J. Inst. Eng. India Ser. B
https://doi.org/10.1007/s40031-020-00508-y

REVIEW PAPER

Reconfigurable FPGA Architectures: A Survey and Applications

Praveenkumar Babu[1] · **Eswaran Parthasarathy**[1]

Received: 23 August 2019 / Accepted: 21 October 2020

Abstract Reconfigurable computing is a potential paradigm which has been effectively performing mostly in the developments of devices likely Field Programmable Gate Arrays (FPGAs). This paper illustrates the reconfigurable architecture of FPGA and its types. Most widely used high-speed computation fabrics utilized in reconfigurable computing are FPGAs. This paper demonstrates the architectures used in reconfigurable computing and shows the various advantages of using reconfigurable computing design over conventional Application-Specific Integrated Circuits for achieving high level of performance for a desired application. The survey deals with the architecture of FPGAs and their types in detail. This paper also explains the highlights and challenges of fine-grained and coarse-grained architectures. FPGAs have supported partial reconfiguration over the few years. This survey also includes the partial reconfiguration techniques and the various applications of reconfigurability.

Keywords Reconfigurable computing · FPGA · ASIC · Fine-grained architecture · Coarse-grained architecture · Partial reconfiguration

Introduction

Reconfigurable computing is an architecture deal with the flexibility of high performance of hardware and software components by processing with computing platforms like Field-programmable gate arrays (FPGAs). They are reprogrammed to specific application based on their functionality needs after manufacturing. This characteristic of FPGAs differentiates from ASICs, which are designed for a particular application or task [1]. FPGAs find their usage in widely different fields, where their reprogrammable ability renders various benefits over ASICs implementations [2, 3]. This capability of FPGAs permits hardware designs to upgrade or reuse after implementation. When compared to ASICs or full-custom design, FPGAs offer many advantages, which include the low cost in the silicon chip area, an increase in its performance and low power consumption [4]. On the other hand, vendors of these FPGAs include Xilinx, Altera, and Atmel are analyzed, and they are compared by various parameters.

The architectures of reconfigurable systems can be divided into two categories depending on their granularity which are fine-grained and coarse-grained architectures. These architectures are configured by parallelism such as bit-level and word-level, respectively. The coarse-grained reconfigurable systems provide effective area utilization, low power consumption and efficient performance compare to fine-grained systems. The other important aspect is partial reconfiguration (PR) which defines the process of modifying one or multiple portions of the logic blocks while the other sections are unchanged. PR is favorable for embedded hardware systems, because these systems are adjustable to computation in different conditions while processing data.

✉ Eswaran Parthasarathy
eswaranp@srmist.edu.in

[1] Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu 603203, India

Published online: 17 November 2020

This paper is summarized in this way: In "FPGA as a Reconfigurable Architecture" section, it gives a general view of FPGA architectures and types of FPGAs. "Fine-Grained Reconfigurable Architectures" and "Coarse-Grained Reconfigurable Architectures" sections include fine-grained and coarse-grained architectures, respectively. In "Partial Reconfiguration" section, it explains partial reconfiguration and some examples. Finally, the survey concludes with the wide area of applications of reconfigurable computing are mentioned in "Applications" section.

FPGA as a Reconfigurable Architecture

FPGAs provide better reconfiguration in which bit-level configuration is performed. The common FPGA architecture includes an array of programmable logic blocks which implement digital logic functions. The reconfiguration can be classified into two modes. They are static and dynamic reconfiguration. The configuration of the reconfigurable unit is performed only one time and cannot be modified when the task is running is known as static or compile-time reconfiguration. In dynamic reconfiguration, the configuration can be changed during the course of execution. The reconfiguration time should be minimized to enhance the overall performance of the reconfigurable processors when compared with that of general-purpose microprocessors. The minimization of reconfiguration time needs to be focused on [5]. FPGAs also support implementation level reconfiguration which offers several product families by suppliers like Xilinx®, Intel® Altera®, etc. In FPGA architecture, a logic block consists of few logic cells made up of Look-up tables (LUTs), full-adder and D-flip flop as shown in Fig. 1.

The architecture of FPGA composed of three major constituents such as programmable logic blocks, programmable routing (interconnects) and Input/Output (I/O) blocks.

