conditions are the following: LUT input \(K = 6\), cluster size \(N = 8\), \(F_s = 3\) (Wilton type), and \(F_c = 0.5\). All wire is a single line length. Physical parameters are those of a CMOS 65nm standard cell. When generating an activity, test vectors are generated so that \(P_{in} = 0.5\) and \(D_{in} = 0.2\) which are switching probability and switching activity of input vector, respectively.
Table 1 Accuracy of each power consumption (4-bit adder) [mW].

| Block | Switch Int. Leak | Total |
|-------|------------------|-------|
| Case1 |                  |       |
| SB    | 0.041 0.107 0.021 | 0.169 |
| CB    | 0.019 0.029 0.014 | 0.062 |
| LCB   | 0.030 0.095 0.042 | 0.167 |
| BLE   | 0.050 0.198 0.014 | 0.261 |
| Total | 0.140 0.429 0.090 | 0.659 |
| Case2 |                  |       |
| SB    | 0.041 0.102 0.022 | 0.165 |
| CB    | 0.019 0.028 0.015 | 0.061 |
| LCB   | 0.030 0.092 0.044 | 0.166 |
| BLE   | 0.054 0.188 0.014 | 0.256 |
| Total | 0.143 0.410 0.095 | 0.648 |
| Case3 |                  |       |
| SB    | 0.041 0.102 0.022 | 0.165 |
| CB    | 0.019 0.028 0.015 | 0.061 |
| LCB   | 0.030 0.092 0.044 | 0.166 |
| BLE   | 0.006 0.167 0.014 | 0.187 |
| Total | 0.095 0.389 0.095 | 0.579 |

In this evaluation, we use a simple 4-bit adder circuit. Table 1 shows the results in all cases. Note that since EasyPower and Power Compiler have different wire load models, wire capacitance is ignored. When case 1 is considered as the baseline, the error rate for total power is 1.1% in case 2 and 8.0% in case 3. Thus, there is little difference between cases 1 and 2. In contrast, cases 1 and 3 use different tools to obtain switching activity. Since EasyPower calculates switching activity internal to LUTs, EasyPower underestimates the power consumption of Basic Logic Element (BLE). However, other values are almost the same. This validation shows that when the activities are the same, EasyPower calculates power with high accuracy.

5. Evaluation

In this section, we compare the architecture of the proposed 3D FPGA (four-layer stacking) with that of a 3D FPGA (two-layer stacking), evaluating the area, the critical path delay, and power. An island-style 2D FPGA is used as the baseline for evaluations. An analysis of the evaluation results of the proposed 3D FPGAs is also given in this section.

5.1 Evaluation Condition

The parameter values for the target architectures are listed in Table 2. All implemented 3D FPGA architectures are homogeneous FPGAs with a LUT size of 6 and a cluster size of 8. For the routing architecture, we used a Wilton-type SB with $F_s = 3$. The $F_{cin}$ parameter was set to 0.5, which means that half of the tracks in the routing channel are connected to an LB input through a CB. We set the area of each TSV to $144 \mu m^2$ and calculate TSV capacitance using a compact multi-TSV coupling model [20] with FreePDK3D [21]. In this evaluation, we utilize the smallest delay of TSV is $47.80[ps]$ as a representative delay.

The 12 VPR benchmark circuits were selected as the evaluation test suite. Physical FPGA parameters are extracted from the 65-nm CMOS technology standard cell library. For the area calculation, logic tile areas of all architectures were taken from results synthesized with Synopsys Design Compiler F-2011.09-SP2. The total area was calculated by multiplying the number of tiles by the area of one tile and adding the total area of TSVs. All multiplexers (MUXes) in CBs, SBs, and LUTs are treated as 2-to-1 MUXes in this evaluation.

Figure 8 shows the evaluation flow. We use the conventional FPGA CAD flow, performing elaboration with ODIN II [22], technology mapping with ABC, and placement with VTR 7.0. Routing and power estimation are executed using our developed tools, EasyRouter and EasyPower. EasyPower utilizes switching information from ACE 2.0.

5.2 Results

To obtain the final area, delay, and power, we have to determine the channel width for each architecture. This process consists of three steps: circuit partitioning, minimum channel width searching, and choice of how many TSVs to use a 3D FPGA with 4 layers. First, Table 3 shows the partitioning result for each benchmark circuit (first column). The second and third columns are the number of primary inputs and outputs, respectively. The fourth column is the number of LUTs in the original circuits. The fifth column is the number of logical cones in the original circuits. The sixth and seventh columns are the number of LUTs (partitioning ratio) in face-down FPGA1 and face-down FPGA2, respectively. The eighth and ninth columns are the numbers of signals crossing between the two face-down FPGAs. For most circuits, we obtained nearly 1:1 circuit partitioning. Despite this, the number of signals crossing between the FPGAs is biased.

