Chapter 12

Ultrafine grain FPGAs with polarity controllable transistors

Xifan Tang[1,2], Pierre-Emmanuel Gaillardon[2], Ian O'Connor[3], and Giovanni De Micheli[1]

Abstract

In the quest to push further the Moore's scaling laws, intensive development efforts have been invested on seeking for alternatives to planar CMOS transistors at advanced technology nodes. In particular, recent years have witnessed the massive commercialization of the *fin-based field effect transistors* (FinFETs) at the 10, 14 and 22-nm technology nodes [1–3]. In addition to FinFET technology, several devices are currently investigated by following the trend toward 1D structures. Among them, carbon nanotubes FETs [4] and vertically stacked *silicon nanowires* FETs (SiNWFETs) [5] are promising extensions to current tri-gate FinFETs technology, which exploit the 1-D properties of their channels to exhibit superior performances. Moreover, these novel transistor technologies employ the gate-all-around (GAA) structure which can improve the electrostatic control of the channel, leading to a higher I_{ON}/I_{OFF} ratio and reduced leakage current [6].

More than the performance improvement, these novel transistor technologies present another possible direction in pushing the Moore's scaling laws: functionality-enhanced device. Especially at advanced technology nodes, transistors are strongly affected by Schottky contacts at the source and drain interfaces. As a result, transistors may operate with an ambipolar behavior, i.e., the device can exhibit n- and p-type characteristics simultaneously. Indeed, to achieve pure n- and p-type polarity, the ambipolar behavior of the devices is typically suppressed through additional process steps. However, new design methodologies [7–9] have shown attractive opportunities in controlling the ambipolar phenomenon through programmable polarity devices. By engineering the source and drain contacts and by constructing independent double-gate structures, the device polarity can be electrostatically programmed to be either n- or p-type. Such functionality-enhanced devices have been demonstrated using silicon

[1]School of Computer Science and Communication, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[2]Department of Electrical and Computer Engineering, University of Utah, USA
[3]Department of Electronic, Electrical and Control Engineering, Ecole Centrale de Lyon, France

[10,11] and carbon electronics [12,13]. In this chapter, we focus on a *double-gate* SiNWFET (DG-SiNWFET), built using a top-down fabrication flow [14].

Combining both performance improvement and device-level reconfigurability, the functionality-enhanced devices has been exploited to build ultrafine grain reconfigurable logic blocks [7]. This technology shift allows a reconfigurable *logic cell* (LC) which is realized with only seven transistors to perform many different two-input Boolean operations [7]. The cell compactness opens an opportunity in rethinking the standard reconfigurable architectures. Take the example of the traditional *field-programmable gate arrays* (FPGAs) architectural; only less than 15% of the area is dedicated to the logic computation, while the other resources are used for the structure reconfigurability [15]. Such a large imbalance results in a large cost in terms of area, routing delay and power consumption.

In this chapter, we exploit the ultrafine grain LCs based on the DG-SiNWFET to improve the FPGA architecture:

1. We introduce a novel architectural organization where the standard *look-up tables* (LUTs) are replaced by ultrafine grain LCs. While it seems counterintuitive to further reduce the size of the logic blocks, therefore worsening the logic/routing imbalance, we will see that the ultrafine grain elements can be organized in compact matrix arrangements of LCs, called MClusters that become competitive as compared with the typical combinational elements of the FPGAs. To prevent the reconfigurable interconnection overload within the MClusters, we interconnect the cells through a layered, fixed and incomplete interconnection pattern, i.e., that an output on a given layer is permanently connected to a subset of inputs on the next layer.
2. To support the novel architecture, we propose a complete benchmarking tool flow suited to this organization. In particular, we develop a new tool, called MPack, aiming at mapping logic functions to MClusters and clustering MClusters into logic blocks of FPGA architectures. We integrate MPack into a state-of-the-art FPGA flow, enabling a completed evaluation on area, delay, and power consumption of the novel FPGA architectures.

Architecture-level results show that an FPGA structure using 2-by-2 MClusters in replacement of the traditional LUTs gives an average area and delay reduction of 43% and 23%, respectively, compared with a conventional CMOS FPGA at 22-nm technology node. This large performance gains are directly accounted from the efficient use of ultrafine grain LCs. In addition, the proposed approach opens a promising path to correct two intrinsic limitations of the FPGA circuits: (1) an 8% reduction in the logic/routing imbalance and (2) a better control in the distribution of the routing delay.

The rest of this chapter is organized as follows. Section 12.1 surveys the necessary background on FPGA architecture, functionality-enhanced devices, and ultrafine grain LC designs. Section 12.2 presents the novel architectural scheme based on ultrafine grain reconfigurable LC, by highlighting the concept and the organization at each architectural level. Section 12.3 introduces the specific benchmarking tool MPack, with an emphasis on the MCluster packing. Section 12.4 evaluates the performance of

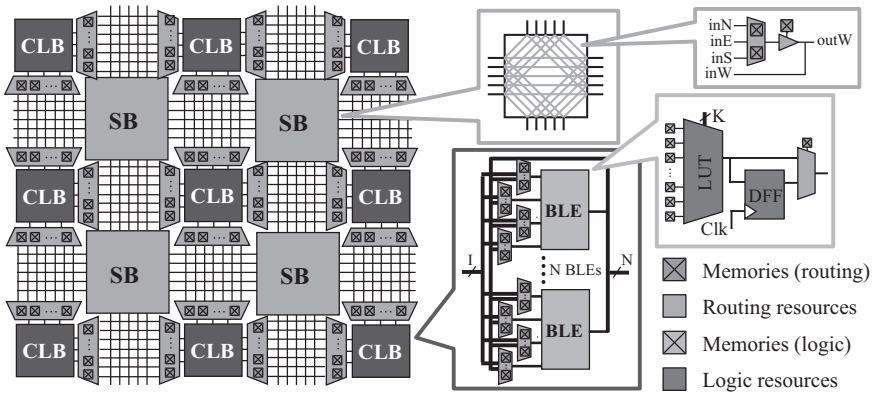


Figure 12.1 Baseline field-programmable gate arrays architecture [15]

the targeted architecture, in terms of granularity and compares them with its CMOS FPGA counterpart. Section 12.5 concludes this chapter.

12.1 Background

In this part, we first introduce the necessary background on FPGA architecture (see Section 12.1.1). Then, we provide a brief review on controllable-polarity devices (see Section 12.1.1) and associated ultrafine grain computation topology (see Section 12.1.3).

12.1.1 FPGA architecture

As shown in Figure 12.1, FPGAs are highly regular circuits typically consisting of an array of identical *configurable logic blocks* (CLBs) surrounded by reconfigurable routing resources [15]. Each CLB contains a set of N *basic logic elements* (BLEs) and rich local routing resources. A BLE is formed by a K-input LUT, a flip-flop and a two-input reconfigurable routing multiplexer to select either a combinational or a sequential output. The local routing resources, consisting of reconfigurable routing multiplexers, interconnect the I CLB inputs and the CLB outputs to the inputs of BLEs.

All design parameters N, K, and I can be set by the FPGA architect depending on the targeted system granularity. The reconfigurable routing resources outside CLBs are called global routing architecture, which is organized by large channels of width W, interleaved between the CLBs. The channels cross each other and the signals can be routed within the FPGA using *switch boxes* (SBs). An SB is a matrix at the intersection of channels, which is made of reconfigurable routing multiplexers.

As we see in Figure 12.1, routing resources appear in every hierarchy of FPGA architectures, being a dominating factor on the total area, delay and power

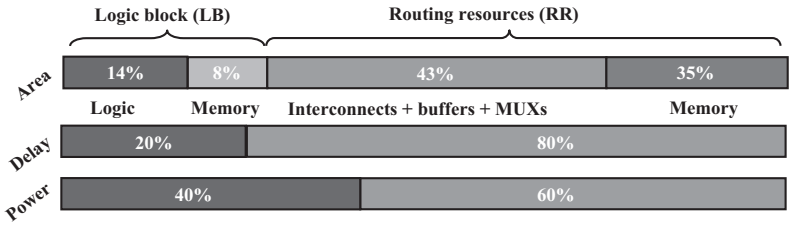


Figure 12.2 Field-programmable gate arrays area/delay/power repartition per block [16]

consumption. Figure 12.2 shows the area, delay and power breakdown of the various components of a baseline *static random access memory* (SRAM)-based FPGA. The heavy use of reconfigurable routing resources leads to the imbalance between logic and routing. Only 14% of the total area is used for actual computation elements, i.e., LUTs. In particular, roughly half of the area in both the logic blocks and the routing resources are consumed by configuration memories. In addition to occupying most of the die area, routing resources significantly contribute to FPGAs hurdles. Interconnect delays are reported to account for roughly 80% of the total path delay [16]. They also contribute to the high-power consumption of FPGAs, with more than 60% of the total dynamic power consumption.