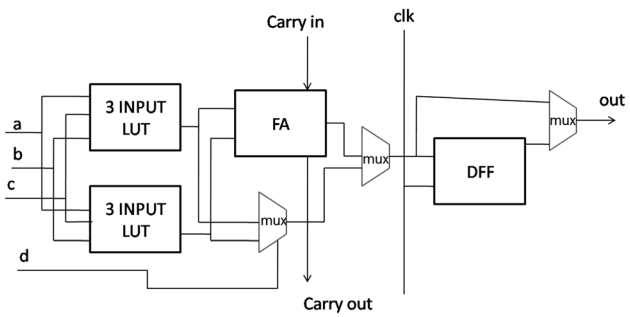

Fig. 1 Structure of a logic cell

Programmable Logic Blocks

The programmable logic block gives basic computational and some storage elements which find their applications in digital logic systems. A basic logic element (LE) comprises of programmable combinational logic, a flip-flop and a carry logic for reducing delay and area cost. Recent FPGAs contain various combination of blocks such as multiplexers and dedicated memory blocks. Configuration memory is needed for these programmable logic blocks to perform any specific task of each LE [1]. These logic blocks require routing channels to connect each other and I/O blocks to interface with external signals.

Programmable Routing

The programmable routing architecture comprises of pre-manufactured programmable wiring segments and switches which are organized in horizontal and vertical routing channels. It provides a connection between logic blocks and I/O blocks. In routing architecture, hardened multiplexers are used to save the SRAM cells and pass transistors are used for the connection of output pins. They are used to bring logic elements together like clusters [4]. The programmable logic elements are connected by interconnecting resources. Generally, these resources are configurable, because the signal path can be determined during compile or run time before the time of fabrication. This flexibility of interconnection between logic blocks can be used for mapping the reconfigurable hardware with a variety of circuit structures based on their requirements.

There are primarily four global routing architectures: Island, Cellular, Long-line, and Row. In island architecture, configurable logic blocks (CLBs) look like islands surrounded by routing interconnect, as shown in Fig. 2. CLBs are gridded on a 2D mesh-like structure. The

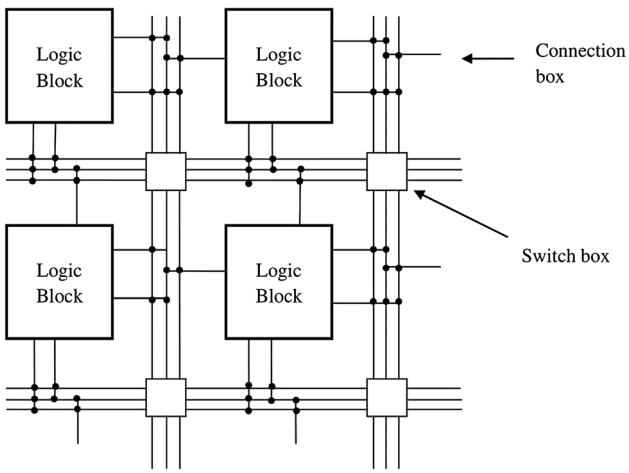

Fig. 2 A general structure of island architecture

programmable routing network connects I/O blocks around FPGA chip.

Programmable I/O

The logic blocks and routing architecture to the external components are interfaced by using programmable I/O pads. The I/O pads are surrounded by a logic circuit which form as I/O cells which occupy a large area of the FPGA's total utilization. Because of variations in the voltages (input voltage and reference voltage), the complexity of the design of I/O programmable blocks is high. It is important to select the design standards of I/O architecture [1] and these standards support for increasing the required silicon chip area for I/O cells. The development of FPGA architecture results in the addition of more dedicated programmable function blocks. These blocks contain up to four I/O elements (IOE). They are partitioned into two categories. They are two row I/O blocks for row interconnects and column I/O blocks for column interconnects.

The I/O Blocks in Xilinx Virtex FPGA are arranged in groups on the periphery of the device which can be used independently, or they can be combined in the group connected directly to a switch matrix [2] as shown in Fig. 3. The special functional blocks like ALUs, embedded processors, memories, multiplexers, generic DSP blocks have been added to the FPGA based on the requirement for various applications.

