# Chapter 8
# Advanced Devices and Architectures


**Masato Motomura, Masanori Hariyama and Minoru Watanabe**

**Abstract** The last chapter of this book is for advanced devices and brand new architectures around FPGAs. Since the basic logic blocks of FPGAs are consisting of LUTs, they are called fine-grained reconfigurable architectures. In contrast, coarse-grained reconfigurable architectures use processing elements to improve the performance per power for computation-centric applications. Dynamic reconfiguration is also easily done in such an architecture, and the configuration data set is called a hardware context. By switching hardware context frequently, they can achieve better usage of semiconductor area. The next part is asynchronous FPGA which can be a breakthrough of high-performance operation with low-power consumption. The handshake mechanism, a key component of such architectures, is explained in detail. 3D implementation is another new trend, while 2.5D is now in commercial use. The last part of this chapter is for activities of optical techniques around FPGAs for drastic improvement I/O and reconfiguration performance.

**Keywords** CGRA · Hardware context · Asynchronous FPGAs
Optical I/O · Optical reconfiguration

## 8.1 Coarse-Grained Reconfigurable Architecture

As was explained in Chap. 1, FPGAs started as devices for prototyping small-scale logic circuits. As they become larger in accordance with the shrink in transistor size, the idea to use FPGAs as acceleration devices is getting more popular, as

M. Motomura
Hokkaido University, Sapporo, Japan
e-mail: motomura@ist.hokudai.ac.jp

M. Hariyama (✉)
Tohoku University, Sendai, Japan
e-mail: hariyama@tohoku.ac.jp

M. Watanabe
Shizuoka University, Shizuoka, Japan
e-mail: tmwatan@ipc.shizuoka.ac.jp



H. Amano (ed.), *Principles and Structures of FPGAs*,
https://doi.org/10.1007/978-981-13-0824-6_8

demonstrated in Chap. 7. This approach, known as reconfigurable computing or reconfigurable systems, is becoming more important as CPUs performance improvement are slowing down. It is natural to use an array of LUTs when the main purpose of an FPGA is prototyping.

As a device for reconfigurable computing, however, it may make more sense to use other primitive elements. To fit better for the acceleration of computing functions, such elements might be less versatile, but they should be more efficient in computing than LUTs. This is how coarse-grained reconfigurable architectures (CGRAs) have been proposed and investigated.

### *8.1.1 CGRA Basics and History*

Starting from the 1980s, CGRAs have been mostly presented by universities and startups. The well-known ones are PipeRench from CMU and XPP from PACT, in addition to others [1]. Recently proposed good examples of such architectures, both in Japan, are CMA from Keio University [2] and LAPP from NAIST [3]. As shown in Fig. 8.1, a CGRA can be represented as an array of operation units and memories, associated with a network structure connecting them. As for the operation units granularity, there are varieties such as: 4, 8, 16, and 32 bits. The finer the architecture is the more it becomes like an FPGA. On the other hand, the coarser it is the more it becomes like a traditional parallel processor. As for the instruction set, traditional arithmetic logic operations are commonly found, as well as extended instructions for customized acceleration of target applications. The array configuration may not necessarily be a two-dimensional one as in FPGAs, but also a one-dimensional array when a target application is sufficient with linear processing. Either dynamic switching on-chip routing networks or static switching interconnection fabrics are used for the network in Fig. 8.1.

### *8.1.2 CGRA Design Space*

Since CGRAs are equipped with operation units customized for given applications, they surpass FPGAs, in general, in processing performance and density. High density means more parallel operation units can be integrated into the same area. This also translates into better processing performance. In addition, configuration information can become much smaller compared to FPGAs. It is known that a major portion of configuration information is spent on interconnections: FPGA architectures require specifying bit-level interconnections. While in the CGRA case, interconnections are bundled to the coarse granularity specified by the architecture. Another important CGRA merit to note is its familiarity to design tools. This is very important since software programmers, in the reconfigurable computing field, are the users who map target applications onto the CGRA architecture.

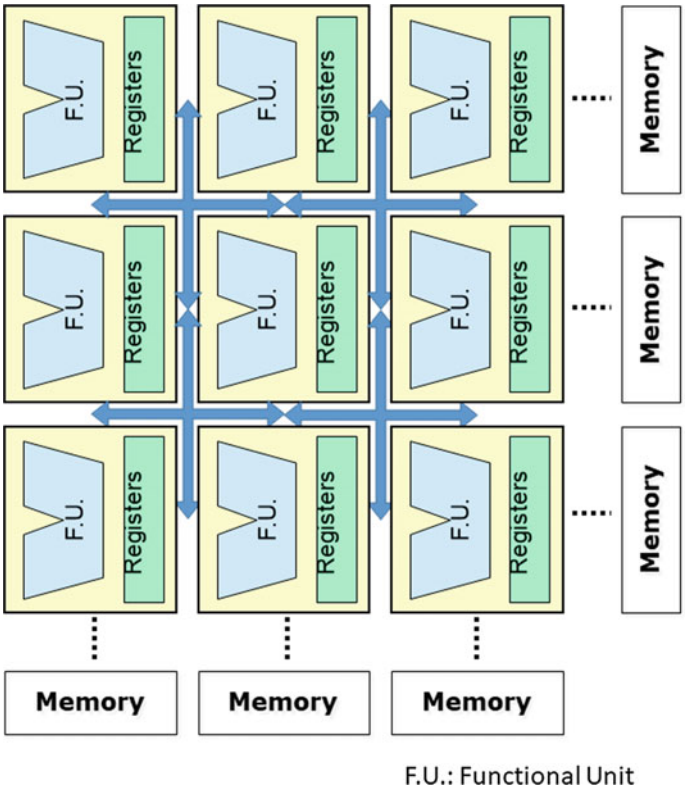



**Fig. 8.1** Generalized CGRA architecture

There are also drawbacks in the CGRA approach. First, they become less general purpose when compared to FPGAs because of their structure. Considering that FPGAs have occupied a large portion of the market, because it is a general-purpose device (at least from the HW prototyping point of view), this is a major issue to be carefully considered. Many CGRA architectures ended up being just research prototypes because of this reason.

Another issue to carefully consider when defining the CGRA architecture is to efficiently utilize the hardware-based computation as much as possible. For example, when choosing a single bit from an 8-bit data, an FPGA needs just to wire the desired bit. Whereas in the CGRA case, an 8-to-1 selector or shifter is required. This means that an overhead is incurred both in performance and area.

Based on considerations such as the ones above, CGRA architectures are mainly considered as a tightly coupled accelerator to a CPU core, but not a replacement to FPGA. For example, CMA and LAPP allow the CGRA core to directly access CPU registers so that it can accelerate sub-tasks of a CPU (such as frequently executed loops).

### 8.1.3 Dynamically Reconfigurable Architecture

When an FPGA is used as a computation device, a resource limitation issue may rise; i.e., what should be done when the required computation does not fit within the hardware resources of a given FPGA? This problem is easy to solve when FPGAs are considered for prototyping: Just increase the number of FPGA chips and make the connection between them. If only computation is concerned, just like software does, a reconfigurable computation solution should robustly account for variously sized applications. This is the reason why dynamically reconfigurable architectures were investigated.

One of the earliest works in this area is WASMII from Keio University, Japan [4]. This work proposed quite an advanced concept where hardware is considering as a set of pages, which can swap in and out (Fig. 8.2). In conventional operating systems, a virtual memory allows a large memory space that does not fit in a physically available memory, to be allocated to applications using a page-by-page swapping method. Similarly to virtual memory, in WASMII, the page-oriented hardware architecture allows virtual hardware, where a virtually large hardware can be put on a physically existing small reconfigurable device.

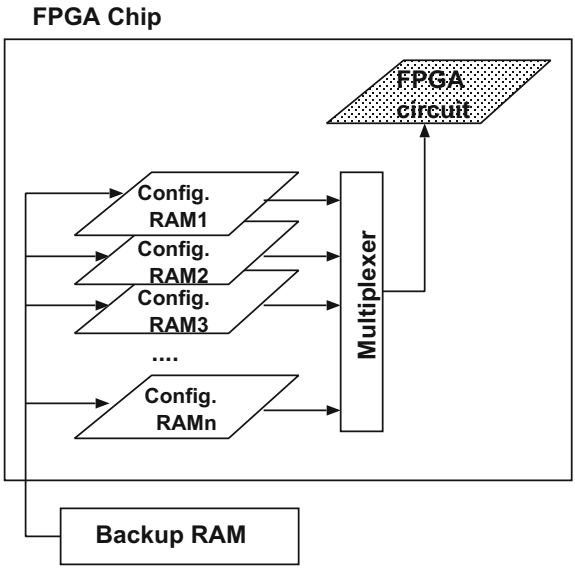



**Fig. 8.2** WASMII execution model

### 8.1.4 Case Study: DRP

Dynamically Reconfigurable Processor (DRP) is a coarse-grained, dynamically reconfigurable architecture proposed by NEC Corporation in 2002 [5]. The architecture was mainly developed for an IP core integrated into an SoC. This section overviews DRPs as an example of dynamically reconfigurable architectures (they are also an example of a CGRA described in the previous section).