In this chapter, we will develop novel logic block structures, by exploiting the reconfigurability at the device level, offered by novel device technologies, in the purpose of reducing the imbalance between logic and routing.

12.1.2 Transistors with controllable polarity

At advanced technology nodes (45 nm and below), the superposition of n-type and p-type carriers is observable even under normal bias conditions in several nanoscale FET devices. The phenomenon, called ambipolarity, occurs in many different materials, such as silicon [17], carbon nanotubes [18] and graphene [19]. The control of this ambipolarity enables device-level reconfigurability, allowing us to adjust the device polarity online. Transistors with a controllable polarity have been experimentally fabricated in several novel technologies, such as carbon nanotubes [12], graphene [13] and silicon nanowires (SiNWs) [10,11]. The operation of these FETs is enabled by the regulation of Schottky barriers at the source/drain (S/D) junctions thanks to an additional gate.

In this chapter, we consider here vertically stacked SiNWFETs [14], as illustrated in Figure 12.3. Featured by vertically stacked nanowires and *GAA* structures, the SiNWFETs can be regarded as a natural evolution of FinFET structures. As depicted in Figure 12.3(a), a SiNWFET has four electrodes: (1) the control gate; (2) the source node; (3) the drain node; and (4) the polarity gate (PG). Similar to conventional transistors, the control gate acts conventionally by turning *on* and *off* the device, while the source and drain nodes are connected to the channels. The key difference lies in the fourth electrode, the PG, acts on the side regions of the device, in proximity

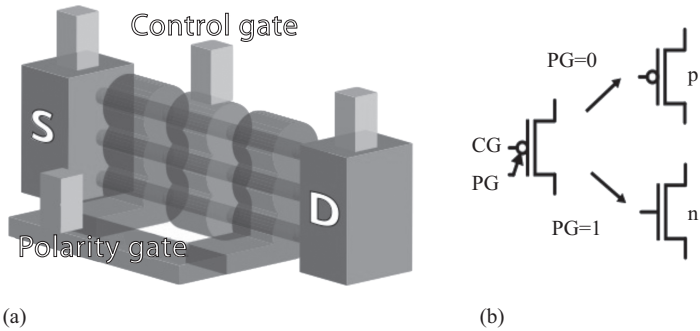


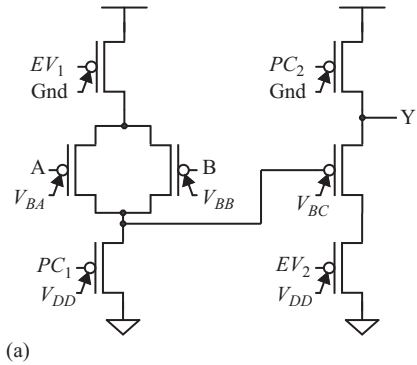
Figure 12.3 (a) 3D sketch of the SiNWFETs featuring two independent gates and (b) its associated symbol [14]

of the S/D Schottky junctions, switching the device polarity dynamically between n- and p-type. As illustrated in Figure 12.3(b), when the PG is configured to logic `0`, the SiNWFET behaves a p-type transistor, while its functionality can be switched to an n-type transistor by applying logic `1` to the PG. In addition, the control gate and the PG are fabricated with GAA structure, providing better electrostatic control over the channel and consequently superior scalability properties [14]. Compared to planar transistors, transistors with GAA structure can drive a much lower leakage current, below 4*fA* [14]. Note that the input and output voltage levels are compatible, logic gates can be cascaded as CMOS technology [14]. For a complete review on the design opportunities brought by these transistors, we refer the reader to [8].

12.1.3 Ultrafine grain reconfigurable logic gates

The in-field reconfigurability of ambipolar transistors has been exploited in [7] to build a compact reconfigurable cell. As shown in Figure 12.4(a), the reconfigurable cell can realize any of the eight Boolean logic functions Y of the two inputs A and B, which is detailed in Figure 12.4(b). The cell, consisting of only seven transistors, is implemented in the principle of the dynamic logics [20], arranged in two stages: logic function and follower/inverter. Signals PC_1, EV_1, PC_2, and EV_2 are the global precharge and evaluation signals of the two stages, respectively. The reconfiguration of the cell depends on the signals applied to the PGs, V_{BA}, V_{BB} and V_{BC}. Each of these signals configures the related transistors to either n- or p-type, thereby customizing the gate internal circuit. For a detailed description of the reconfigurable cells, we refer the reader to [7].

In this chapter, for practical reasons, we consider only binary configuration signals (either V_{DD} or *GND*), limiting the gate reconfigurability to eight functions. Previous work [7] considered ternary configuration signals, increasing the logic capability of the cell to 14 functions. We qualify the circuit as operating at ultrafine grain. The terminology of ultrafine grain is twofold. First, it covers the size of the cell, which is highly compact, as opposed to its CMOS equivalent. Second, it also



V_{BA}	V_{BB}	V_{BC}	Y
0	0	0	$\overline{A.B}$
0	0	1	A.B
0	1	0	$\overline{A}$+B
0	1	1	A.$\overline{B}$
1	0	0	A+$\overline{B}$
1	0	1	$\overline{A}$.B
1	1	0	A+B
1	1	1	$\overline{A+B}$

(b)

Figure 12.4 Ultrafine grain reconfigurable cell schematic [7] (a) and associated configurations (b)

describes the granularity of the covered logic functions, as compared with a traditional four-input LUT.

12.2 Leveraging the ultrafine granularity at the architecture level

In Section 12.1.3, we showed the circuit-level implementation of the new class of ambipolar devices. In this section, we will study, at the architectural level, the impact of the ultrafine gain cells onto the FPGA architecture.

In the vision of our FPGA architecture, the traditional LUTs are replaced by an ultrafine grain LC. In a conventional FPGA architecture, such a modification will require each cell to be directly connected to a full connectivity unit. As the typical cell size is comparable with a 1-b SB [7], this projection would cause a large overhead in terms of connections and also would worsen the already significant imbalance between routing and logic resources at the FPGA level. To correct the granularity issues, we propose to pack the cells into an intermediate structure, compatible in terms of size with the traditional levels of FPGAs, and called MClusters for Matrix Clusters.

In the rest of this section, we will detail the organization of MClusters from three aspects: multilayer structure (see Section 12.2.1), inter-MCluster interconnection (see Section 12.2.2) and integration to FPGA architectures (see Section 12.2.3).

12.2.1 Multilayer organization

To increase the logic coverage of the MClusters, we propose a 2-D matrix assembly of ultrafine grain logic. The LCs are arranged in a layered structure, with connections only existing between adjacent layers, as depicted in Figure 12.5. As such, long

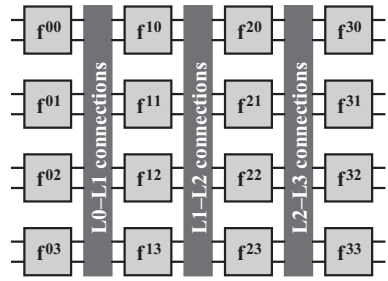


Figure 12.5 MClusters_4_4 arrangement for reconfigurable architectures

connections inside MCluster can be avoided and local connectivity can be maximized. Note that each layer is simply an array of LCs without pipeline or signal restoration.

In the rest of this chapter, MClusters will be defined by the d (depth) and w (width) parameters and named as *MClusters_d_w*. Note that, in Figure 12.5, each LC has two inputs, one is the output Y of the LC [Figure 12.4(a)], while the other is a duplication output Y. As such, we keep the same number of the input and output terminals per layers. The matrix patterns exhibit a fundamental structural regularity, being easily transposable to regular fabrics [8]. Note that the architectural concept is not limited to the presented architecture. Indeed, it is possible to consider diamond-shaped or triangular matrices and to have interconnects crossing different layers. However, to keep the focus of this chapter, only squared matrices, i.e., $d \times w$ LCs and $d = w$, and layer-to-layer connections are considered.