Anti-Fuse-Based FPGAs

FPGAs use anti-fuses are having two terminals which are connected to inner and outer layers of antifuses, and a dielectric is kept in between these layers as shown in Fig. 4. Initially, there is no current flow between the layers

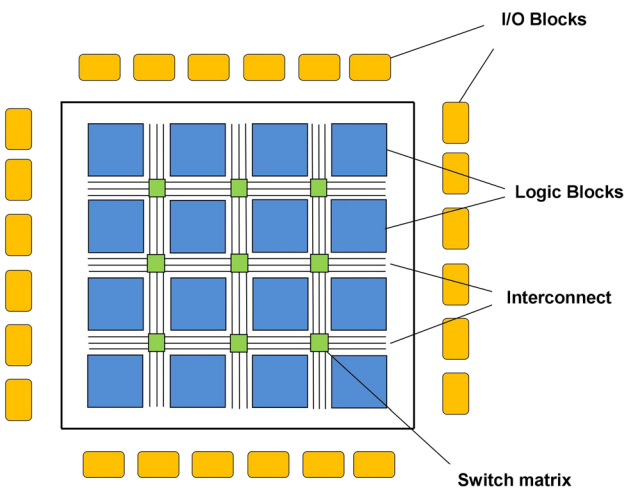

Fig. 3 FPGA architecture

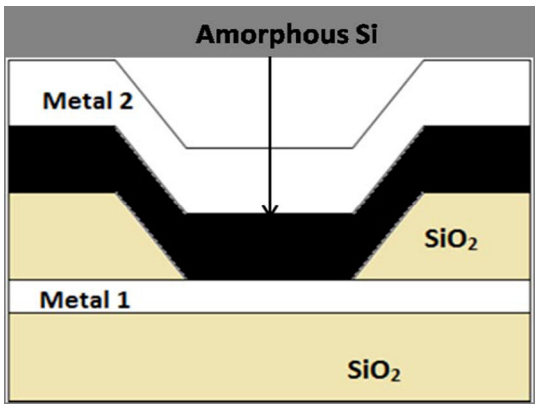

Fig. 4 Actel PLICE antifuse

due to the resistance of the dielectric is high. The power dissipation is large when high voltage is applied, which affects the dielectric and results in melting it. This process significantly decreases the resistance, and a permanent link will be formed and connects the two layers [6]. Since antifuses can be built using modified CMOS technology, they are very much suitable for FPGAs [7].

The most important advantages of the anti-fuse chips are small area utilization, low resistance-capacitance (RC) delays in the routing process, low resistance, and parasitic capacitance. This type of FPGAs cannot be repeatedly reprogrammed. Hence, they are otherwise called as one-time programmable FPGAs since they are programmed once by the user and cannot be changed anymore.

SRAM-Based FPGAs

SRAM cells are used to store configuration data and program the routing interconnect. In SRAM-based FPGA, the output of SRAM Cells can control the functionality bits of the logic blocks and their interconnections [6]. The configuration of the logic blocks and the connection can be done by Static RAM (SRAM). These FPGAs may include in the boards of Xilinx (Spartan and Virtex families), Actel (now Microsemi), Lattice, QuickLogic, Atmel and Altera (FLEX, Stratix, and Cyclone). The foremost feature of SRAM-based module is that FPGAs can be reprogrammed or configured numerously. The value assigned to the SRAM cells can be changed to make a new connection or a new function. The complexity of this technology is SRAM cells occupy a large chip area. Since the device is volatile, the data stored in the static memory is not available if there is a power failure [8]. To keep the configuration data and store it into the FPGA-device, non-volatile memories or external sources are required at every power-up.

Flash-Based FPGAs

These types of FPGAs use flash as a major source for configuration storage, and there is no need for SRAM. The main advantage of the flash-based FPGAs is their low power consumption. The radiation effects do not affect the performance of Flash-based FPGAs. Some of the devices support this technology may include Actel(Igloo and ProASIC3), Xilinx (XC2C Family, XC95 Family), Lattice (Mach X02, Mach XO, LFXP, iSP Mach), Altera (Max V, Max II) and Atmel ATF families. In Flash-/EEPROM-based programming technology such as Actel ProASIC chips, there is a floating gate between the two transistors stores the programming information. There is a sensing transistor which is used to write and verify the voltage of floating gate while the other is the switching transistor, which is used to erase the floating gate. Table 1 shows the power comparisons of flash and SRAM-based FPGAs.