Figure 8.3 shows its basic architecture. A processing element (PE), that constitutes a two-dimensional array, is composed of two general-purpose 8-bit ALU, register file, and instruction memory. The two ALUs have bit-manipulation instructions such as bit mask/select, so that it can cover bit-level operations that FPGA can handle well. PEs are interconnected to each other with an 8-bit-width hierarchical bus. Bus selectors connect those ALUs and a register file with vertical/horizontal buses.

An instruction memory stores a set of hardware configurations, from which one configuration is selected, i.e., hardware dynamic reconfiguration. Each instruction includes operation codes for the two ALUs, as well as control bits for the bus selectors. For example, it is possible to bypass the register file in a PE and connects the ALU outputs to inputs of other PEs in a flow-through manner. Ordinary processors store outputs produced at one cycle into registers and then read those registers for following cycles. The PEs in a DRP, on the other hand, spatially connect plural PEs for constructing a customized datapath.

The PE array is associated with an state transition controller (STC), which is responsible for managing the dynamic reconfiguration. A basic role of an STC is to dispatch instruction pointer to the array: Each PE receives the pointer, then selects, and reads a specified instruction. The STC has a sequencer which keeps track of the state transitions of a given application. When a new pointer is dispatched, all the instructions of PEs, as well as all the interconnections in the PE array, change at once. This operation can be interpreted such that the hardware configuration changes among the datapath contexts stored in the memory. This makes it similar to the WASMII's concept explained in Fig. 8.2. Conditional state transitions require branches which need to examine branch conditions. Information required for this examination is returned back from the datapath as event signals to the STC.

A DRP core is made up of multiple tiles of PEs. Each tile has its own STC to control the reconfiguration. Hence, multi-tiled DRPs can run multiple state machines in parallel. There is also a mechanism to connect multiple tiles and control them by a single state machine.

The execution model of DRPs is usually learned from high-level synthesis tool which is a tool to compile a program written in high-level language like C to an executable hardware. Generally speaking, a program has control flows which are composed of conditional branches and loops, etc., and data flows which are trees of data handling operations. High-level synthesis tools compile control flows to finite-state machines, and data flows to hardware datapaths. DRP architectures feature clear one-to-one correspondence with this generic high-level synthesis model: STCs manage finite state machines, and PE arrays handle datapath. Here, a datapath asso-

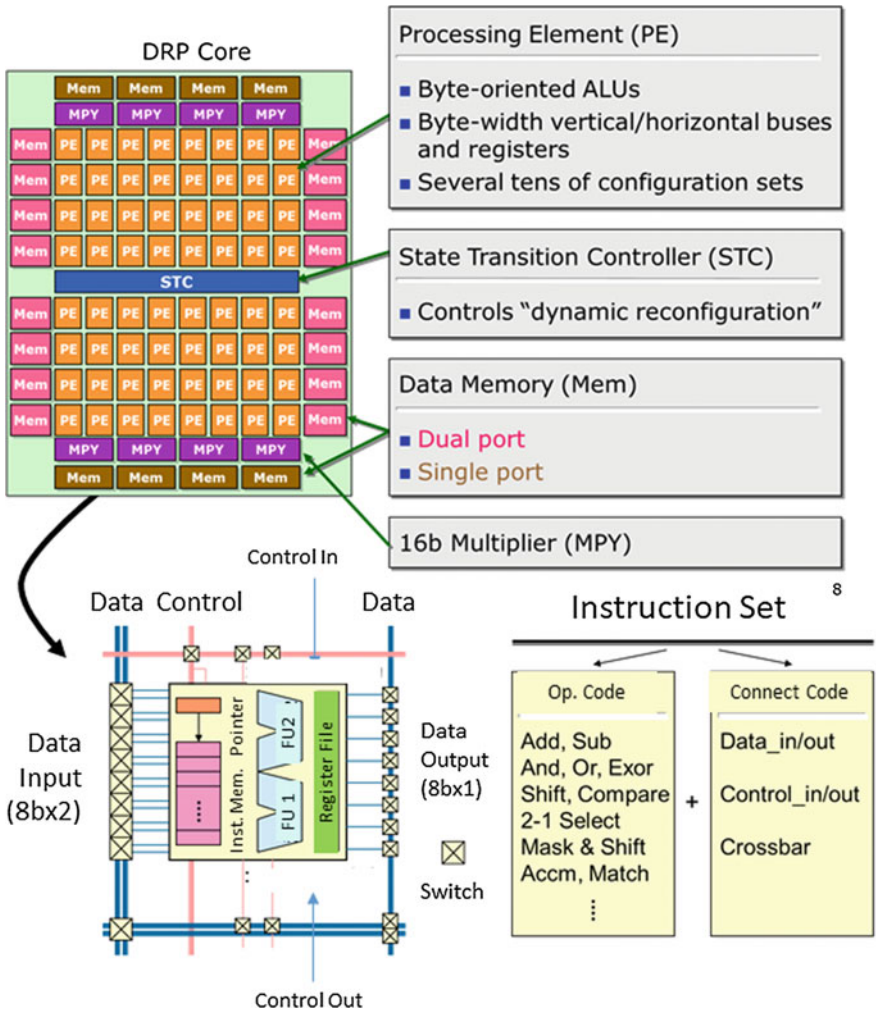



**Fig. 8.3** DRP architecture

ciated with each state is called "context." Contexts are generated when (1) they are associated with states in the control flow, or (2) when there is a resource limitation and a context should be divided into multiple ones.

DRPs feature GUI-based high-level synthesis-oriented tool flow (Fig. 8.4). Context generation is all handled by this tool, and designers do not have to worry about how to decompose hardware datapaths. The DRP core, which is now owned by Renesas Electronics and a commercially used dynamically reconfigurable architecture, has been used in products such as video cameras and digital cameras.

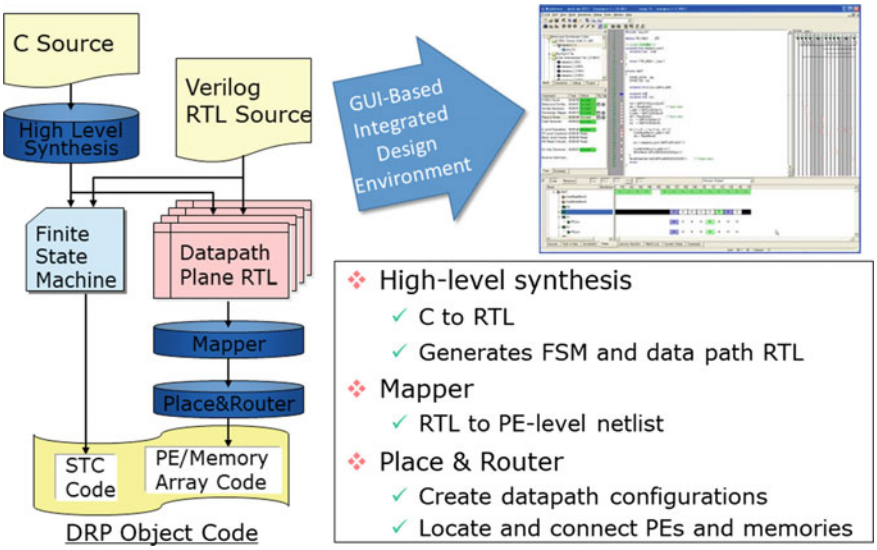



**Fig. 8.4** DRP design tool

### *8.1.5 Relation to Parallel Processors*

An important consideration in conducting dynamic hardware reconfiguration is the need to load large amount of configuration information at once. Typically, dynamically reconfigurable architectures feature one to several clock cycle latencies for this hardware context switch (for DRP, it is less than a single cycle). This is the reason why such architectures, including DRPs, adopt CGRAs in order to reduce configuration information.

Dynamically reconfigurable CGRA hardware may look very similar to on-chip many-core parallel processors (such as Xeon Phi from Intel). The difference becomes clear when their execution models are examined:

- **Dynamically reconfigurable CGRA**: A block of instructions are first spatially mapped on an array of processing elements (it constitutes a hardware context). Then, the hardware contexts are multiplexed in time (space to time order).
- **On-chip many-core parallel processor architecture**: It first assigns a block of instructions to a single processor as a thread. Then, multiple threads are mapped onto a processor array among which the synchronization will take place from time to time (time to space order).