12.2.2 Intramatrix interconnecting

In relaxing the imbalance between logic and routing resources, the intramatrix interconnect will avoid introducing additional reconfigurable routing multiplexers. Therefore, any interconnecting topology giving a full interconnectivity is prohibited, due to the associated wiring complexity and intensive use of routing multiplexers. Similar to computer networks, our approach adapts incomplete interconnection sets to the matrix architecture. We employ *multistage interconnection networks* (MINs) to interconnect matrix layers. In computer engineering, MINs are used to interconnect layers of *SBs* to route information packets only. This concept has been applied in interconnect strategies for VLSI [21] and also FPGAs in order to reduce the wiring complexity between the logic blocks [22]. Note that the main difference lies in that *SBs* in the network context are replaced by LCs in our MCluster, thus introducing computing within the network itself. Furthermore, the use of very local MIN-style interconnect has a drastic impact on the size and the wire length. Among many MIN topologies and combinations [23,24], we report four typical permutations in Figure 12.6:

1. Banyan [Figure 12.6(a)],
2. Baseline [Figure 12.6(b)],

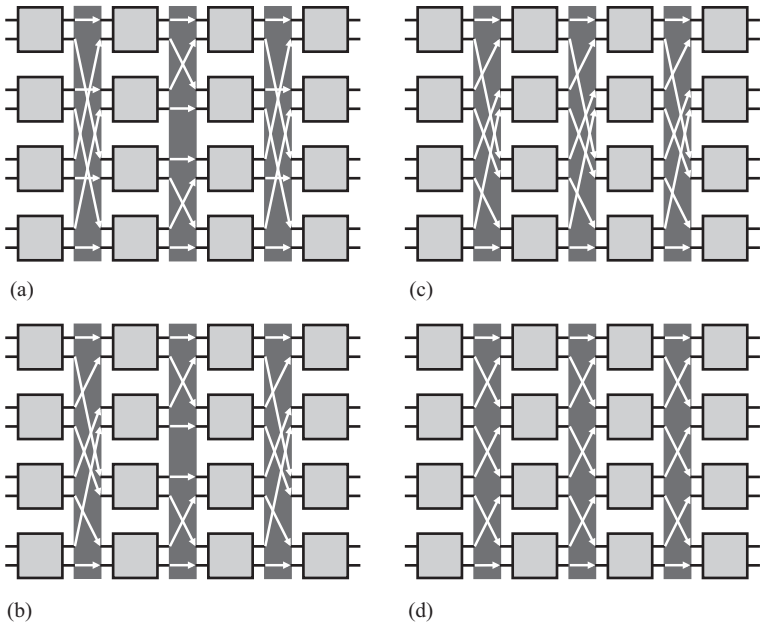


Figure 12.6 Matrix of 16 reconfigurable gates with fixed interconnect topology—(a) banyan, (b) baseline, (c) flip and (d) modified omega

3. Flip [Figure 12.6(c)] and
4. Modified omega [Figure 12.6(d)], where the modifications to standard omega maximize the shuffling between the layers.

Since the interconnect topology is fixed and static, the choice of topology is made by considering the application context. In the context of matrix-based logic, the performance of the different topologies has been investigated in [25]. It has been shown that a modified omega topology has the best performance in terms of mapping efficiency. Therefore, in the rest of this chapter, we consider only modified omega interconnect topology.

12.2.3 Integration into FPGA architecture

As MClusters contain rich reconfigurable LCs, they can potentially perform a large set of combinational logic functions, being a competitive candidates for substituting the original LUTs. Figure 12.7 shows the organization of the logic blocks at the CLB level. The CLB includes N MCluster-based BLEs, each of which consists of an *MClusters_d_w*, whose outputs can be individually latched on demand. Similar to conventional CLB organization, all the BLE outputs, i.e., $w \times N$ signals, are connected to the local routing resources. Each BLE input can be reached by every BLE outputs and every CLB inputs, thus achieving a full connectivity pattern.

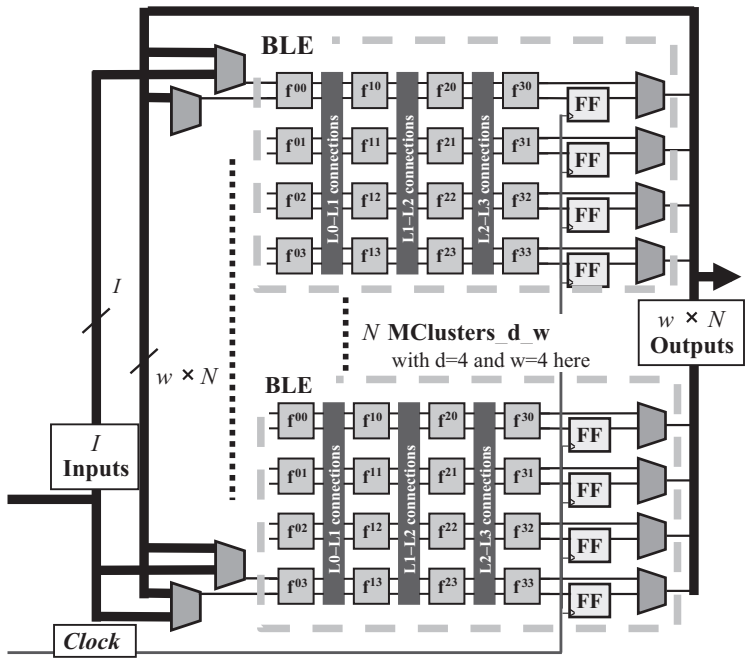


Figure 12.7 MCluster-based CLB proposal

At the top architectural level, we consider a standard FPGA organization shown in Figure 12.1, where the CLBs are interconnected by a global routing architecture.

12.3 MCluster CAD flow

To analyze the performance gain of the proposed FPGAs from an application perspective and to compare it to conventional counterparts, the benchmarking of standard circuits has to be carried out. The conventional FPGA computer-aided design (CAD) flow is developed for LUT-based FPGA architecture. To unlock the potential of MCluster, in this section, we develop novel CAD tool MPack supporting the MCluster-based FPGA architectures. And MPack is integrated into a complete benchmarking tool flow based on the well-known *verilog-to-routing* (VTR) flow [26].

In the rest of this section, we first give an overview of the developed benchmarking flow strategy in Section 12.3.1. Then, we introduce the principles of a novel CAD tool, called MPack, which is specifically designed to handle the MCluster packing step with fixed interconnect topologies (see Section 12.3.2). We detailed the mapping algorithms and clustering algorithms of MPack in Sections 12.3.3 and 12.3.4, respectively.

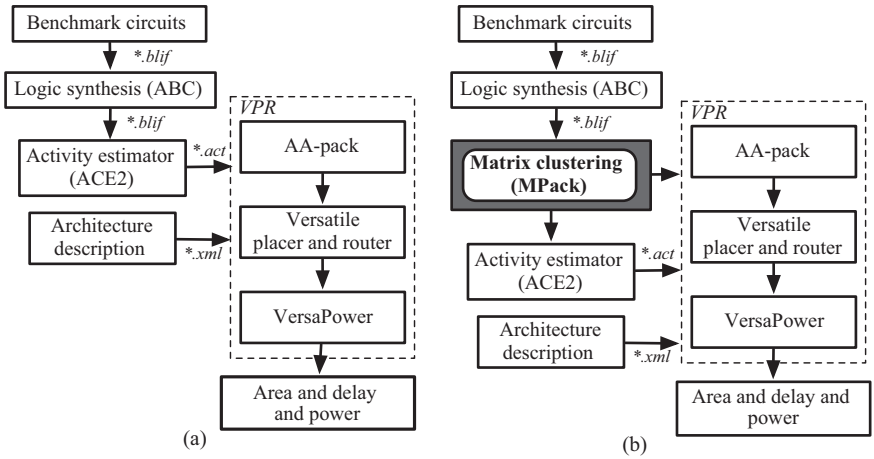


Figure 12.8 Comparison between (a) VTR flow for conventional FPGA architectures and (b) proposed MCluster-compatible benchmarking flow

12.3.1 General overview of the flow

For conventional FPGAs, architecture-level evaluations are conducted with a traditional FPGA tool flow, i.e., VTR flow [26], as illustrated in Figure 12.8(a). A set of logic circuits, called benchmarks, are first optimized at logic level using the ABC synthesis tool [27]. Subsequently, the logic packing of the synthesized circuit into CLBs is performed, using AA-PACK [28]. Finally, the placement and routing are carried out using *versatile place and route* (VPR) [15]. Power results are estimated by VersaPower [29] with the net activities computed by ACE2 [30].