Fine-Grained Reconfigurable Architectures

Fine-grained reconfigurable architectures (FGRAs) comprise of Look up tables (LUTs) made up of a small number of logic blocks. These architectures provide better performance and high flexibility based on the amount of complexities in the bit-level configuration [12]. Programmable Logic Devices (PLDs) support these fine-grained architectures since logic blocks are as small and simple as the macro cells. Atmel AT40K is the best example for fine-grained architecture made up of small identical cells with an array of 4 x 4 sector size arranged symmetrically, as shown in Fig. 5. It consists of 8-sided core cells, and they are connected to each other in all directions. These small cells in this architecture execute Boolean functions and provide greater functionality based on the requirements. [13]

The logic block in this architecture consists of logic elements and some flip-flops. The logic elements require more number of data bits for configurations, since the array of symmetrical cells has several configuration points to perform small computations [13]. These programmable logic blocks are useful for image processing, decoding, and encryption [14].

Table 1 Comparisons of power parameters for flash- [9] and SRAM-based FPGAs [10]

Parameters (mW)	Flash-based FPGAs	SRAM FPGAs
Configuration power	Low	High
Static power (standby)	Ultra low	High
Active power	Low	High
Low power (sleep)	Ultra low	Low

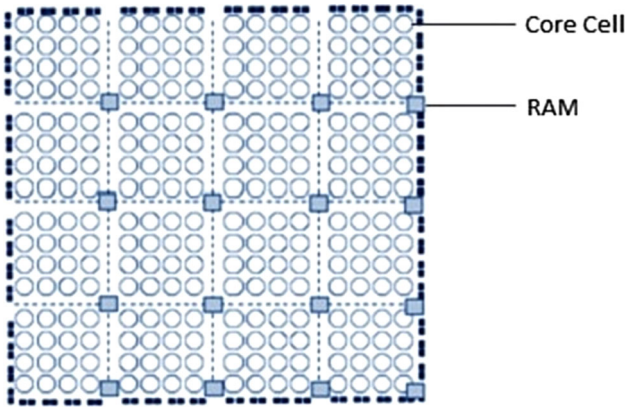

Fig. 5 ATMEL AT40K

Fine-grained dynamically reconfigurable (FDR) devices composed of a large number of identical reconfigurable LEs which ensure configuration into LUTs and interconnects randomly. The configuration improves the flexibility of allocation of hardware resources between interconnects and LUTs considerably based on its requirements.

The drawbacks of FGRAs are large routing area, reduced clock speed, huge configuration data, poor area utilization and reduction in bandwidth, etc. Some of the major setbacks are listed below.

Low performance and high power consumption: The configuration logic blocks (CLBs) are connected by programmable switch boxes for the formation of word-level modules. This leads to a decrease in performance and power consumed by the device is quite high. [15]

Large configuration context and time: The amount of time required for configuration is high for more number of configuration bits. If the configuration is at the bit level, individual configuration signals can be applied to every CLBs and interconnecting wires. This shows more configuration context should be transferred from the memory and therefore configuration time is increased. When reconfiguration are required frequently, the large reconfiguration time may degrade the performance [4].

Large routing overhead: The interconnection of CLBs increases routing overhead to form a word-level parallelism or datapath functions[12].

Poor area utilization: As CLBs are not suitable for performing logic operations, they are used to pass signals to each other for routing purposes. Generally, FPGAs available at commercial market utilized nearly 90 percent of the chip area for routing resources [16].

Huge context memory: The requirement of context memory is also increased due to the large reconfiguration contexts and complex datapath functions. So external memories are used to store the reconfiguration context.

Table 2 Comparison of types of FPGAs [11]

Parameters	SRAM FPGAs	Flash-based FPGAs	Anti-fuse-based FPGAs
Volatility	Yes	No	No
Reprogrammability	Yes	Yes	No
Area utilization for	High	Moderate	Low
Storage element size	(6 transistors)	(1 transistor)	(0 transistors)
Switching capacitance (fF)	1–2	1–2	≤ 1
Switching resistance (Ω)	≈ 1000	≈ 1000	≈ 100
In system programmable	Yes	Yes	No

This will further increase the reconfiguration time (Table 2).

Coarse-Grained Reconfigurable Architectures

Coarse grain reconfigurable architectures (CGRAs) encompass functional units (FUs) arranged like a mesh-type network, programmable interconnecting wires, and memory (Table 3). Unlike the fine-grained programmable logic, blocks are incapable of controlling the functions, and the coarse-grained logic blocks can perform multiple bit datapath and complex operations. CGRAs have several advantages when compared with FGRAs [17].