Next, a minimum channel width exploration was performed for the benchmark circuits. The obtained channel width of a 2D FPGA, $CW = 100$, was fixed to be 1.2 times...
Table 3  Partitioning results.

| Benchmark | PI PO | LUTs (original) | Logic cones (original) | LUTs in FPGA1 | LUTs in FPGA2 | Signals (FPGA1 → FPGA2) | Signals (FPGA2 → FPGA1) |
|-----------|------|----------------|------------------------|---------------|---------------|-------------------------|-------------------------|
| sv_chip1_ | 69   | 145            | 24,552                 | 11,775        | 11,932 (49%)  | 207                     | 163                     |
| ucsb_152 | 14   | 29             | 22,297                 | 19,289        | 11,775        | 212                     | 216                     |
| raygentop | 256  | 305            | 16,708                 | 7,088         | 8,692 (51%)   | 8,338 (49%)             | 6                       |
| sv_chip0_ | 170  | 198            | 14,181                 | 13,219        | 6,997 (49%)   | 7,203 (51%)             | 240                     |
| or1200   | 389  | 594            | 9,000                  | 3,133         | 5,724 (50%)   | 5,692 (50%)             | 1,473                   |
| mkSMAdapter4B | 199 | 205            | 8,990                  | 5,636         | 4,736 (47%)   | 5,411 (53%)             | 1,534                   |
| des_perf | 123  | 64             | 3,264                  | 1,984         | 1,600 (49%)   | 1,664 (51%)             | 0                       |
| clma     | 383  | 82             | 2,930                  | 1,141         | 1,141 (47%)   | 1,582 (53%)             | 27                      |
| s38417   | 29   | 106            | 2,433                  | 1,524         | 1,231 (50%)   | 1,225 (50%)             | 120                     |
| s38584.1 | 39   | 304            | 2,272                  | 1,416         | 1,160 (51%)   | 1,120 (49%)             | 35                      |
| rs_decoder_2 | 22 | 11              | 1,920                  | 627           | 1,409 (54%)   | 1,224 (46%)             | 155                     |
| rs_decoder_1 | 14 | 7               | 1,018                  | 524           | 554 (49%)     | 571 (51%)               | 73                      |

Table 4  Logic Block Utilization.

| Benchmark | # of LB | # of utilized LB | Utilization[%] | # of LB | # of utilized LB | Utilization[%] |
|-----------|---------|-----------------|----------------|---------|-----------------|----------------|
| sv_chip1_ | 3,249   | 3,226           | 99.3%          | 3,362   | 3,350           | 96.7%          |
| ucsb_152 | 2,809   | 2,788           | 99.3%          | 2,888   | 2,878           | 96.6%          |
| raygentop | 2,116   | 2,116           | 100.0%         | 2,178   | 2,178           | 98.3%          |
| sv_chip0_ | 1,849   | 1,778           | 96.2%          | 1,922   | 1,922           | 92.5%          |
| or1200   | 1,156   | 1,119           | 96.8%          | 1,458   | 1,428           | 97.9%          |
| mkSMAdapter4B | 1,156 | 1,120          | 96.9%          | 1,458   | 1,268           | 87.0%          |
| des_perf | 441     | 409             | 92.7%          | 450     | 409             | 90.9%          |
| clma     | 400     | 374             | 93.5%          | 450     | 383             | 85.1%          |
| s38417   | 324     | 300             | 92.6%          | 338     | 303             | 89.6%          |
| s38584.1 | 289     | 279             | 96.5%          | 288     | 280             | 97.2%          |
| rs_decoder_2 | 256 | 245             | 95.7%          | 392     | 335             | 85.5%          |
| rs_decoder_1 | 144 | 129             | 89.6%          | 162     | 143             | 88.3%          |

Fig. 9  Delays with different numbers of TSVs.

Fig. 10  Area result.

the minimum channel width. Array size is also determined for each benchmark circuit. EasyRouter performed channel width exploration for 3D-FPGAs, using the method described in Sect. 3.2. In this exploration, the channel widths of the routing and logic layers were 90 and 10, respectively. Finally, we examined how many TSVs per channel are needed for a 3D-FPGA with 4 layers. Figure 9 shows delay performance for each TSV. Since the CW per direction in the routing layer was 45, this figure shows the delay when the number of TSVs per logic tile, $TTP_T$ in Sect. 3.3, is 15, 20, 30, or 45 (full). Each delay was normalized by the delay for $TTP_T = 45$. In case of $TTP_T = 15$, routing failed for “or1200”. However, in most cases, the results were almost same for $TTP_T = 20, 30, 45$, so we decided that the number of TSVs per channel was 20.