However, the MCluster architecture introduces new logic circuit different from LUTs, which is not handled by traditional FPGA tools. In particular, the layered structure with various interconnect topology is not supported by traditional packing tools, such as AA-PACK [28]. Therefore, a specific tool, called MPack, is developed, and a novel design flow is proposed to leverage the potential of MPack. As shown in Figure 12.8(b), MPack maps the logic functions into LCs and MClustering, by considering the specific interconnect topologies as well as the associated buffering generation required by the layered structure. Details of the tool are further described in Sections 12.3.2–12.3.4.

12.3.2 MPack: the matrix packer

The matrix packer, MPack, is developed to pack netlists of LCs into MClusters, whose internal organization in shown in Figure 12.9. MPack consists of two principal algorithms: the mapper and the clusterer. The mapper maps simple logic functions on a unique MCluster. The clusterer splits the input logic network into subgraphs that can be subsequently mapped on individual MClusters thanks to the mapper. For a seamless

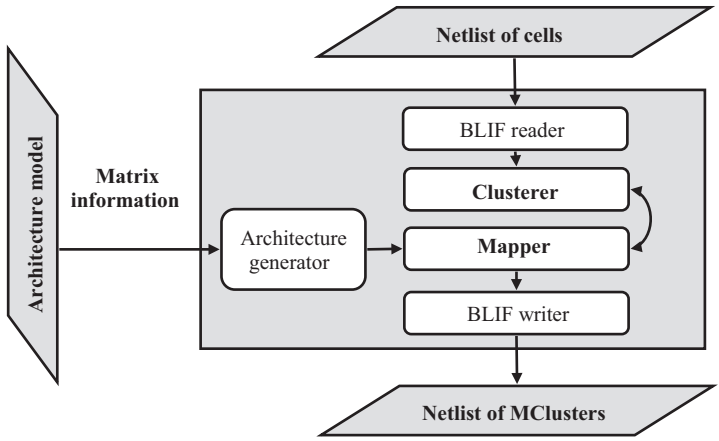


Figure 12.9 MPack model flow

integration with VTR flow, the input and output logic network are represented in the *Berkeley logic interchange format* (BLIF) file format.

MPack supports customized MCluster structures. A built-in Architecture Generator can automatically generate the MCluster template, with user-specified depth, width and interconnecting topology scheme. Figure 12.10(a) shows an example of a MCluster featured by $d = w = 3$ and modified omega interconnecting topology. In Figure 12.10(b), the nodes f^{ij} are uniquely addressed by the indices i and j, where i indicates the row while j denotes the column. MPack employs the adjacency matrix of the graph to represent the connectivity between nodes (LCs), where a 1 at the position (i,j) means that the node i is connected to the node j. The local adjacency matrices X_{nm} between the LC stages at depth n to m are defined as the individual cross-connectivity matrices. For instance, Figure 12.10(b) shows two examples: X_{01} and X_{12}, corresponding to the MCluster shown in Figure 12.10(a). In the example, the first row of the matrix X_{01} shows the connectivity of the cell f_{10}, while f_{10} is connected to f_{00} and f_{01}.

12.3.3 Matrix mapping algorithm

The mapping algorithm of MPack aims at fitting a netlist of LCs onto an MClusters architecture. The module is split into two parts: (1) an architecture optimization is performed to adapt the netlist to the physical architectural scheme (Section 12.3.3.1) and (2) the logic function netlist is mapped to the architecture netlist (Section 12.3.3.2).

12.3.3.1 Architectural optimization

The architectural optimization is a preprocessing stage, which aims at adapting the logic netlist to the physical architectural scheme. The layered structure forces the logic to be pipelined. The input netlist requires a processing where its connections is adapted to conform to the physical topology. For instance, a typical post-synthesis netlist

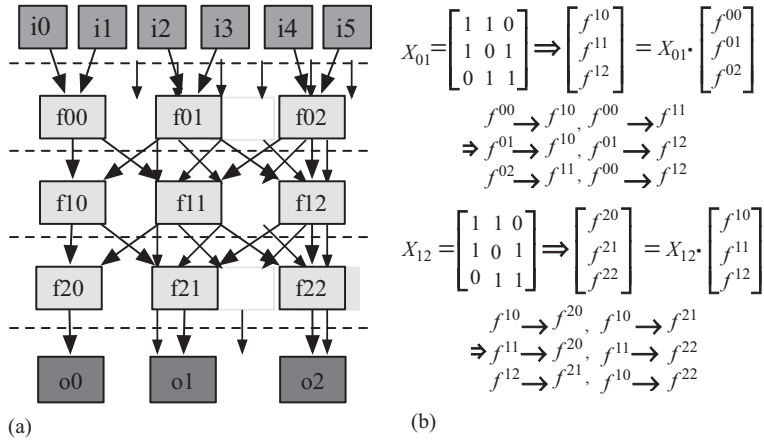


Figure 12.10 *(a) MClusters_3_3 with a modified omega topology (logic cells are labeled by "f", virtual input nodes are labeled by "i", virtual output nodes are labeled by "o") and (b) associated cross-connectivity matrices*

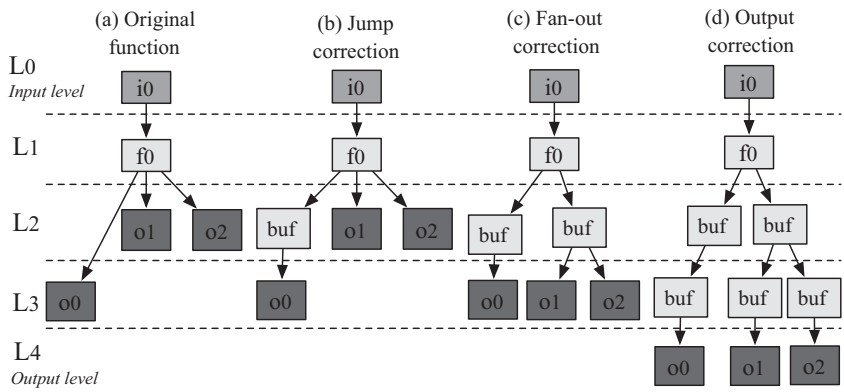


Figure 12.11 *Architectural optimization of the function graph. (a) Original function before formatting. (b) Jump correction. (c) Fan-in–fan-out correction. (d) Output layer correction by buffer insertion*

allows any logic gate output to be connected to several gate inputs, while MClusters have a fixed fan-in and fan-out. Therefore, to be compatible with MClusters, buffering nodes are added to the netlist, in order to limit the fan-out. Figure 12.11 provides an optimization example, where we see that the netlist is processed in three steps:

1. **Jump correction:** Since the layered topology, a connection prohibits jumps at least one logic layer, a connection between two layers must pass through an LC.

Consequently, instead of a jump, it is necessary to create a path by the addition of a buffer cell, in order to handle such a situation, as shown in Figure 12.11(a).

2. **Multiple output correction:** In the MClusters organization, the LCs are connected to a limited number of other cells due to a fan-in/fan-out limitation. Larger fan-in/fan-out requires duplicating the signals using buffers. Figure 12.11(a) shows an example case, where node f_0 drives three different nodes. Since each LC has only two outputs, a buffer is added to satisfy the constraint [Figure 12.11(b)].
3. **Output layer correction:** In this final optimization step, all the outputs of the logic functions are placed on the physical output layer. In Figure 12.11(b), we considered a logic function where the outputs are on layer $L3$. To meet the constraints of the MCluster architecture (here a MCluster with a depth of three and a width of three), buffers are added to place the outputs on the layer $L4$ [Figure 12.11(c)].