Minimal configuration contexts and time: The interconnections are configured efficiently by control (configuration) bits in the configuration memory. The coarse-grain reconfigurable units (CGRUs) require very fewer control bits for the specific operations. Since interconnecting wires are configured at datapath level, a few control bits are required to set up the interconnection among the units. The reconfiguration time is very much reduced as a result of the low configuration context, which allows coarse-grain reconfigurable systems to find their uses in various purposes [15].

Reduced context memory size: Being coarse-grained in nature, CGRAs acquire small configuration context overhead. The context memory size is reduced concerning the reduction in configuration contexts. The switching of configuration from one to other with minimum context overhead permits the usage of configuration memory efficiently.

Increase in performance and low power consumption: Since the design of interconnections is optimal and the implementation of CGRUs is hardwired, the performance of CGRUs is high, and power consumption is also low.

Area efficiency and decrease in routing overhead: CGRUs are permanently connected units for specific designs. They are not designed by the combination of CLBs and interconnecting wires, ensuring reduced routing overhead and area utilization is also improved [13].

Marco Lanuzza et al., presented MORA (Multimedia Oriented Reconfigurable Array) a new approach of CGRA, especially for multimedia processing applications. This reconfigurable system has been designed for offering effective support to basic arithmetic functions, extensive bandwidth and memory resources is dispersed efficiently [18].

Becker et al., implemented a new architecture for coarse-grained dynamically reconfigurable hardware architecture called DReAM (Dynamically and Reconfigurable Hardware Architecture for Mobile Communication Systems) for wireless communication systems to achieve the required aspects for mobile communication [19].

CGRAs try to solve the flexibility and routing overhead by supplying complex operators and multiple-bit wide data paths at the bit-level configuration. In silicon, the wide datapath permits the implementation of complex operators efficiently. The complex operators from bit-level processing units are removed by generating the routing overhead [20]. FGRAs are more flexible than CGRAs [13], but CGRAs are easier to program, and reconfiguration is faster. CGRAs can be categorized based on the positioning of

Table 3 Different types of devices based on configuration

Configuration Technology	Coarse grained	Fine grained
SRAM-based FPGAs	Xilinx : Versal, Virtex, Kintex, ,Spartan; Intel Altera: Stratix, Arria, Cyclone; eASIC: NextremeSL; Lattice: EC, ECP; Achronix Speedster	ATMEL: AT6000, AT40K
Flash memory	Lattice XP, MACH XO	Actel: ProASIC3, Igloo, Fusion
Anti-fuse-based FPGAs	QuickLogic Eclipse II, PolarPro	Actel MX, Accelerator AX

processing elements such as Mesh-based [21, 22], linear arrays based [23, 24], Crossbar-based architectures [25], etc.

Partial Reconfiguration

Partial reconfiguration (PR) is defined as the modification of logic blocks by transferring partial bit files, whereas other parts of circuitry continue to run without any interruption. Instead of full reconfiguration, partial reconfiguration is required when configurations do not employ the entire reconfigurable device entirely, or only a part of it needs alteration. There may be different configurations performed in the hardware areas when configurations do not utilize the entire area accessible inside the range. PR can be partitioned into static and dynamic depending upon their functionalities. In static PR, FPGA becomes active only during the process of reconfiguration. Although some of the data is redeployed to the device, remaining part of the device is inactive and becomes active when the configuration is finished. Dynamic (active) partial reconfiguration which enables the portion of the device to be modified or configured when the remaining parts of FPGA are still operating.

Partial reconfiguration can be performed through either an internal host residing in the core logic, or as an external host via dedicated device pins. The advantage of the internal host is that it stores all the logic needed to perform PR on the device, without the need for external devices [26].

Partial reconfigurable hardware architectures can support for increasing entire reconfiguration flexibility, reducing power consumption and efficient utilization of FPGA and these architectures are employed in Xilinx Virtex families (Kintex-7, Artix-7, Virtex-4 to Virtex-7) and the SoC family(Zynq^{TM}-7000) [27, 28]. PR can minimize the amount of data configuration which should be loaded into reconfigurable hardware [29].