### 8.1.6 Other Architecture Examples

FPGA-based (i.e., fine-grained) dynamically reconfigurable hardware architectures were proposed in Tabula [1]. Tabula exploits dynamic reconfiguration for speeding up FPGA operational frequency. That is, it divides the critical path into several segments and maps them to different hardware contexts. Tabula was proposed for realizing over-GHz range FPGA for high-end applications (the project was suspended in 2015).

## 8.2 Asynchronous FPGA

### 8.2.1 Problems of Conventional Synchronous FPGAs

In conventional synchronous circuits, some serious problems become obvious as the miniaturization of semiconductor process continues. Figure 8.5 shows the global clock network of synchronous FPGAs. The global clock network is connected to the clock inputs of all registers. A register loads data only at the rising edge of the clock pulse. As soon as the data is loaded, it appears on the output. The data on the register output is used as the input of the logic circuits. The result of these circuits is used as the input of the following register. FPGAs usually have much larger circuits and have much more registers than application-specific integrated circuits (ASICs). Therefore, conventional synchronous FPGAs have larger parasitic capacitance of the global clock network and require much more clock buffers to reduce clock skews. This causes the following problems:

- The clock network consumes larger power.
- The speed is limited by the clock skews.

As for conventional synchronous FPGAs, lowering the power consumption is not so easy compared to ASICs due to the following reasons:

**Difficulty to use clock gating**: Clock gating is a major technique to reduce the power consumption in ASICs. It prevents the input of a circuit from causing unnecessary signal transitions when the circuit is unused. When using clock gating for ASICs, designers should carefully design the customized clock network to avoid clock skews, and the clock network is fixed at the manufacturing phase. As for FPGAs, the clock network cannot be customized for a certain circuit since various circuits are implemented on FPGAs. Moreover, it is not recommended to use the clock network that can be reconfigured for the clock gating, since reconfiguring the clock network causes faults due to clock skews.

**Difficulty to use power gating**: Power gating is another major method to reduce the power consumption in ASICs. It turns the circuits power off when they are not in use and wake them up just before being used. The power gating requires control circuits

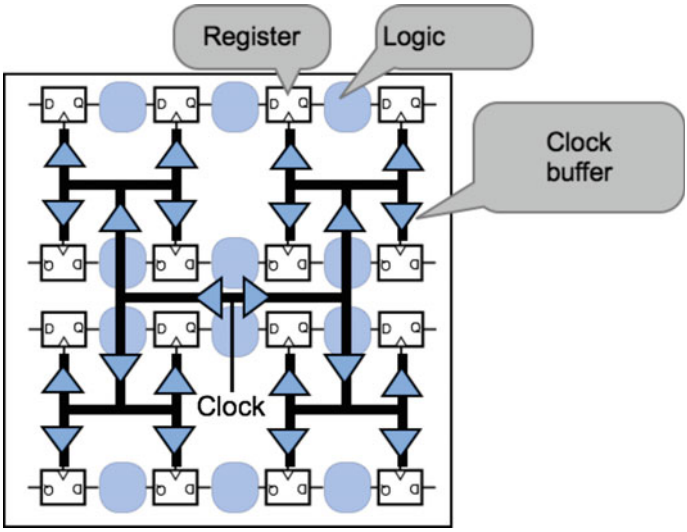


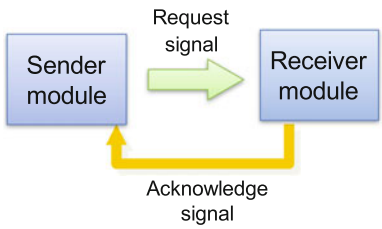

**Fig. 8.5** Global clock network of synchronous FPGAs

and dedicated connections to distribute these control signals. Especially in FPGAs, these overheads are significantly large due to the flexibility of FPGAs.

### *8.2.2 Overview of Asynchronous FPGAs*

In order to solve the problems of the synchronous FPGAs, FPGAs based on asynchronous circuits are proposed. Figure 8.6 shows the basic behavior of an asynchronous circuit. Data transfers between processing modules are done using a handshake protocol as follows. At first, the sender sends the data and a request signal to the receiver. The request signal is used to inform the receiver of the data arrival. After receiving the request signal, the receiver takes the data in and sends the acknowledge signal to the sender. The acknowledge signal is used to inform the sender that the

**Fig. 8.6** Basic behavior of an asynchronous circuit



receiver has completed receiving the data. After the acknowledge signal reception, the sender sends a new data in the same manner.

The advantages of asynchronous circuits over synchronous ones are summarized as follows:

**No dynamic power consumption in the inactive state**: An asynchronous circuit does not consume power consumption when not processing data. This is because it does not have the global clock network that always transfers the clock pulse.
**Low peak power or peak current**: In asynchronous circuits, processing modules start to process data after they receive it. Since the data arrival times vary from each other, the durations of the power consumption peaks (and current peaks) of the modules vary from each other as well. As a result, the average power consumption of the whole circuit becomes low.
**Low-Level electromagnetic radiation**: Since the peak current is low as described before, the level of the electromagnetic radiation is also low.
**Robust to the fluctuation of the supply voltage**: Even when the supply voltage decreases slightly, it is guaranteed that the output of the circuit is correct thanks to its clock-less operation.

The major disadvantage of asynchronous circuits is their larger amount of hardware for control, e.g., circuits that detect data arrival, and the additional wires for acknowledge and request signals.

In asynchronous circuits, there are three major types of handshake protocols [6]:

(1) Bundled data protocol,
(2) Four-phase dual-rail protocol, and
(3) Level-encoded dual-rail (LEDR) protocol.

The bundled data protocol is also called single-rail protocol. It represents one bit of data by using a single-rail-like synchronous circuits. A word to be transferred consists of multiple data bits and 1-bit request signal. Hence, the overhead for control is only 1-bit per word, which can be considered as very small. The disadvantage of the bundled data protocol is that it requires a timing constraint for the request signal; the request signal must arrive at the receiver module after the data. In order to ensure this, a delay buffer is usually inserted into the request signal wire as shown in Fig. 8.7.

Figure 8.8 shows the data transfer and the encoding of the four-phase dual-rail protocol. In this protocol, a word to be transferred from the sender has 2 bits: 1 bit for data and 1 bit for the request signal, as depicted in Fig. 8.8a. The receiver sends the 1-bit acknowledge signal back to the sender. In general, 1-bit acknowledge signal is enough for multiple words. Figure 8.8b illustrates the encoding. Data "0" and "1" is represented by code words $(D_t, D_f) = (0, 1)$ and $(D_t, D_f) = (1, 0)$, respectively. As a separator between data, the spacer $(D_t, D_f) = (0, 0)$ is used. Note that $(D_t, D_f) = (1, 1)$ is invalid. Let us consider an example of data transfer represented in Fig. 8.8c, where data "1", "1", "0", and "0" are transferred. Since the sender sends a code word and the spacer alternatively, the receiver can detect the data. In the four-phase dual-rail protocol, the racing problem caused by the arrival timings of data and the request signal does not occur. In other words, $D_t$ and $D_f$ do

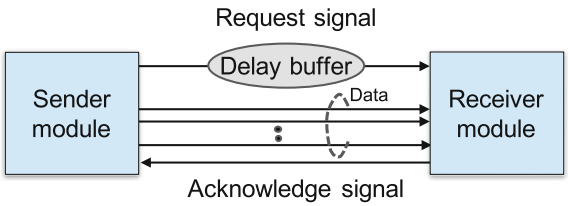



**Fig. 8.7** Bundled data protocol

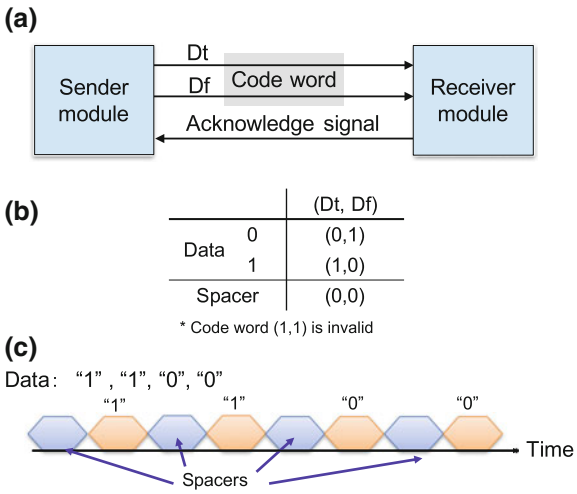



**Fig. 8.8** Four-phase dual-rail protocol

not change at the same time. This is because the code words are designed such that the Hamming distance between any two code words is one. Hence, the four-phase dual-rail protocol is more robust for timing variations than the single-rail protocol.