12.3.3.2 Mapping algorithm

After the architecture optimization steps, the logic function netlist is mapped to the MCluster structure, using the recursive algorithm reported in Algorithm 1. The mapping algorithm consists of two steps:

1. In the first step, the adapted netlist is analyzed in depth, where child branches are identified and recursively explored for each node in the MCluster structure. During this depth exploration, any edge orientation in the graph is not considered. As such, the connections between the nodes are identified and the mapping sequence can be initialized.
2. In the second step, logic nodes are assigned to ultrafine grain LCs, with respects to the physical interconnections between layers. The connectivity matrices of each layer are compared with the relevant interlayer architectural connectivity allowing (or not) the assignment of functions to cells.

Branching, i.e., the exploration of the immediately preceding alternative, is used when the arbitrary choice leads to a dead-end. The process is iterated until all functional nodes are assigned to physical cells.

For instance, let us consider a matrix with a depth of three and a width of three (*MClusters*_3_3) using a modified omega interconnect topology [Figure 12.10(a)]. Figure 12.12(a) shows a simple function to be mapped on a MCluster. With architecture optimization methodology explained in Section 12.3.3.1, the original function graph is modified to maximize the matching with the targeted MCluster. The architecture optimization engine will correct the position of the inputs/outputs and interlayer jumps, resulting in the graph as shown in Figure 12.12(b). The graph exploration is then triggered and the following sequence is obtained:

$$\begin{aligned} \text{pi0} \quad & -\text{n1} - \text{pi1} - \text{buf} - \text{n4} - \text{n3} - \text{n2} - \text{pi2} - \text{pi3} \\ & -\text{buf} - \text{n5} - \text{po1} - \text{po0} \end{aligned} \tag{12.1}$$

where buf represents synchronization nodes. Subsequently, the graph assignment is performed.

Take the example shown in Figure 12.12(b), the first point n_1 is assigned to the ultrafine grain LC f^{00}. According to the path defined in the previous step, a buffer is

Algorithm 1: MPack mapping algorithm (pseudo-code)

```
Function Mapping_procedure(Current node):
    if (all children/parents of current node placed) then
        for (empty cell connected to children or parents) do
            Store the potential positions;
        end
    else
        Store all the positions on the current level
    end
    if (No positions have been found) then
        MAPPING FAILURE;
    end
    for (Each potential position) do
        Place the current node at the position;
        Mapping_procedure(Next node);
        if (Return is MAPPING FAILURE) then
            Unplace the current node from the position;
        end
        if (Return is MAPPING SUCCESS AND No more nodes) then
            MAPPING SUCCESS;
        end
    end
    if (No potential positions have been correct) then
        MAPPING FAILURE;
    end
end
```

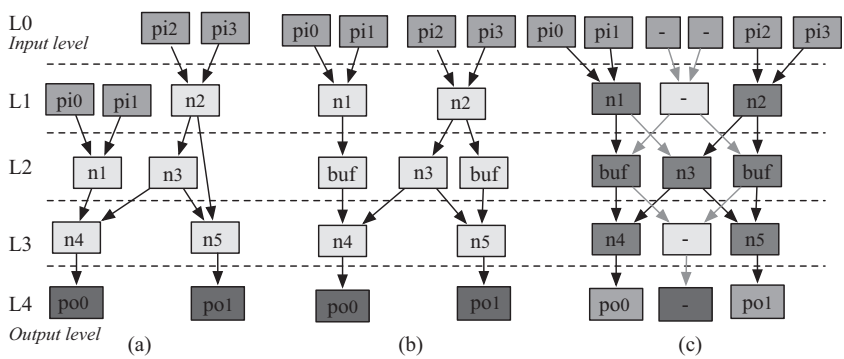


Figure 12.12 Mapping example. (a) Original function graph to map onto an MClusters_3_3 with a modified omega interconnect topology. (b) Function graph after correction step. (c) Mapped function

the next node to assign to a ultrafine grain LC in the matrix. Since f^{00} is physically connected to f^{10} and f^{12}, the LC with lower index (here f^{10}) is arbitrarily chosen for the buffer assignment, and the other possibility is memorized. In our example, the final programmed matrix is shown in Figure 12.12(c), where the nodes of the logic function graph and the nodes added for synchronization purposes are correctly placed on the cell matrix.

12.3.4 Clustering algorithm

The clustering algorithm is the hardcore of the cluster shown in Figure 12.9. It aims at splitting the initial logic network into subgraphs that will subsequently be mapped onto MClusters. Such functionality has been widely used in many CAD tools. The MPack tool borrows the clustering algorithms from VPACK [15]. Each MCluster is constructed sequentially, and the ultrafine grain LCs are greedily packed into MClusters. The pseudo-code for the algorithm is shown in Algorithm 2.

It begins by choosing a seed LC for the current MCluster. The clustering algorithm selects the unclustered LC with the most used inputs as the seed, which has been proved as an efficient way [15]. This approach prioritizes early placement of cells using the largest number of cluster inputs, which are a scarce resource. Next, the algorithm selects the LC with the highest attraction to the current MCluster and checks if it could legally be added to the current cluster. The mapping algorithm described in

Algorithm 2: Clustering algorithm (pseudo-code)

```
Remain_Cells: set of unclustered logic cells.
C: set of cells packed in the current MCluster.
MClusters: set of MClusters.
while (Remain_Cells != NULL) do
    /* More Cells to cluster                                    */
    C = Seed(Remain_Cells);
    /* Most Used Inputs and legal Cell                          */
    while (|C| < (MCluster_w.MCluster_d)) do
        /* Cluster is not full                                  */
        Best_Cell = MaxAttractionLegalCell(C,Remain_Cells);
        if (Best_Cell == NULL) then
            BREAK;
        end
        Remain_Cells = Remain_Cells - Best_Cell;
        C = C ⋃ Best_Cell;
    end
    MClusters = MClusters ⋃ C;
end
```

Section 12.3.3.2 evaluates the legality of adding new LCs. If the cluster is mappable, then the LC is definitively added to the current cluster and the mapping result is recorded. Similar to [15], we define the attraction between an LC and the current MCluster (MC) as a function related to the number of the inputs and outputs they have in common:

$$Attraction(LC) = |Nets(LC) \cap Nets(MC)|. \tag{12.2}$$

This greedy procedure continues until either

1. the MCluster is full or
2. adding any additional cell would make the current cluster illegal.

If the current cluster is full, a new seed is selected and the packing starts for another MCluster.

12.4 Experimental results

In this section, we perform architecture evaluation for the novel FPGA architecture based on MClusters introduced in Section 12.2 with the novel benchmarking flow presented in Section 12.3. We first introduce our methodology of comparing the performance of FPGA architecture in Section 12.4.1. Then, we perform an architectural exploration to identify the best MClusters sizing in Section 12.4.2. With best MCluster sizing, we compare the novel FPGA architecture to its traditional CMOS-based counterpart in Section 12.4.3.

12.4.1 Methodology

In this chapter, our major objective is to study the impact on replacing standard CMOS LUTs by SiNWFET-based MClusters in FPGA architecture. We evaluate the performances of a set of logic circuits taken from the *Microelectronics Center of North Carolina* (MCNC) [31] and ISCAS'89 [32] mapped on simple homogeneous FPGA architectures. Our baseline FPGA architecture is a fully optimized homogeneous FPGA architecture, as shown in Figure 12.1. The CMOS reference architecture is composed of four-input non-fractionable LUTs with a CLB organization of ten BLEs and 22 external inputs ($K = 4, N = 10, I = 22$). The CLB architecture is optimal for homogeneous CMOS-based FPGAs [33]. We then proceed to a one-to-one replacement of the LUTs by MClusters according to Figure 12.7, i.e., keeping $N = 10$. The size of MCluster under evaluation is swept from *MClusters_1_1* to *MClusters_4_4*. To ensure a proper routability of the logic cluster, the number of BLEs inputs and outputs are determined by considering the size of the MCluster. The number of CLBs external inputs is set to

$$I = \frac{(N + 1) \times nb\_MCluster\_inputs}{2} \tag{12.3}$$

This relation typically ensures that 98% BLEs are utilized on average [33]. For example, an *MClusters_2_2*-based CLB has 22 external inputs ($I = 22$), while an

MClusters_4_4 has 44 external inputs ($I = 44$). For these two examples, the local routing circuits consist of a set of 42- and 84-b multiplexers, respectively. SRAM circuits are used to store the configuration information for both LUTs and MClusters.