Xilinx initially proposed two categories of PR are module-based and difference-based methodologies [30, 31] followed by the Early Access Partial Reconfiguration (EAPR) flow. Module-based partial reconfiguration allows reconfiguring different modules at the time of design specification. Difference-based partial reconfiguration can perform only a few changes to the design parameters. The EAPR flow is an approach which permits signals to cross over the boundaries of the partially reconfigurable area and maintains 2D reconfigurable rectangular module shapes. Since there are no constraints in EAPR flow, it helps in fixing the problems raised in the modular design methodology. The main goal of EAPR flow is some parts of the FPGA remains stationary, while remaining parts are dynamically reconfigurable during execution. This flow uses Bus macros (BMs) to ensure a connection can be done properly between static and dynamic reconfigurable modules during the process of reconfiguration [32]. The Internal Reconfiguration Access Port (ICAP) is a configuration port which allows accessing the FPGA configuration memory at the time of execution [33]. The ICAP is available in almost every Xilinx FPGAs from basic to advanced device architectures [34]. In Virtex-II series, the ICAP provides an 8-bit I/O data buses with the input clock. ICAP configuration is upgraded in Virtex-4 Series with high bandwidth (32-bit I/O data bus) [35]. An embedded soft-core processor such as PowerPC® or MicroBlaze® can be implemented in the fabric of FPGAs to control dynamically reconfigurable system [36]. ICAP provides better solution for user-defined applications.

Virtex devices support dynamic reconfiguration without any interruption, when a configuration bit holds the same value before and after reconfiguration, there is no disruption in operation from the resource controlled by that bit without the inclusion of LUT RAMs and Shift Registers (SRL16). This constraint was removed in the Virtex-II/Pro FPGAs and Virtex-4 devices by introducing EAPR flow tools.

Z.Xiao, et al., explained the new Altera partial configuration flow and identified the problem of a wide range of sizes for partial reconfiguration bitstream [37].

Vipin et al., addressed the major role of partial reconfiguration in Xilinx Zynq which comes under hybrid FPGA platforms and developed an open source controller called ZyCAP provide more efficient use of hardware resources in these architectures [38].

PR can bring quite a few benefits for designing FPGA hardware. Moreover, it enables a large implementation to be embedded on a small chip by increasing the effective logic density of the chip. When compared with full reconfiguration, PR has a distinct advantage of reduced reconfiguration time concerning the size of the configuration module, which corresponds to the chip area, which is to be reconfigured. Partial Reconfiguration architecture is suitable for adaptive hardware systems because of their easy adaptation to varying conditions when the data processing is done [39].

Partial Reconfiguration Design and Implementation

The two main techniques of PR are difference based and Module based [32]. The difference-based technique is achieved by creating a change to design and generate a bitstream from the variations of the designs. This method performs the switching of module configuration from one implementation to another quickly, since the differences between the bitstream are not significantly larger than the

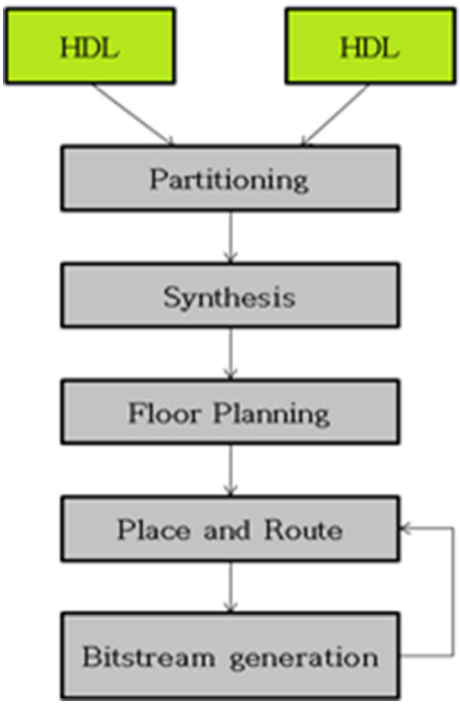

Fig. 6 Xilinx Partial Reconfiguration Flow [1]

whole device bitstream. This technique shall not be followed, as the changes to the design are large, and the signal integrity is not assured on reconfigurable modules boundaries.

Module-based design is a Xilinx Development System Option which works autonomously by merging into unique FPGA design from different modules [39]. The design of each module is at the top level design, which is considered as an independent module, and the parallel development permits modification of a module while the other modules are unaffected. Module-based PR involves in executing a series of procedures at the stage of design specification [40]. Then bit-stream is created individually for each reconfigurable module of the design. These reconfigurable modules have been represented in VHDL, and they are implemented on the FPGA board of Xilinx Virtex families with partial reconfiguration [41–43]. Xilinx PlanAhead Partial reconfiguration flow is shown in Fig. 6. Partial reconfiguration saves the silicon area, which enables multiple configurations to be interchanged and provide flexibility to replace the configuration. Vivado® Design Suite software tools unlock the capability to reconfigure a section of a Xilinx FPGA device, whereas the remaining parts of the device remain functioning [44–49]. The Vivado® Design Suite suggests a new approach for ultra-high productivity with next-generation IP-based and C/C++ design [50–52]. The design implementation of floor planning and I/O planning of Xilinx Zynq XC7Z020 using Vivado Design Suite is shown in Figs. 7 and 8, respectively.