The circuit of the four-phase dual-rail protocol is simpler than that of the LEDR protocol, described below, since the data corresponds to a single code word. The disadvantage of the four-phase dual-rail protocol over the LEDR protocol is its lower throughput due to the insertion of the spacer.

The LEDR protocol is suitable for high-throughput data transfer. Figure 8.9 shows the data transfer and the encoding of the LEDR protocol. The way of data transfer is the same as the four-phase dual-rail protocol, as presented in Fig. 8.9a. The big difference between the four-phase dual-rail and LEDR protocols is the encoding, as demonstrated in Fig. 8.9b. Data "0" is encoded by two different code words: $(V, R) = (0, 0)$ in phase 0 and $(V, R) = (0, 1)$ in phase 1. Data "1" is also encoded by two different code words: $(V, R) = (1, 1)$ in phase 0 and $(V, R) = (1, 0)$ in phase 1. Let us consider an example of a data transfer using the phases, as shown in Fig. 8.9c, where data "1", "1", "0", and "0" are transferred. In the LEDR protocol, the code word in phase 0 and the code word in phase 1 are alternatively transferred. The receiver can detect the change of data by detecting the change of phases. Since the LEDR protocol dose not need spacers, it can achieve high throughput. The disadvantage of this protocol is that it requires larger circuits since one data value has two different code words.

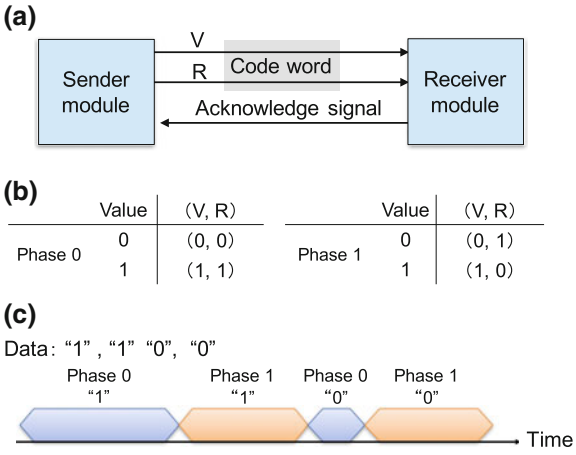



**Fig. 8.9** LEDR protocol

Hereafter, we explain asynchronous FPGAs. Asynchronous FPGAs using the bundled data protocol have been proposed [7, 8]. Although they benefit from the used small circuits, the main disadvantage is their low performances due to the large delay buffers inserted in the request signal wire to ensure the correct behavior for various datapaths.

As for asynchronous FPGAs protocol, the dual-rail protocol is ideal since it can avoid the racing problem mentioned above without any timing constraints on the data nor the request signal. Figure 8.10 shows a basic architecture based on the dual-rail

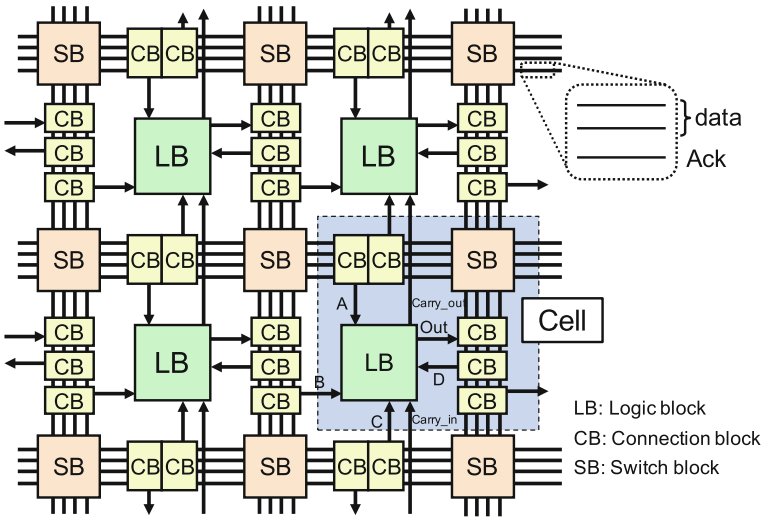



**Fig. 8.10** Asynchronous FPGA based on the dual-rail protocol

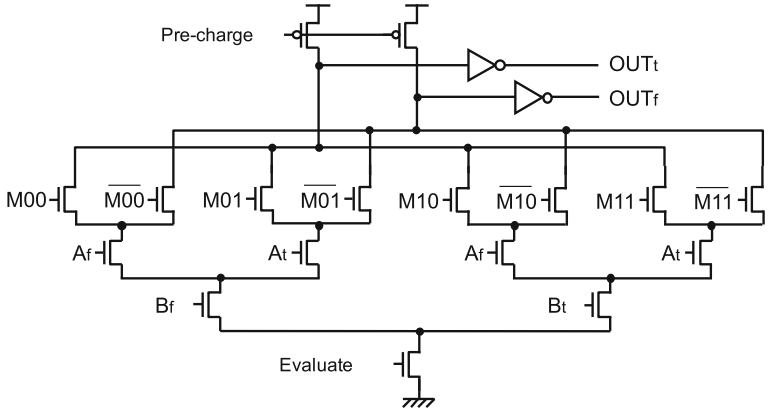



**Fig. 8.11** LUT for the four-phase dual-rail protocol (2-input LUT)

protocol [9]. Similar to conventional synchronous FPGAs, logic blocks (LBs) are connected to each other via connection blocks (CBs) and switch blocks (SBs). An interconnection unit consists of wires for code words and the acknowledge signal.

Among the dual-rail protocols, the four-phase dual-rail protocol is employed to implement small circuits [10, 11]. Figure 8.11 shows the structure of an look-up table (LUT), where, for simplicity, the numbers of inputs and outputs are limited to two and one, respectively. The four-phase dual-rail protocol is suitable for dynamic circuits which are used for area-efficient design. This is because the pre-charge signals and the evaluation signal are easily generated from the spacer. The number of bits of the configuration memory is $2^N$ for an $N$-input LUT like conventional synchronous FPGAs. In the case of Fig. 8.11, the 2-input LUT has a 4-bit configuration memory (M00, M01, M10, M11). The output $(OUT_t, OUT_f)$ is determined according to the configuration memory and the external inputs $(A_t, A_f)$ and $(B_t, B_f)$.

### *8.2.3 Design for Low Power, High Throughput, and Modularity*

For low power, asynchronous circuits can provide some design information, and intelligent control is realized based on this information. For example, the receiver module can detect the data arrival by using the request signal; the sender module can know whether the receiver module is ready to load data. In [12], fine-grained adaptive control of the supply voltage is proposed to reduce the dynamic power consumption. According to the state of the receiver module, the sender module adaptively controls its supply voltage and processing speed. In [13], fine-grained power gating is proposed to reduce the static power consumption caused by the leakage current of transistors. Each module detects the arrival of its inputs data by

using the request signal that is sent from the sender module. If the inputs do not arrive within a predefined time, the module automatically turns the supply voltage of the core circuit off. When the inputs come, the module wakes its core circuit up [13].

In order to achieve high throughput, fine-grained pipelining is frequently employed for high throughput datapaths [11, 14–16]. From the point of view of data transfer, the protocol hybrid architecture is proposed, where the LEDR protocol is used for high-throughput data transfer and the four-phase dual-rail protocol is used for simple datapaths [9, 17]. Moreover, the hybridization of synchronous circuits and asynchronous circuits is proposed [18]. When a significant amount of input data continuously arrives, synchronous circuits are considered to be efficient in terms of power consumption. On the other hand, asynchronous circuits are efficient for the case when the data arrival is less uniform. Based on this observation, an LUT is designed to be used in both asynchronous circuit and synchronous circuit while sharing the circuit. Depending on the used applications, the blocks of LUTs are configured as asynchronous or synchronous circuits. Note that both of asynchronous and synchronous circuits can coexist on a single FPGA. High throughput and low power can be optimally achieved by combining the asynchronous and synchronous circuits based on their aptitudes for different applications.

One major problem in asynchronous circuits is their difficulty to program. The reason is that the design for modularity is not easy in asynchronous circuit. To solve this problem, design methods using handshake components are proposed for general asynchronous circuits [19, 20]. Handshake components are basic building blocks to describe the data flow and control flow, including arithmetic/logic operations, conditional branch, and sequence control. Designing circuits is easily done by connecting such handshake modules. In [21], an asynchronous FPGA is proposed whose logic blocks are suitable to implement the handshake components.