For CMOS LUT-based FPGA architecture, we evaluate by using the benchmarking flow described shown in Figure 12.8(a). For SiNWFET-based MCluster FPGA architecture, we consider the benchmarking flow described shown in Figure 12.8(b). The physical parameters of the two FPGA architectures are extracted for a 22-nm technological node [1]. MClusters and LUTs area evaluation estimates the size of the blocks as a function of the elementary transistors area [1,14] and the transistor sizing. For instance, a 2-by-2 MCluster costs an area of 2.22 μm^2, while a four-input LUT costs 5.45 μm^2. To extract the basic average delay and power consumption of LUTs and MClusters, electrical analysis are conducted by using HSPICE simulations [34]. For CMOS circuits, the SPICE netlists are built with FinFET high performance at 22-nm model took from [35]. For SiNWFET-based circuit, the SPICE netlists are constructed with a simple table-based model for DGSiNWFETs took from [14]).

The architectural evaluation considers four metrics: (1) the area, (2) the critical path delay, (3) the dynamic power consumption and (4) the leakage power. These metrics are computed during the place and route iterations of the flow. The area corresponds to the sum of the logic area, i.e., the area of used CLBs, and the routing area, i.e., the area of the used routing resources. The critical path delay corresponds to the most constrained delay through the implemented structures. Finally, the power consumptions include both the contribution of logic blocks and the contribution of routing structures. All the metrics are normalized with respect to the most constrained CMOS design.

12.4.2 Impact of the granularity

We study the impact of MCluster sizes on the area, the delay, and the power consumptions averaged over the considered benchmark circuits. In this part, we sweep the dimensions of squared-shaped MClusters from *MClusters_1_1* to *MClusters_4_4*. Note that averaged over the different benchmarks and different cluster sizes, the utilization rate of the MClusters, i.e., the ratio between the employed LCs and the total number of LCs, stays at 90%, thereby indicating a good level of performances of MPack.

Figure 12.13(a) presents the area estimation of an FPGA based on MClusters. Small MCluster sizes, i.e., *MClusters_1_1* and *MClusters_2_2*, lead to best area results. Note that, the area increases drastically when a higher granularity (*MClusters_3_3* and *MClusters_4_4*) is applied. For large MClusters, the area overhead comes from the incomplete interconnectivity in their internal routing. In these cases, a large number of LCs are used only for buffering purposes, leading to a low mapping efficiency.

Figure 12.13(b) and (d) presents the critical path delay, the dynamic power consumption, and the leakage power consumption, respectively, under different MCluster sizes. Similar to the area results, the MClusters with small granularities, i.e., *MClusters_1_1* and *MClusters_2_2*, guarantee best delay and power consumption. Note that,

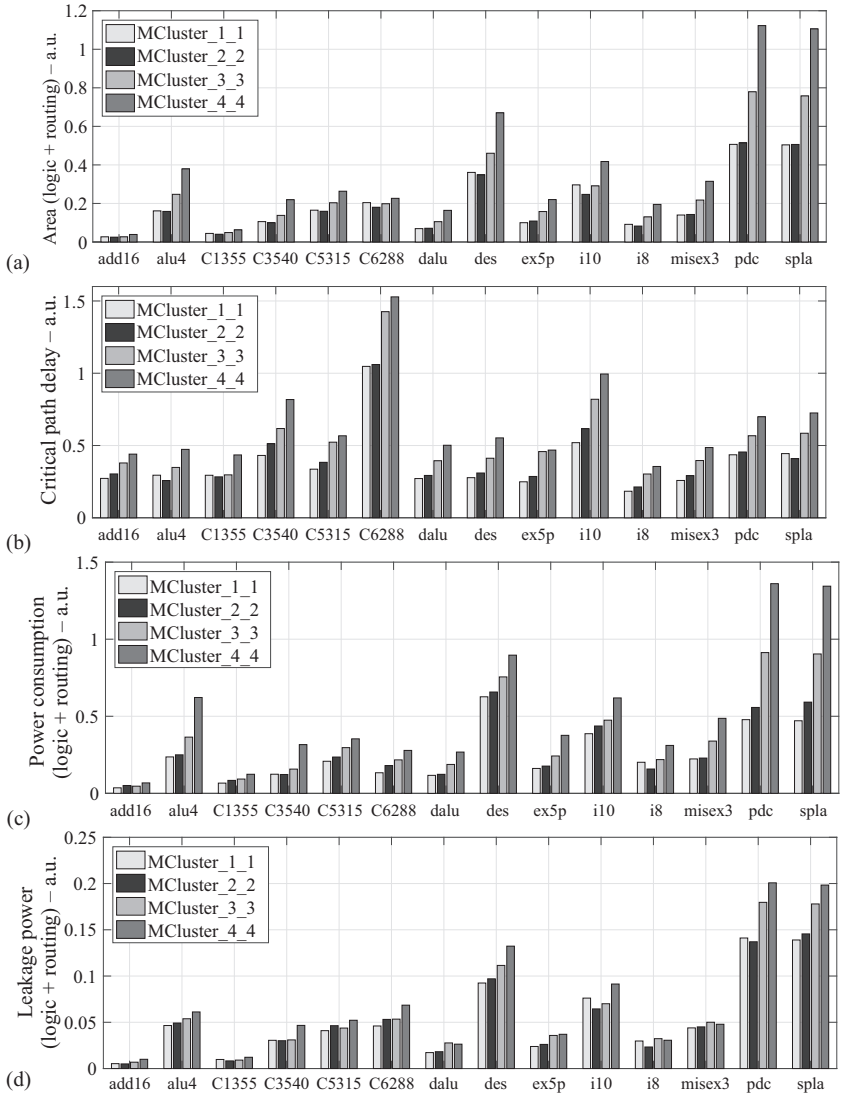


Figure 12.13 Impact of various granularities on MCluster-based FPGAs (a) global area, (b) critical path delay, (c) dynamic power consumption, (d) leakage power consumption. MClusters are sized as squares ranging from 1 to 4 reconfigurable cells

the delay and the power consumption are almost linear to the size of the MClusters. As large MClusters contain deeper logic pipeline, many LCs are used as buffers in order to fit the interlayer interconnecting topology. Therefore, the mapping inefficiency directly reduces the performances and increases the power consumption.

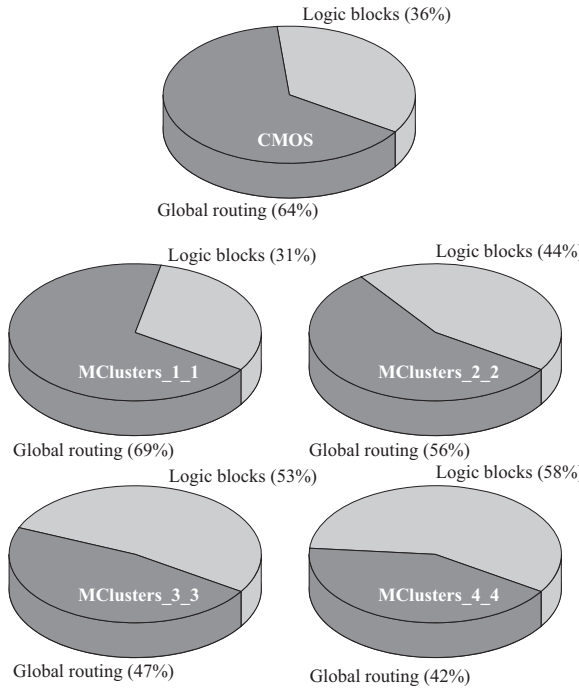


Figure 12.14 Area breakdown of an interconnection intensive benchmark (i10) for 2-by-2 MCluster-based and CMOS-based FPGAs

In terms of performance levels, *MClusters_1_1* overperforms slightly *MClusters_2_2* with 5%, 4%, 12%, and 1% gains in the area, the delay, the dynamic power and the leakage power, respectively. However, when determining the best MCluster size, we also consider more factors, i.e., the impact of the different MCluster granularities on the area breakdown between the logic blocks and global routing. Figure 12.14 shows the area breakdown for an interconnection intensive benchmark (i10). As discussed in Section 12.1.1, a fundamental drawback of the traditional FPGA architecture is its strong imbalance between the routing resources and the LCs. As shown in Figure 12.14, the use of *MClusters_1_1* worsens the natural imbalance. This violates our initial hypothesis as explained in Section 12.1, where the direct transposition of an FPGA architecture to ultrafine grain would help alleviate interconnection overhead. On the other side, large clusters can significant increase the logic share, but they will also strongly limit the performances of the architecture, as shown Figure 12.13.