Figure 10 shows area results normalized by the 2D-FPGA area. In this evaluation, we calculated FPGA area for each benchmark circuit. The area was obtained the total number of tiles × logic tile area + $TNTSV × TSVarea$ to be same logic capacity for each FPGA. 3D FPGAs with two layers and four layers were 43% and 68% smaller on average than the 2D FPGA. Since the total area occupied by TSVs was about 1.9%, we found that the TSV area overhead was relatively small in a 3D FPGA with 4 layers.

Figure 11 shows delay results normalized by the 2D-FPGA delay. A 3D FPGA with 2 layers exhibited an improvement of 10% on average while a 3D FPGA with 4 layer was 9% slower on average. For “ucsb 152”, a 3D FPGA with 4 layers was 2 times slower. The reason for this was the overhead of signal communication between face-down FPGAs. Meanwhile, EasyRouter does not support timing
driven routing algorithm. We have to develop timing driven mode as a future work.

Before showing power consumption, Table 4 denotes logic block utilization. In this table, the number of LB means total number of LB (logic tile). For example, when FPGA has $57 \times 57$ logic block, the number of LB is 3,249 in “sv_chip1”. Utilized LB is logic block on which logic modules are assigned actually. We defined that utilization[$\%$] = the number of utilized LB / the total number of LB $\times 100$. The number of logic block of 3D-FPGA with 4 layer increased compare with another FPGAs due to duplication among partition process as shown in Table 3. Figure 12 shows power results normalized by 2D-FPGA power. Table 5 shows that the ratio between the dynamic power and the static power for the 2-layer 3D FPGA is improved by 8% on average, while the 4-layer 3D FPGA is 15% worse for both dynamic and static power. The dynamic power increase may occur as the result of routing distance increasing between FPGAs. In addition, Table 6 shows detail analysis for each blocks, such as LB, CB, LCB, SB. Power consumption of TSV includes one of SB. As the number of logic tile increased due to expanding to 4 layers, static power increased.

Figure 13 shows area-delay products normalized by

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Block & Dynamic & Static & Total \\
\hline
SB & 0.93 & 1.02 & 0.94 \\
SB & 0.98 & 1.02 & 0.98 \\
SB & 0.94 & 1.02 & 0.94 \\
CB & 1.06 & 1.03 & 1.06 \\
CB & 1.06 & 1.03 & 1.06 \\
LCB & 1.00 & 1.00 & 1.00 \\
LCB & 1.00 & 1.00 & 1.00 \\
BLE & 0.97 & 0.97 & 0.97 \\
BLE & 0.97 & 0.97 & 0.97 \\
BLE & 0.97 & 0.97 & 0.97 \\
BLE & 0.97 & 0.97 & 0.97 \\
BLE & 0.97 & 0.97 & 0.97 \\
BLE & 0.97 & 0.97 & 0.97 \\
\hline
Total & 0.93 & 0.90 & 0.98 \\
Total & 0.93 & 0.90 & 0.98 \\
Total & 0.93 & 0.90 & 0.98 \\
Total & 0.93 & 0.90 & 0.98 \\
Total & 0.93 & 0.90 & 0.98 \\
Total & 0.93 & 0.90 & 0.98 \\
\hline
3D-FPGA & 2.14 & 1.57 & 1.11 \\
SB & 1.02 & 0.84 & 0.59 \\
SB & 1.04 & 1.03 & 0.59 \\
CB & 2.55 & 1.85 & 1.23 \\
CB & 0.93 & 0.80 & 0.37 \\
CB & 1.11 & 1.00 & 0.35 \\
LCB & 1.98 & 0.79 & 0.33 \\
LCB & 1.03 & 1.03 & 0.97 \\
LCB & 0.90 & 0.96 & 0.92 \\
BLE & 1.01 & 0.99 & 0.97 \\
BLE & 1.03 & 1.03 & 0.97 \\
BLE & 1.37 & 1.26 & 1.14 \\
Total & 0.99 & 0.95 & 0.96 \\
Total & 1.05 & 1.05 & 1.05 \\
\hline
3D-FPGA & 2.14 & 1.57 & 1.11 \\
SB & 1.02 & 0.84 & 0.59 \\
SB & 1.04 & 1.03 & 0.59 \\
CB & 2.55 & 1.85 & 1.23 \\
CB & 0.93 & 0.80 & 0.37 \\
CB & 1.11 & 1.00 & 0.35 \\
LCB & 1.98 & 0.79 & 0.33 \\
LCB & 1.03 & 1.03 & 0.97 \\
LCB & 0.90 & 0.96 & 0.92 \\
BLE & 1.01 & 0.99 & 0.97 \\
BLE & 1.03 & 1.03 & 0.97 \\
BLE & 1.37 & 1.26 & 1.14 \\
Total & 0.99 & 0.95 & 0.96 \\
Total & 1.05 & 1.05 & 1.05 \\
\hline
\end{tabular}
\caption{Power distribution for each block normalized by 2D-FPGA.}
\end{table}
2D-FPGA. A 3D FPGA with 2 layers and 4 layers showed an improvement of 49% and 65% on average, respectively. Meanwhile, Figure 14 shows power-delay products normalized by 2D-FPGA. A 3D FPGA with 2 layers exhibited an improvement of 16% on average while a 3D FPGA with 4 layer was 25% worse on average. From these results, a 2-layer 3D FPGA appears to provide reasonable area, delay and power characteristics. On the other hand, if a slight increase in delay and power overheads can be tolerated, a 4-layer 3D FPGA is very attractive in terms of area.