## 8.3 3D FPGA

As described before, FPGAs consist of a configuration memory, programmable interconnection units, and programmable logic circuits to achieve a high degree of flexibility. Such redundant resources lead to a lower area efficiency compared to ASICs. Moreover, the complex interconnection causes a large delay and degrades the performance. These problems will be more serious in the near future since the miniaturization of the semiconductor manufacturing process nears the physical limit.

Based on this background, applying 3D integration technologies such as TSV (Through Silicon Vias) [22–24] to FPGAs is strongly desired. 3D FPGAs are classified into two types: heterogeneous and homogeneous.

Figure 8.12 shows the conventional 2D FPGA architecture and the 3D heterogeneous architecture. As shown in Fig. 8.12 (bottom), different resources such as logic blocks, routing blocks, and a configuration memory are distributed into different layers; the resources in different layers are connected by using interconnections such as TSV. Therefore, the heterogeneous architecture can increase the resource density per

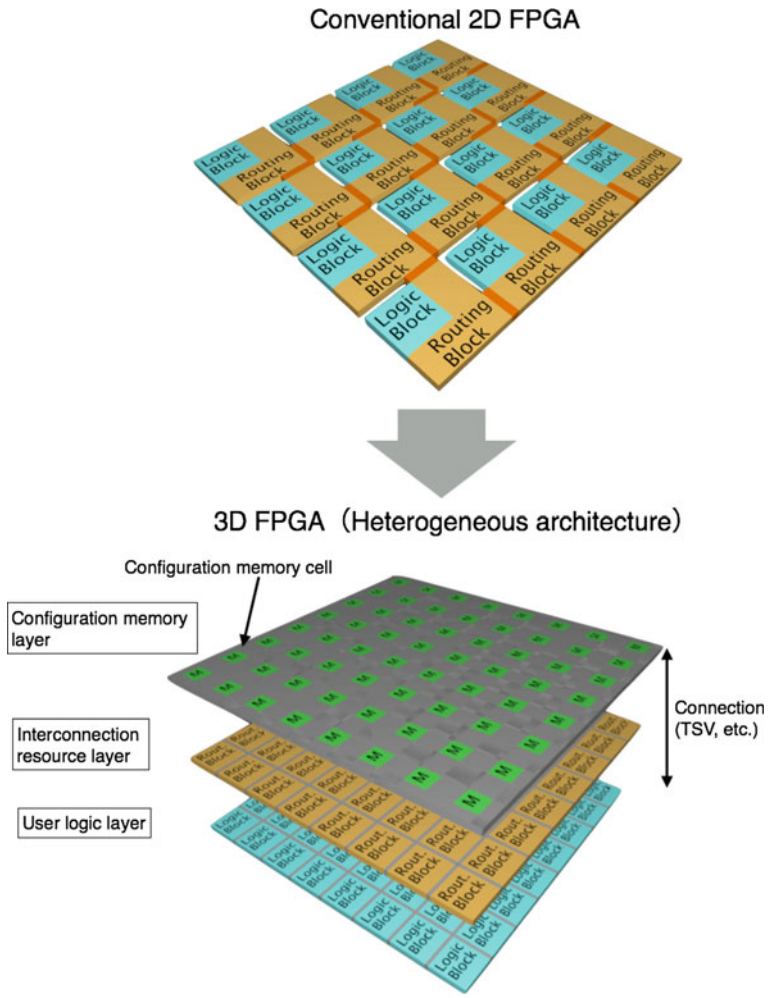



**Fig. 8.12** 3D FPGA (heterogeneous architecture)

footprint [25–28]. The scalability along the vertical direction of the heterogeneous architecture is lower than that of the homogeneous architectures, described below, since the number of layers is limited by the number of resource types.

Figure 8.13 shows the 3D homogeneous architecture. Each layer has the same functions as a 2D FPGA, that is, logic blocks, routing blocks, a configuration memory. The routing block is designed such that it connects the logic blocks in the same layer and also connects the routing blocks in different layers via vertical connections [29–32]. Hence, the homogeneous architecture is an extension of the 2D FPGA architecture to the third dimension. When the number of stacked layers increases in accordance with the progress of the 3D integration technology, the total circuit

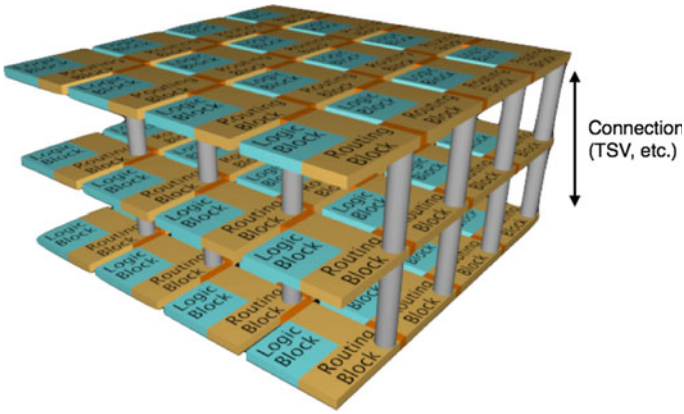



**Fig. 8.13** 3D FPGA (homogeneous architecture)

size of a 3D FPGA can also increase linearly. Moreover, the performance could be improved compared to 2D FPGAs. When mapping circuits with complex topology onto a conventional 2D FPGA, the connected circuit blocks are not always mapped onto near logic blocks. As a result, it may result in mapping with long wires. 3D FPGAs can map such circuits onto near logic blocks by using different layers.

Hereafter, the issues of 3D FPGAs are summarized. The first issue is the challenge facing the technologies to use for the vertical connections, which should be cheap and highly reliable. Moreover, when increasing the number of layers in the homogeneous architecture, the thermal radiation can be critical as well as CAD support [33–39].

## 8.4 High-Speed Serial I/O

Microsoft recently announced a server using the Stratix V FPGAs to be used in its data center for Bing search engine [40]. Although the introduction of FPGAs has increased the power consumption by 10%, it enhanced the throughput by 95% when compared to the software implementation. Thus, this has emphasized the effectiveness of FPGAs. In this implementation, the 10-Gbps high-speed communication port of the FPGA is used for a mutual network that is indispensable in a data center. In recent years, this case underscores the further emerging importance of networks that can leverage FPGA applications.

As described in Chap. 3, recent FPGAs provide many general-purpose inputs/outputs (GPIOs) that can accommodate various devices such as memories. GPIOs readily realize an interface with various devices connected to an FPGA at a high bandwidth. Recent FPGAs are equipped with serial I/Os that allow high-speed communications of Gbps order in addition to GPIOs, as highlighted in the data center case described above. Accordingly, short-distance communications between FPGA

chips, middle-distance communications between systems including FPGA chips, and long-distance network communications between systems including FPGA chips have been implemented at higher speeds. Xilinx Inc. and Altera Corp. mutually compete in terms of performance, and they are locked in a development race to mount more high-speed serial I/Os on their own cutting-edge FPGAs. As a result, FPGAs communication performance has improved rapidly in recent years. The importance of serial I/Os in FPGAs is anticipated to further increase in the future. Therefore, this section describes these high-speed serial I/Os in the Stratix family devices, as an example.

### *8.4.1 LVDS*

The Stratix family supports differential interfaces of small amplitude such as Low Voltage Differential Signaling (LVDS) [41], Mini-LVDS [42], and Reduced Swing Differential Signaling (RSDS) [43]. Mini-LVDS and RSDS are standards derived from LVDS for computer displays, formulated, respectively, by Texas Instruments Inc. and National Semiconductor. This section describes LVDS, which has been standardized by ANSI/TIA/EIA-644. The Stratix family adopts LVDS that satisfies this standard [44, 45].

LVDS is a one-way signal transmission standard under which a signal is transmitted from a transmitting side to a receiving side using two lines, as indicated in Fig. 8.14. For example, in cases where a signal "1" is transmitted from the transmitting side, transistors (1) and (2) in Fig. 8.14 are turned ON for transmission. In this event, the current flows from the current source on the transmitting circuit to the upper line via transistor (1). A terminator is mounted on the receiving circuit. The great portion of the current flows into the terminator and returns to the transmitting circuit via the other line to flow into VSS via transistor (2). At this time, the potential between both terminals of the terminator on the receiving side rises to about +350 mV. A differential amplifier on the receiving circuit detects this state. Then, the receiving circuit determines it as a signal of "1".

On the other hand, when transmitting a signal of "0", transistors (3) and (4) on the transmitting circuit are turned ON. Thereby, the current flows from the current source through the lower line. Similarly to the description above, the current passes through the terminator, returns to the transmitting terminal via the upper line, and flows into VSS via transistor (3). At this time, a potential of −350 mV occurs at the terminator. Consequently, the current flows in an opposite direction according to the transmitted value of "1" or "0", and a potential of +350 mV occurs on the receiving circuit. The receiving circuit judges whether the transmitted value is "0" or "1" by detecting this potential. This small amplitude allows high-speed and low-power communications.