Overall, *MClusters_2_2* provides a best trade-off between performance (Figure 12.13) and logic/routing balance (Figure 12.14). In the rest of this chapter, we consider *MClusters_2_2* as the ultrafine grain FPGA architecture when comparing to CMOS FPGAs.

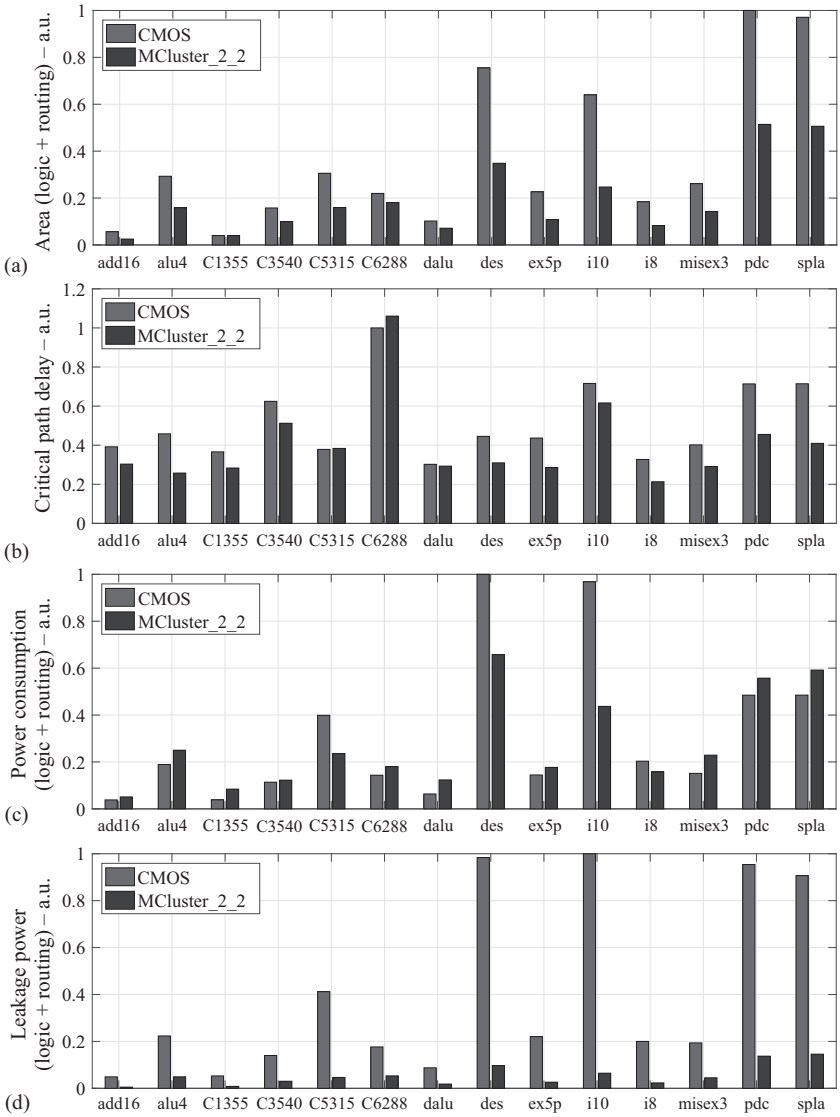


Figure 12.15 Performance comparisons between CMOS-based and MClusters_2_2-based FPGAs (a) global area, (b) critical path delay, (c) dynamic power consumption and (d) leakage power consumption

12.4.3 Performance comparison with CMOS

Figure 12.15 compares the area, delay and power consumption of *MClusters_2_2*-based FPGA and its CMOS counterpart.

In terms of area, Figure 12.15(a) shows that *MClusters_2_2*-based FPGA brings a reduction of up to 61%, with an average of 43%. The area benefits can be accounted: (1) to the performance of a logic-gate-based computation (as compared to the LUT approach) and (2) to the low area impact of ultrafine grain LCs, compared with the rather larger area required by a CMOS LUT. As mentioned in Section 12.4.1, a 2-by-2 MCluster is 2× smaller than a four-input LUT. Furthermore, in terms of implementing multi-output functions, *MClusters_2_2* is potentially more capable than its equivalent LUT. A four-input LUT can realize any function with four inputs and a single output, but two LUTs are required for a two-output function. While *MClusters_2_2* can only cover a subset of the single-output functions reachable by a LUT, they can produce four-input and two-output functions, beyond the capability of a LUT. Hence, MClusters can be regarded as outputting 2× more results for roughly 2× less area. Correlated to the efficiency of MPack, this demonstrates a clear advantage of our proposed FPGA architecture as compared with the CMOS approach.

Similar conclusion can be drawn for the delay [Figure 12.15(b)], the dynamic power consumption [Figure 12.15(c)] and the leakage power consumption [Figure 12.15(d)]. MCluster-based FPGAs can improve the critical path delay by up to 44% with an average value of 23%, with regard to the standard FPGA counterpart. The performance improvement comes from the higher performance of the MCluster structure as compared with LUTs. A dynamic power consumption reduction of up to 54% is observed, though, on average, we observe an increase of 19% in the dissipated power. Nevertheless, we emphasis that large dynamic power consumptions reductions are achieved on the larger benchmarks, indicating a promising trend toward the power reduction of larger circuits. Finally, leakage power can be reduced by up to 94% with an average reduction of 83%. This large reduction is due to the ultralow power capabilities of the controllable-polarity transistors, coming from the GAA structure, coupled to the resources reduction coming from the MCluster approach. Note that even if the granularity chosen in the previous section is not optimal for delay and power numbers, the improvements as compared with traditional FPGAs are significant and can be further pushed by choosing an appropriate granularity.

In addition to the performance improvement, we extend the study to the critical net routing delay. This metric strongly relates to the FPGA architecture efficiency as well. Figure 12.16 analyzes the distribution of the critical routing delay. In addition to improving the average routing delay by 43%, the use of MClusters allows us to reduce the standard deviation of the delay distribution by 45%. We can remark that while the CMOS distribution is quite large, the use of a *MClusters_2_2* implies a lowering of the extremes and tends to globally improve the performance of the mapped circuits. This behavior can be explained through the global impact of ultrafine granularity on the benchmarking tool flow. The ultrafine granularity induces a predominance of local inter-CLB interconnect instead of long wire connections, thus leading to the reduction of long critical paths. Such results are interesting from an architectural perspective as it tends to homogenize the results of mapped circuits and makes them less dependent to the route phase.

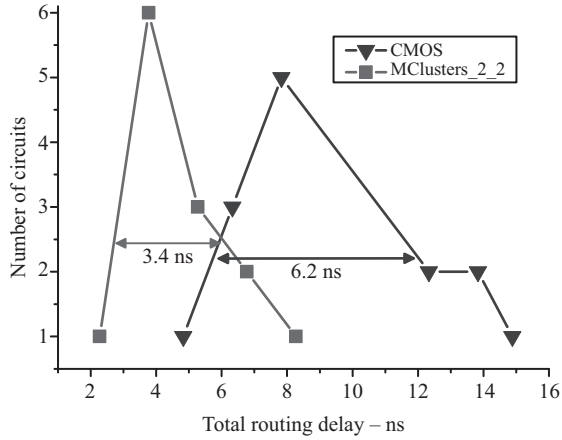


Figure 12.16 Critical net routing delay repartition for CMOS-based and 2-by-2 MCluster-based FPGAs

12.5 Conclusion

This chapter presents a new architectural scheme for FPGAs, which is based on the use of controllable-polarity transistors, i.e., SiNWFETs. Thanks to the device-level reconfigurability, ultrafine grain reconfigurable logic gates can be exploited at both circuit-level and architecture-level. We propose to replace LUTs in FPGAs by MClusters, where the ultrafine grain reconfigurable logic gates are organized in matrices and interconnected through a fixed interconnection pattern. To evaluate this approach objectively, we developed a novel tool MPack and its associated complete benchmarking flow, which is capable of packing logic circuits onto the considered architecture. We evaluated the potential of the MCluster-based architecture and compared it to a well-optimized homogeneous CMOS FPGA. We showed that 2-by-2 MClusters give an average improvement of 43% and 23% area and delay, respectively, with respect to CMOS LUT-based FPGAs at 22-nm technology node. Finally, we observed that the proposed approach opens a promising path to correct two intrinsic limitations of the FPGA circuits, with a reduction in the logic/routing imbalance, and a better control in the distribution of the routing delay.