Applications

The programming of RC systems can be done by directly program the logic gates inside the FPGA. Since FPGAs are free from instruction fetching operations, the instructions are built into the FPGA data path itself. FPGAs and their reconfigurability find their path in different applications such as data streaming, wireless communication systems, cryptography, network security, multimedia, DSP applications, medical research and therapy, bioinformatics, consumer electronics, automotive industries, military applications, artificial intelligence, deep learning and so on. Some of the applications benefited are listed in Table 4.

Cao Liang et al., represented an efficient architecture called SMARTCELL for data streaming applications using Altera Stratix II EP2S30 device benchmark platform [53]. This architecture showed better performance and low power consumption of about 52% than configurable FPGAs. Scott Hauck et al., explored the configuration compression and developed an algorithm to reduce the amount of data during reconfiguration in Xilinx XC6200 [54]. Usamah Algemili investigated the performance of the reconfigurable hardware FPGA for processing data streams using Moore's voting algorithm and implemented on Xilinx Zynq® board [55].

FPGA have gained quick acceptance over the past decades in wireless communication. Xue Liu et al., implemented FPGA-based reconfigurable down converter for wide band applications [56]. The proposed architecture was implemented in Xilinx FPGA XC7K70T-2FBG676 (Kintex-7 FPGA family) and Xilinx Zynq®. Jordane Lorandal et al., proposed FPGA implementation to evaluate performance and power comparison of wireless communication base band systems [57].

Xilinx Zynq® SoC supports for the fields of wireless technology and its protocols. B.Drozdenko et al., analyzed a method for hardware and software co-designs for wireless transceivers using Xilinx Zynq-based reconfigurable platform [58]. C. Ebeling et al., developed an architecture called RaPiD—Reconfigurable Pipelined Datapath Architecture [59] in 1996. Later T.H.Pham et al., implemented an Orthogonal frequency-division multiplexing (OFDM) receiver in the reconfigurable architecture [60]. OFDM baseband design for FPGA-based cognitive radios with partial reconfiguration is proposed by them.

Being an integral part of embedded systems, FPGA provides security and mechanisms for privacy. Multiple cryptographic algorithms can be implemented on Xilinx Zynq 7000 series FPGA [61]. The implemented design showed high performance, better resource utilization and system flexibility. Huiyun Li et al., demonstrated an experiment of ionic attack into SRAM-based cryptographic