The Stratix IV GX 40-nm FPGAs are equipped with 28–98 LVDS ports that support high-speed communications up to 1.6 Gbps [46]. The number of ports described above is expressed by the number of full-duplex channels through which transmission and reception are conducted simultaneously; e.g.,"28 ports" denote that there

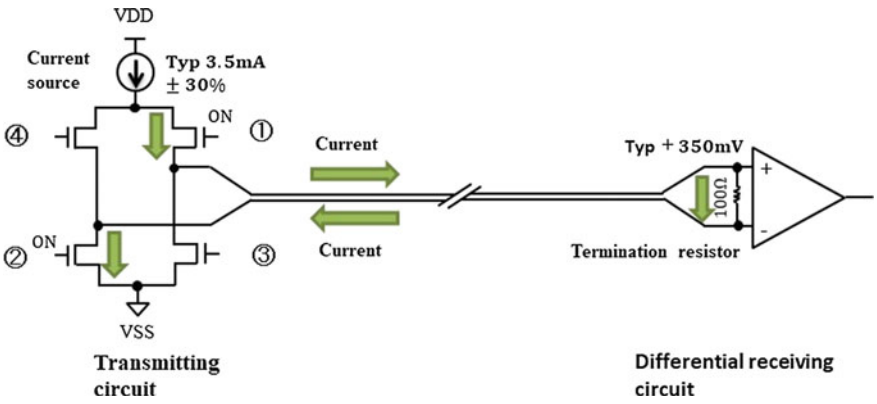



**Fig. 8.14** Schematic view of LVDS transmitting and receiving circuits

are 28 LVDS ports for transmission and 28 LVDS ports for reception. The number of available ports might be different depending on the package, even for FPGAs of the same size. Meanwhile, the Stratix V manufactured by the 28-nm TSMC process supports LVDS ports up to 1.4 Gbps [44]. The Stratix V GX device is equipped with 66–174 full-duplex LVDS ports [45].

The Stratix family I/Os for high-speed communications have a built-in hard macro serializer/deserializer (SerDes) circuit up to 10 bit. It is difficult to build a communication circuit that can directly operate as fast as 1.4–1.6 Gbps inside an FPGA. However, the hard macro of the serializer can easily convert a signal from a parallel transmitting circuit using, for example, a 10-bit FIFO operating at a low clock frequency into a high-speed serial signal as fast as 1.4–1.6 Gbps. The block diagram of a transmitting circuit is presented in Fig. 8.14. The deserializer at a receiving circuit can convert a 1-bit high-speed serial signal to a 10-bit parallel signal in the same way, so that a receiving circuit can be constructed with a FIFO operating at a low clock frequency. Furthermore, a resistance of 100 Ω that terminates the differential signal at the receiving side of an LVDS is programmable in the Stratix V, so that high-speed communications between FPGA chips can be easily implemented without extra parts just with the board design considering the impedance. A guideline for board design is provided from Altera. One report of the relevant literature is particularly useful [46] in describing the board design (Fig. 8.15).

### *8.4.2 28-Gbps High-Speed Serial I/O*

Stratix still supports more high-speed serial I/O in addition to LVDS. For instance, Stratix V GX FPGA and Stratix V GS FPGA are equipped with up to 66 high-speed communication ports that operate at 12.5 Gbps. The Stratix V GT FPGA is equipped with four 28-Gbps high-speed communication ports in addition to 32 14.1-Gbps communication ports.

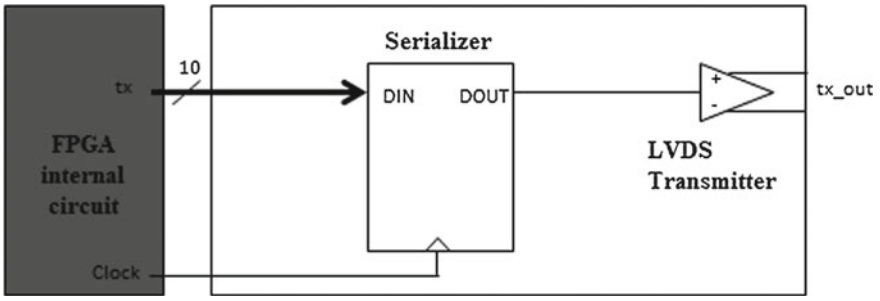



**Fig. 8.15** Schematic block diagram of LVDS transmitting circuit of Stratix V

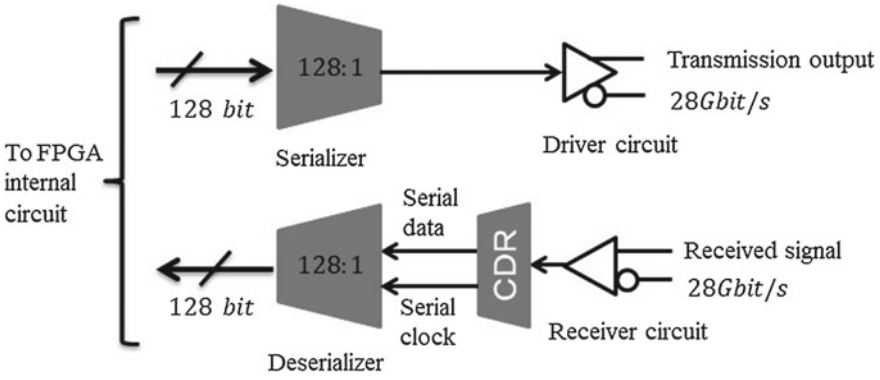



**Fig. 8.16** Stratix V 28-Gbps transmitting circuit

The Stratix has an embedded hard macro of a serializer/deserializer between 128 and 1 bit, so that 128-bit parallel data (provided via FIFO) is converted into a 1-bit high-speed serial signal by the hard macro of the serializer. Finally, the signal is transmitted via a driver circuit in the same manner as the LVDS, previously described in Fig. 8.16. Similarly, at the receiving side, the deserializer parallelizes the received 28-Gbps high-speed serial signal into 128 bits to pass it through low-speed FIFO. An error-free receiver circuit includes a clock data recovery (CDR) circuit which can detect a phase shift between the internal clock and received data and can correct it continually. Xilinx also supplies FPGAs that support such high-speed serial communications. For example, the Virtex-7 HT FPGA has 28-Gbps communication ports, which provide excellent communication performance.

### *8.4.3 FPGA with 120-Gbps Optical I/O*

As described above, a transfer rate of the order of Gbps can be implemented even with metal wiring. However, optical communications are beneficial in terms of power


**Fig. 8.17** Appearances of 0.7424-mm pitch LGA sockets mounted on Stratix IV package (left) with MicroPOD optical modules (Avago Technologies Ltd.) (right)

consumption for a distance of 10 m or more, as indicated in a report by Altera. Optical modules have been adopted by Xilinx and Altera and have been mounted on FPGA boards in recent years to support optical communications on the board level. However, Altera and Avago Technologies developed and announced a more pioneering optical FPGA with an optical communication interface mounted on an FPGA chip in March, 2011 [47]. Although it is only a trial chip, and no plans for marketing have been announced, its advanced architecture is introduced hereafter.

This optical FPGA is prototyped based on Stratix IV GT FPGA with 11.3-Gbps I/Os for high-speed communications. The salient difference in this optical FPGA from conventional FPGAs is that two of the four corners on its package are provided with sockets of a 0.7424-mm pitch land grid array (LGA), as presented in Fig. 8.17: one for transmission and the other for reception. Each socket is plugged with a dedicated optical module for optical communications supplied by Avago.

This Stratix IV GT FPGA has 32 full-duplex I/O ports for high-speed communications, 12 of which are allocated to these optical I/Os. Twelve 11.3 Gbps high-speed serial I/Os on the FPGA are connected to the sockets for transmission, and 12 11.3 Gbps high-speed serial I/Os are connected to the sockets for reception. The optical communication module is as small and compact as 8.2 mm $\times$ 7.8 mm, as illustrated in Fig. 8.18.

Twelve vertical cavity surface emitting lasers (VCSELs) are embedded in optical communication modules of the transmitting side, whereas 12 GaAs PIN photodiodes are mounted in the optical communication module of the receiving side. Optical communications are conducted through a 12-core fiber cable. The VCSEL lasers can be aligned in two dimensions like common transistors on an integrated circuit. The VCSEL can build a compact laser array [48]. This optical module allows 10.3125-Gbps data transfer per channel consisting of one VCSEL and one GaAsPIN photodiode. In all, 12 channels in the module realize an overall transmission speed of 120 Gbps. In spite of such high-speed communications, a multimode fiber of OM4 grade accommodates long-distance transmissions as far as 150 m. It is highly likely that such optoelectronics will become indispensable when ultra-high-speed I/Os over 28 Gbps become necessary in the future.