References

[1] Auth C, Allen C, Blattner A, *et al.* A 22 nm High Performance and Low-Power CMOS Technology Featuring Fully-Depleted Tri-Gate Transistors, Self-Aligned Contacts and High Density MIM Capacitors. In: 2012 Symposium on VLSI Technology (VLSIT); 2012. p. 131–132.

[2] Intel's 10 nm Technology: Delivering the Highest Logic Transistor Density in the Industry Through the Use of Hyper Scaling. 2017. Available from:

https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/09/10-nm-icf-fact-sheet.pdf.

[3] Natarajan S, Agostinelli M, Akbar S, *et al.* A 14 nm Logic Technology Featuring 2nd-Generation FinFET, Air-Gapped Interconnects, Self-Aligned Double Patterning and a 0.0588 μm^2 SRAM Cell Size. In: 2014 IEEE International Electron Devices Meeting; 2014. p. 3.7.1–3.7.3.

[4] Wong HSP, Mitra S, Akinwande D, *et al.* Carbon Nanotube Electronics – Materials, Devices, Circuits, Design, Modeling, and Performance Projection. In: 2011 International Electron Devices Meeting; 2011. p. 23.1.1–23.1.4.

[5] Suk SD, Yeo KH, Cho KH, *et al.* High-Performance Twin Silicon Nanowire MOSFET (TSNWFET) on Bulk Si Wafer. IEEE Transactions on Nanotechnology. 2008;7(2):181–184.

[6] Bangsaruntip S, Cohen GM, Majumdar A, *et al.* High Performance and Highly Uniform Gate-All-Around Silicon Nanowire MOSFETs with Wire Size Dependent Scaling. In: 2009 IEEE International Electron Devices Meeting (IEDM); 2009. p. 1–4.

[7] O'Connor I, Liu J, Gaffiot F, *et al.* CNTFET Modeling and Reconfigurable Logic-Circuit Design. IEEE Transactions on Circuits and Systems I: Regular Papers. 2007;54(11):2365–2379.

[8] Gaillardon PE, Amarù LG, Bobba S, *et al.* Nanowire Systems: Technology and Design. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences. 2014;372(2012). Available from: http://rsta.royalsocietypublishing.org/content/372/2012/20130102.

[9] Jamaa MHB, Gaillardon PE, Frégonèse S, *et al.* FPGA Design with Double-Gate Carbon Nanotube Transistors. ECS Transactions. 2011;34(1):1005–1010.

[10] Appenzeller J, Knoch J, Tutuc E, *et al.* Dual-Gate Silicon Nanowire Transistors with Nickel Silicide Contacts. In: 2006 International Electron Devices Meeting; 2006. p. 1–4.

[11] Heinzig A, Slesazeck S, Kreupl F, *et al.* Reconfigurable Silicon Nanowire Transistors. Nano Letters. 2011;12(1):119–124.

[12] Lin YM, Appenzeller J, Knoch J, Avouris P. High-Performance Carbon Nanotube Field-Effect Transistor with Tunable Polarities. IEEE Transactions on Nanotechnology. 2005;4(5):481–489.

[13] Harada N, Yagi K, Sato S, Yokoyama N. A Polarity-Controllable Graphene Inverter. Applied Physics Letters. 2010;96(1):012102.

[14] Marchi MD, Sacchetto D, Frache S, *et al.* Polarity Control in Double-Gate, Gate-All-Around Vertically Stacked Silicon Nanowire FETs. In: 2012 International Electron Devices Meeting; 2012. p. 8.4.1–8.4.4.

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

[16] Lin M, Gamal AE, Lu YC, *et al.* Performance Benefits of Monolithically Stacked 3-D FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2007;26(2):216–229.

[17] Colli A, Pisana S, Fasoli A, *et al.* Electronic Transport in Ambipolar Silicon Nanowires. Physica Status Solidi (B). 2007;244(11):4161–4164. Available from: http://dx.doi.org/10.1002/pssb.200776154.

[18] Martel R, Derycke V, Lavoie C, *et al.* Ambipolar Electrical Transport in Semiconducting Single-Wall Carbon Nanotubes. Physical Review Letters. 2001;87(25):256805.

[19] Geim AK, Novoselov KS. The Rise of Graphene. Nature Materials. 2007;6(3):183–191.

[20] Rabaey, JM, Anantha PC, Nikolic B. Digital Integrated Circuits. Vol. 2. Englewood Cliffs, NJ: Prentice Hall, 2002.

[21] Lewis DL, Yalamanchili S, Lee HHS. High Performance Non-Blocking Switch Design in 3D Die-stacking Technology. In: VLSI, 2009. ISVLSI'09. IEEE Computer Society Annual Symposium on. IEEE; 2009. p. 25–30.

[22] Marrakchi Z, Mrabet H, Masson C, *et al.* Efficient Mesh of Tree Interconnect for FPGA Architecture. In: Field-Programmable Technology, 2007. ICFPT 2007. International Conference on. IEEE; 2007. p. 269–272.

[23] Wu CL, Feng TY. On a Class of Multistage Interconnection Networks. IEEE Transactions on Computers. 1980;100(8):694–702.

[24] Adams GBI, Agrawal DP, Siegel HJ. A Survey and Comparison of Fault-Tolerant Multistage Interconnection Networks. Computer. 1987;20(6): 14–27.

[25] Gaillardon PE, Clermidy F, O'Connor I, *et al.* Interconnection Scheme and Associated Mapping Method of Reconfigurable Cell Matrices based on Nanoscale Devices. In: Nanoscale Architectures, 2009. NANOARCH'09. IEEE/ACM International Symposium on. IEEE; 2009. p. 69–74.

[26] Rose J, Luu J, Yu CW, *et al.* The VTR Project: Architecture and CAD for FPGAs from Verilog to Routing. In: Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays. ACM; 2012. p. 77–86.

[27] Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification; Release 20160717. Available from: https://people.eecs.berkeley.edu/\∼alanmi/abc/.

[28] Luu J, Anderson JH, Rose JS. Architecture Description and Packing for Logic Blocks with Hierarchy, Modes and Complex Interconnect. In: Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays. FPGA '11. New York, NY, USA: ACM; 2011. p. 227–236. Available from: http://doi.acm.org/10.1145/1950413.1950457.

[29] Goeders JB, Wilton SJE. VersaPower: Power Estimation for Diverse FPGA Architectures. In: 2012 International Conference on Field-Programmable Technology; 2012. p. 229–234.

[30] Lamoureux J, Wilton SJE. Activity Estimation for Field-Programmable Gate Arrays. In: 2006 International Conference on Field Programmable Logic and Applications; 2006. p. 1–8.

[31] Yang S. Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0. Microelectronics Center of North Carolina (MCNC); 1991.

[32] Brglez F, Bryan D, Kozminski K. Combinational Profiles of Sequential Benchmark Circuits. In: IEEE International Symposium on Circuits and Systems (ISCAS). IEEE; 1989. p. 1929–1934.

[33] Ahmed E, The Effect of Logic Block Granularity on Deep-Submicron FPGA Performance and Density. MASc Thesis, University of Toronto. 2001;101.

[34] Synoposys Inc. HSPICE: The Gold Standard for Accurate Circuit Simulation; 2010. Available from: https://www.synopsys.com/content/dam/synopsys/verification/datasheets/hspice-ds.pdf.

[35] Sinha S, Yeric G, Chandra V, *et al.* Exploring Sub-20 nm FinFET Design with Predictive Technology Models. In: Proceedings of the 49th Annual Design Automation Conference. ACM; 2012. p. 283–288.