6. Conclusion

In this paper, we have proposed two types of 3D FPGA architecture with fewer vertical connections to allow simple die stacking. We have also constructed a design flow suitable for architecture exploration of 3D FPGAs. The first type is a two layer 3D FPGA based on face-down stacking which has 3D connections on logic block input and output pins without TSVs. We also considered face-up stacking of face-down FPGAs and placed TSVs on the periphery of FPGAs. According to our evaluation results, the 2-layer 3D FPGA provides reasonable performance when the design is limited to two layers, but the 4-layer type is a better choice when area is emphasized.

References

[1] I. Kuon and J. Rose, “Measuring the Gap Between FPGAs and ASICs,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.26, no.2, pp.203–215, 2007.
[2] M.J. Alexander, J.P. Cohoon, J.L. Colflesh, J. Karro, and G. Robins, “Three-Dimensional Field-Programmable Gate Arrays,” Proc. Eighth Annual IEEE International, pp.253–256, Sept. 1995.
[3] K. Siozos, H. Pavlidis, and D. Soudris, A Novel Framework for Exploring 3-D FPGAs with Heterogeneous Interconnect Fabric, ACM Trans. Reconfigurable Technology and Systems, vol.5, no.1, Article 4, March 2012.
[4] A. Gayasen, V. Narayanan, M. Kandemir, and A. Rahaman, “Designing a 3-D FPGA: Switch Box Architecture and Thermal Issues,” IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol.16, no.7, pp.882–893, July 2008.
[5] L. Jason, B. Vaughn, C. Ted, F.W. Mark, J. Peter, K. Ian, M. Alexander, Y. Andy, and R. Jonathon, “The Verilog-to-Routing (VTR) Project for FPGAs,” http://code.google.com/p/vtr-verilog-to-routing/
[6] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs,” SIAM J. Sci. Comput., vol.20, no.1, pp.359–392, Dec. 1998.
[7] The FreePDK3D45 Predictive 3DIC Process Design Kit, [online] available at https://www.eda.ncsu.edu/wiki/FreePDK3D45:Manual
[8] This is blind review.
[9] Xilinx, “Industry’s First 3D ICs,” http://www.xilinx.com/products/silicon-devices/3ic.html
[10] C. Ababei, H. Mogal, and K. Bazargan, “Three-dimensional Place and Route for FPGAs,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.25, no.6, pp.1132–1140, June 2006.
[11] M. Koyanagi, T. Fukushima, and T. Tanaka “Three-Dimensional Integration Technology and Integrated Systems,” Proc. ASP-DAC, pp.19–22, Oct. 2009.
[12] K. Inoue, H. Yoshio, M. Amagasaki, M. Iida, and T. Sueyoshi, “An Easily Testable Routing Architecture And Efficient Test Technique,” Proc. FPL, pp.291–294, Aug. 2011.
[13] M. Amagasaki, Q. Zhao, M. Iida, M. Kuga, and T. Sueyoshi, “A 3D FPGA Architecture to Realize Simple Die Stacking,” IPSJ Transactions on System LSI Design Methodology, vol.8, pp.116–122, Aug. 2015.
[14] V. Betz, J. Rose, and A. Marquardt, “Architecture and CAD for Deep-Submicron FPGAs,” Kluwer Academic Publishers, March 1999.
[15] Berkeley Logic Synthesis and Verification Group, “ABC: A System for Sequential Synthesis and Verification,” http://www.eecs.berkeley.edu/~alanmi/abc/
[16] Q. Zhao, K. Inoue, M. Iida, M. Amagasaki, M. Kuga, and T. Sueyoshi, “FPGA Design Framework Combined with Commercial VLSI CAD,” IEICE Transactions on Information and Systems, vol.E96-D, no.8, pp.1602–1612, Aug. 2013.
[17] J. Lamoureux and S.J.E. Wilton, “Activity Estimation for Field-Programmable Gate Arrays,” Proc. FPL, pp.1–8, Aug. 2006.
[18] N. Selvakumaran and G. Karypis, “Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization,” IEEE Trans. CAD. of Integrated Circuits and Systems, vol.25, no.3, pp.504–517, Feb. 2006.
[19] R. Marculescu, D. Marculescu, and M. Pedram, “Switching Activity Analysis Considering Spatiotemporal Correlations,” Proc. ICCAD, pp.294–299, Nov. 1994.
[20] T. Song, C. Liu, Y. Peng, and S.K. Lim “Full-Chip Multiple TSV-to-TSV Coupling Extraction and Optimization in 3D Ics” Proc. DAC, Article no.180, May 2013.
[21] FreePDK3D45, https://www.eda.ncsu.edu/wiki/FreePDK3D45
Motoki Amagasaki received the B.E. and M.E. degrees in Control Engineering and Science from Kyushu Institute of Technology, Japan in 2000, 2002, respectively. He was an engineer at NEC Micro Systems Co., Ltd. from 2002 - 2005. He received the D.E. degree from Kumamoto University, Japan, in 2007. He has been an assistant professor in the Department of Computer Science at Kumamoto University until 2007. He has been an assistant professor in the Faculty of Advanced Science and Technology at Kumamoto University since 2016. His research interests reconfigurable system and VLSI design. He is a member of IPSJ and IEEE.