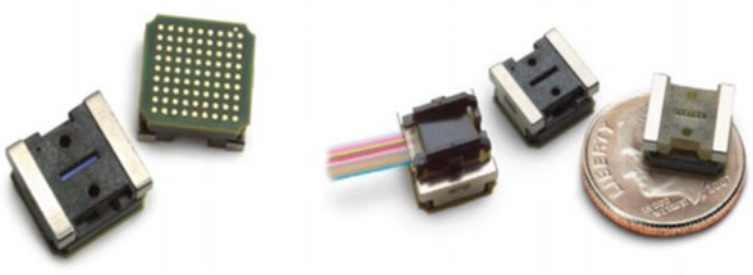


**Fig. 8.18** A MicroPOD optical module is mounted on an FPGA package with an LGA socket. Its packaging area is 8.2 mm × 7.8 mm

### *8.4.4 Optically Reconfigurable Architecture*

**Optically Reconfigurable Architecture by Caltech**: Caltech announced an optically reconfigurable gate array (ORGA) using a holographic memory in April, 1999 [49]. This ORGA, the world's first FPGA that can be reconfigured optically, consists of a holographic memory, a laser array, a photodiode array, and an FPGA component. Because its gate array component has a fine-grained gate array structure that is identical to that of conventional FPGAs, its fundamental function appears to be the same as that of the existing FPGAs to device users. However, unlike conventional FPGAs, its programming method is optical reconfiguration. The holographic memory of this optically reconfigurable gate array is used as a read only memory (ROM). Multiple circuit information can be stored in the holographic memory in advance. Then, this circuit information is addressed by the laser array. It is read out as a two-dimensional diffraction pattern. This diffraction pattern is then recognized by the photodiode arrays, transferred serially to the FPGA, and reconfigured. Studies at Caltech have demonstrated the benefits of this optically reconfigurable gate array: It can use large-scale properties of the holographic memory, it can carry multiple circuit information, and its circuit information is programmable within 16–20 μs (Fig. 8.19).

### *8.4.5 Japanese-Made ORGA*

Research on ORGAs was also started at Kyushu Institute of Technology in Japan in January 2000. The research base was later moved to Shizuoka University. The research is still in progress. Since Caltech has reported no research on optically reconfigurable devices since, Japan is presumably the only research base for ORGAs in the world at present. Japanese ORGAs under development are introduced hereafter.

Several types of ORGAs are undergoing research and development in Japan, including ORGAs that adopt an electrically rewritable spatial light modulation

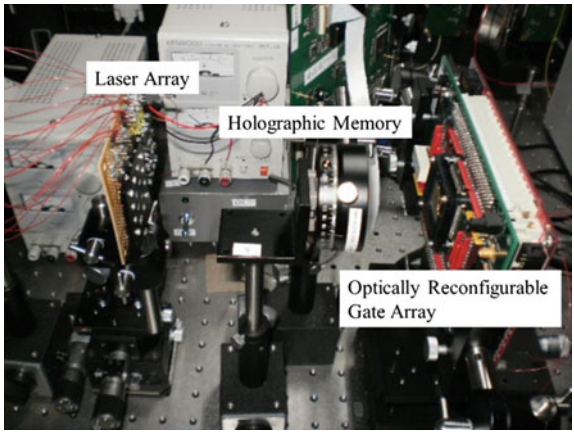



**Fig. 8.19** Optically reconfigurable gate array (Shizuoka University)

element as a holographic memory [50, 51] and ORGAs that employ a laser array and microelectromechanical systems (MEMSs) together to address a holographic memory [52]. An ORGA of a simple architecture, similar to that by Caltech, consisting of a holographic memory, a laser array, and a gate array VLSI is introduced here.

ORGAs under development in Japan adopt a fine-grained gate array like Caltech's ORGA, so that the function of the gate array is the same as the existing FPGAs. However, Japanese devices employ a fully parallel configuration, different from Caltech's, where the gate array has many photodiodes. Two-dimensional light patterns generated by the holographic memory are read in a fully parallel mode by these photodiodes. This optical reconfiguration approach allows dynamic reconfiguration of the gate array in a cycle of 10 ns using large amounts of circuit information stored in advance in the holographic memory. To date, ORGAs with circuit information of 256 types have been developed.

An ORGA stores circuit information in a holographic memory. Theoretically, a holographic memory can store as much as 1 Tbit of information within a volume of one lump of sugar. So, its high capacity is expected to be promising also for the next-generation optical memories [53]. The aim of the ORGA is to implement a virtual large-scale gate array by storing much circuit information in a holographic memory using its high capacity [54, 55].

A holographic memory has no fine structures as it is the case for those of existing SRAMs, DRAMs, or ROMs. It can be made simply by consolidating materials such as photopolymers. Accordingly, its production is extremely simple and inexpensive. Information is written on a holographic memory with a dedicated writer using the interference of light. The writer splits a coherent laser beam into two optical paths, an object light representing the binary pattern of circuit information and a reference light, and records the interference pattern of these two light waves on the holographic memory. Greater amounts of information can be recorded by varying the incident angle of a reference light and the irradiation position on a hologram. The stored information can be read out using a laser beam with the identical coherent light as

the reference light. In the case of an ORGA, circuit information is usually written in with a writer before the device starts to operate. Its holographic memory is used as a ROM while the device is in operation. Because a large amount of circuit information can be stored in a holographic memory, it is possible to select it with a laser array and to dynamically conduct the reconfiguration.

The holographic memory has a characteristic that it can be used even if it has been contaminated by impurities or partial defects. Holographic memory is usually irradiated with a coherent laser beam as a reference light when reading information. This light undergoes phase modulation or amplitude modulation in the holographic memory and is read out from it. The intensity of light at an arbitrary point is determined by the phase of the gathered light from the whole holographic memory. A collection of lights in phase brightens the point, whereas a collection of lights of diverse phases darkens it. Because information is read out by the superposition of many light waves, the holographic memory has long been known as a robust memory that is useful even if it has defects. Research on radiation-hardened ORGAs is in progress using this characteristic of robustness of the holographic memory. The optoelectronic device has not been used widely yet. However, it might overcome obstacles that are difficult to resolve solely by using integrated circuits in the far future.

## References

1. R. Tessier, K. Pocek, A. DeHon, Reconfigurable computing architectures, in *Proceedings of the IEEE*, vol. 103, no. 3, pp. 332–351 (March 2015)
2. K. Masuyama, Y. Fujita, H. Okuhara, H. Amano, A 297MOPS/0.4 mW ultra low power coarse-grained reconfigurable accelerator CMA-SOTB-2, in *Proceedings of The 10th International Conference on Reconfigurable Computing and FPGAs (ReConFig)* (Dec 2015)
3. J. Yao, Y. Nakashima, N. Devisetti, K. Yoshimura, T. Nakada
4. X.-P. Ling, H. Amano, WASMII: a data driven computer on a virtual hardware, in *IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 33–42 (April 1993)
5. T. Toi, T. Awashima, M. Motomura, H. Amano, Time and space-multiplexed compilation challenge for dynamically reconfigurable processors, in *Proceedings of the 54th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 31–39 (Aug 2011)
6. T.E. Williams, M.E. Dean, D.L. Dill, Efficient self-timing with level-encoded 2-phase dual-rail(ledr), in *Proceedings University of California/Santa Cruz Conference Advanced Research*, VLSI (1991)
7. R. Payne, Self-timed FPGA systems, in *Proceedings International, Workshop Field Program Logic, Applications* (1995)
8. V. Akella, K. Maheswaran, PGA-STC: programmable gate array for implementing self-timed circuits. Int. J. Electron. **84**(3) (1998)
9. M. Kameyama, Y. Komatsu, M. Hariyama, Anasynchronous high-performance FPGA based on LEDR/four-phase-dual rail hybrid archicture. Proc. 5th Int. Symp. HEART (2014)
10. R. Manohar, Reconfigurable asynchronous logic, in *Proceedings IEEE Custom Integrated, Circuits Conference* (2006)
11. R. Manohar, J. Teifei, An asynchronous dataflow FPGA architecture. IEEE Trans. Comput. **53**(11) (2004)
12. M. Hariyama, M. Kameyama, S. Ishihara, Z. Xie, Evaluation of a self-adaptive voltage control scheme for low-power FPGA. J. Semicond. Tech. Sci. **10**(3) (2010)

13. M. Kameyama, S. Ishihara, M. Hariyama, A low-power FPGA based on autonomous fine-grain power gating. IEEE Trans. VLSI Syst. **19**(8) (2011)
14. Achronix SpeedSter22 HP (2011), http://www.achronix.com/products/speedster22ihp.html
15. B. Devlin, M. Ikeda, K. Asada, A 65 nm gate-level pipelined self-synchronous FPGA for high performance and variation robus operation. IEEE J. Solid-State Circuits **46**(11) (2011)
16. B. Devlin, M. Ikeda, K. Asada, A gate-level pipelined 2.97 GHz self synchronous FPGA in 65 nm FPGA CMOS. Prof. ASP-DAC (2011)
17. M. Kameyama, Y. Komatsu, H. Hariyama, An asynchronous high-performance FPGA based on LADR/four-phase-dual-rail hybrid architecture. Proc. HEART (2014)
18. Y. Tsuchiya, M. Komatsu, H. Hariyama, M. Kameyama, R. Ishihara, Implementation of a low-power FPGA based on synchronous/asynchronous hybrid architecture. IEICE Trans. Electron. **E94-C**(10) (2011)
19. A. Bardsley, Implementation balsa handshake circuits. Ph.D. Thesis (Eindhovan Universithy of Technology, 1996)
20. M. Roncken, R. Saeijs, F. Schalij, K. Berkel, J. Kessels, The VLSI programming language trangram and its translation into handshake circuits, in *Proceedings European Conference in Design Automation, EDAC* (1991)
21. M. Kameyama, Y. Komatsu, H. Hariyama, Architecture of an asynchronous FPGA for handshake-component-based design. IEICE Trans. Fund. **E88-A**(12) (2005)
22. A.W. Topol, D.C. La Tulipe, L. Shi, D.J. Frank, K. Bernstein, S.E. Steen, A. Kumar, G.U. Singco, A.M. Young, K.W. Guarini, M. Ieong, Three-dimensional integrated circuits. IBM J. Res. Develop. **50**(4), 5 (2006)
23. G. Katti, A. Mercha, J. Van Olmen, C. Huyghebaert, A. Jourdain, M. Stucchi, M. Rakowski, I. Debusschere, P. Soussan, W. Dehaene, K. De Meyser, Y. Travaly, E. Beyne, S. Biesmans, B. Swinne, 3D stacked ICs using Cu TSVS and die to wafer hybrid collective bonding. IEEE Int. Electron Dev. Meeting IEDM (2009)
24. K. Banerjee, S.J. Souri, P. Kapur, K.C. Saraswat, 3-D ICs: a novel chip design for improving deep-submicrometer interconnect performance and sisytes-on-chip intergration. Proc. IEEE **89**(5) (2001)
25. M. Lin, A. El Gamal, Y.-C. Lu, S. Wong, Performance benefits of monolithically stacked 3-D FPGA. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **26**(2) (2007)
26. R. Le, S. Reda, R. Iris Bahar, High-performance, const-effective heterogeneous 3D FPGA Architectures, in *Proceedings the 19th ACM Great Lake Symposium VLSI* (2000)
27. T. Naito, T. Ishida, T. Onoduka, M. Nishigoori, T. Nakayama, Y. Ueno, Y. Ishimoto, A. Suzuki, W. Chung, R. Madurawe, S. Wu, S. Ikeda, H. Oyamatsu, World's first monolithic 3D-FPGA with ifi SRAM over 90 nm 9 layer Cu CMOS, in *Proceedings Symposium VLSI Technology* (2010)
28. Y.Y. Liauw, Z. Zhang, Z. Zhang, W. Kim, A.E. Gamal, S.S. Wong, Nonvolatile 3D-FPGA with monolithically stacked RRAM-based configuration memory. ISSCC (2012)
29. A. Gayasen, V. Narayanan, M. Kandemir, A. Rahman, Designing a 3-D FPGA: switch box architecture and thermal issues. IEEE Trans. VLSI Syst. **16**(7) (2008)
30. F. Furuta, T. Matsumura, K. Osada, M. Aoki, K. Hozawa, K. Takeda, N. Miyamoto, Scalable 3D-FPGA using wafer-to-wafer TSV interconnect of 15 Tbps/w, 33 Tbps/mm$^2$. IEEE Trans. VLSI Syst. (2013)
31. M.J. Alexander, J.P. Cohoon, J.L. Colflesh, J. Karro, G. Robins, Three-dimensional field-programmable gate arrays, in *Proceedings of 8th Annual IEEE International ASIC Conference and Exhibit* (1995)
32. S.A. Razavi, M.S. Zamani, K. Bazargan, A tileable switch module architecture for homogeneous 3D FPGAs, in *Proceedings IEEE International 3D System Integration* (2009)
33. A. Rahman, S. Das, A.P. Chandrakasan, R. Reif, Wiring requerement and three-dimensional integration technology for field programmable gate arrays. IEEE Trans. VLSI Syst. **11**(1) (2003)
34. C. Ababei, H. Mogal, K. Bazargan, Three-diminsional place and route for FPGAs. IEEE Trans. Comput. Aided Design Integr. Circuits Syst. **25**(6) (2006)

35. M. Amagasaki, Y. Takeuchi, Q. Zhao, M. Iiea, M. Kuga, T. Sueyoshi, Architecture exploration of 3D FPGA to minimize internal layer connection, in *ACM/IEEE International Conference on 3D Systems Integration* (2015)
36. M.J. Alexander, J.P. Cohoon, J.L. Colflesh, J. Karro, E.L. Peters, G. Robins, Placement and routing for three-dimensional FPGAs, in *4th Canadian Workshop Field Programmable Devices* (1996)
37. M. Lin, A. El Gamal, A routing fabric for monolithically stacked 3D FPGA, in *Proceedings ACM/IEEE International Conference on FPGA* (2007)
38. N. Miyamoto, Y. Matsuomto, H. Koike, T. Matsumura, K. Osada, Y. Nakagawa, T. Ohmi, Development of a CAD tool for 3D-FPGAs, in *IEEE International Conference on 3D Systems Integration* (2010)
39. Y. Kwon, P. Lajevardi, A.P. Ch, D.E. Troxel, A 3-D FPGA wire resource prediction model validated using a 3-D placement and routing tool, in *Proceedings of SLIP '05* (2005)
40. A. Putnam, et al., A reconfigurable fabric for accelerating large-scale datacenter services, in *ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 13–24 (2014)
41. The Telecommunications Industry Association (TIA), Electrical characteristics of low voltage differential signaling (LVDS) interface circuits, PN-4584 (May 2000)
42. National Semiconductor: RSDS Intra-panel Interface Specification (May 2003)
43. Texas Instruments, mini-LVDS Interface Specification (2003)
44. Altera Corporation: Stratix IV Device Handbook, vol. 1 (June 2015)
45. Altera Corporation: Stratix V Device Handbook, vol. 1 (June 2015)
46. Altera Corporation: High speed board design Ver.4.0, Application Note 75 (Nov 2001)
47. M. Peng Li, J. Martinez, D. Vaughan, Transferring high-speed data over long distances with combined FPGA and multichannel optical modules (2012)
48. H. Li, K. Iga, Vertical-cavity surface-emitting laser devices, in *Springer Series in Photonics*, vol. 6 (2003)
49. J. Mumbra, D. Psaltis, G. Zhou, X. An, F. Mok, Optically programmable gate array (OPGA). Opt. Comput. (1999)
50. H. Morita, M. Watanabe, Microelectromechanical configuration of an optically reconfigurable gate array. IEEE J. Quant. Electron. **46**(9), 1288–1298 (Sept 2008)
51. Y. Yamaguchi, M. Watanabe, Liquid crystal holographic configurations for ORGAs. Opt. Comput. **47**(28), 4692–4700 (2008)
52. Y. Yamaji, M. Watanabe, A 4-configuration-context optically reconfigurable gate array with a MEMS interleaving method, in *NASA/ESA Conference on Adaptive Hardware and Systems*, pp. 172–177 (June 2013)
53. A. Ogiwara, M. Watanabe, Optical reconfiguration by anisotropic diffraction in holographic polymer-dispersed liquid crystal memory. Appl Opt **51**(21), 5168–5188 (July 2012)
54. H.J. Coufal, D. Psaltis, G.T. Sincerbox, Holographic data storage, in *Springer Series in Optical Sciences*, vol. 76 (2000)
55. S.-L.L. Lu, P. Yiannacouras, R. Kassa, M. Konow, T. Suh, An FPGA-based Pentium in a complete desktop system, in *ACM/SIGDA 15th International Symposium on Field Programmable Gate Arrays*, pp. 53–59 (2007)