Masato Ikebe received the B.E. degree in Computer Science from Kumamoto University in 2015. Further, he received the M.E. degree in Computer Science and Electrical Engineering from Kumamoto University in 2017.

Qian Zhao received his B.E. degree in the College of Automation and Electronic Engineering from Qingdao University of Science and Technology, China, in 2007. Further, he received his M.E. and D.E. degrees in Computer Science and Electrical Engineering from Kumamoto University in 2011 and 2014, respectively. He is now a postdoctoral researcher at Kumamoto University. His research interests include architecture and design methods of reconfigurable computing systems. He is a member of IEICE.

Masahiro Iida received his B.E. degree in Electronic Engineering from Tokyo Denki University in 1988. He was a research engineer at Mitsubishi Electric Engineering Co., Ltd. from 1988 to 2003. He received his M.E. degree in Computer Science from Kyushu Institute of Technology in 1997. Further, he received his D.E. degree from Kumamoto University, Japan, in 2002. He was an associate professor at Kumamoto University until 2015, and during 2002-2005, he held an additional post as a researcher at PRESTO, Japan Science and Technology Corporation (JST). He has been a professor in the Faculty of Advanced Science and Technology at Kumamoto University since January 2016. His current research interests include high-performance low-power computer architectures, reconfigurable computing systems, VLSI devices and design methodology. He is a senior member of the IPSJ, and a member of the IEICE and IEEE.

Toshinori Sueyoshi received the B.E. and M.E. degrees in Computer Science and Communication Engineering from Kyushu University in 1976 and 1978 respectively. From 1978 to 1987, he was a research associate at Kyushu University, where he received D.E. degree in 1986. From 1987 to 1989, he was an associate professor of Information System at Kyushu University. From 1989 to 1997, he was an associate professor of Artificial Intelligence at Kyushu Institute of Technology. Since 1997 he has been a professor of Computer Science at Kumamoto University. He is also a guest professor at Kyushu University. His primary research interests include computer architecture, reconfigurable systems, parallel processing. He served as Chair of the Technical Committee on Reconfigurable Systems of the IEICE, Chair of the Technical Committee on Computer Systems of the IEICE, Chair of the IEEE Computer Society Fukuoka Chapter, Chair of the IEEE CAS Society Fukuoka Chapter, and Director of the IPSJ Kyushu Chapter. Currently he also serves as Director-Elect of the IEICE Kyushu Chapter. He is a fellow of the IEICE.