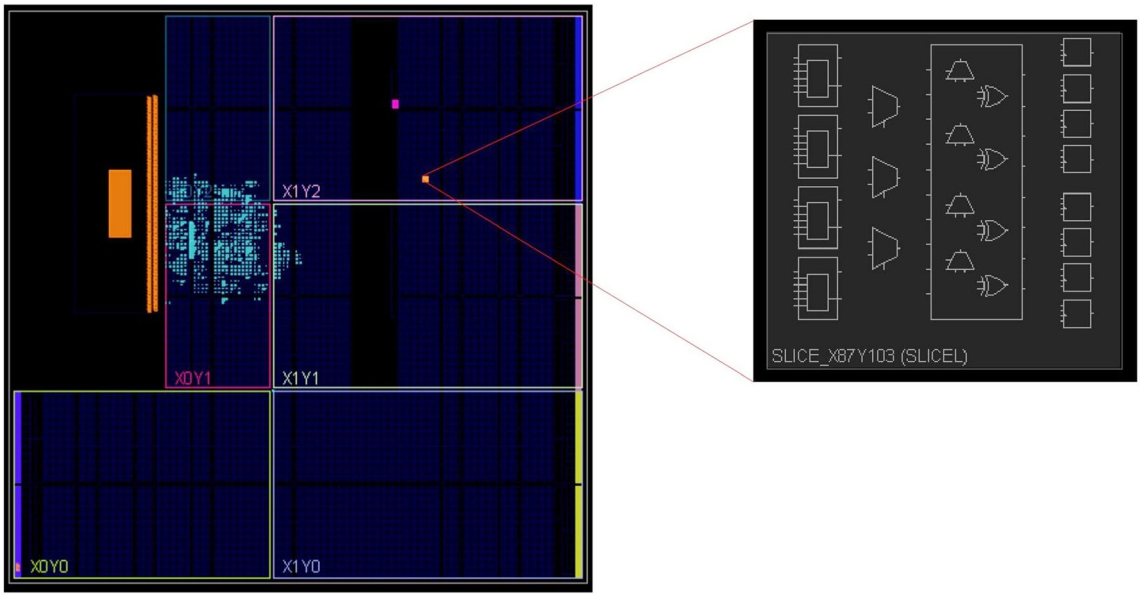

Fig. 7 Floor Planning of Xilinx Zynq XC7Z020 device and internal view of a slice

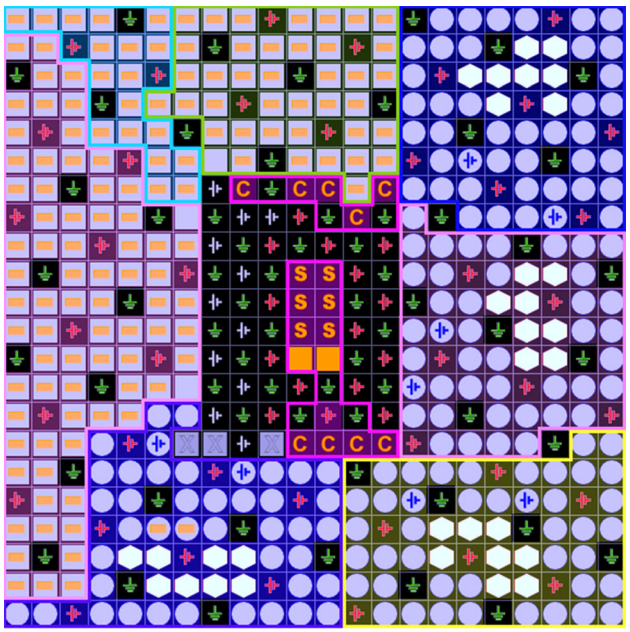

Fig. 8 I/O Planning of Xilinx Zynq XC7Z020 device

integrated circuits [62]. The clustering of FPGA resources made an impact in cryptographic computing and implementing on Altera (DE2-115) development board [63].

FPGAs support the demands of hardware implementations in network protocol and provide intense security. A. Chattopadhyay et al., proposed a protocol to transmit videos from one place to other captured in real time and implemented on Xilinx Zynq XC7Z020 series [64]. FPGA implementations in network infrastructure security and other major issues in hardware techniques were discussed in [65]. [66] developed an efficient and secured wireless local area networks (WLANs) based on FPGA implementation for multimedia applications.

Mishra et al., developed a solution for FPGA-based reconfigurable systems and also presented Reconfiguration Over Network (ReON) protocol with the help of Xilinx Internal Configuration Access Port (ICAP) [67]. The integration of software-defined networking (SDN) and reconfigurable network platforms are described in [68].

FPGA hardware architectures deliver the capability of performing high-quality image and video processing applications by providing a trade-off between resource utilization and performance. Bin Zhang et al., presented a reconfigurable processor implemented on Intel Stratix II EP2S180 Series FPGA for binary image processing [69]. Mahmoud Meribout et al., suggested a non-invasive FPGA-based THz imaging system for multiphase flow measurement and imaging implemented on Altera Stratix V FPGA [70].

Digital Signal Processors lack the flexibility in their architectures which reduce the performance and increase the complexity, whereas FPGAs provide flexibility and parallelism. A. Lindoso et al., implemented fault tolerant LEON3 soft-core processor in Artix-7 FPGA [71]. Javier Hormigo et al., investigated and then experimented floating point implementation in FPGA for DSP applications [72]. Bajaj Ronak et al., demonstrated that the Xilinx FPGAs provide the multi-pumping flexibility in DSP blocks by reducing the resources [73]. Kentaro Sano and Satoru Yamamoto demonstrated the scalability of FPGAs using floating point DSP blocks and compared among FPGAs and GPUs [74].

Table 4 Constructive table for different Applications

Applications	References	Device	Frequency	Power consumption	Energy	Area utilized	Processing time	Implications
Data streaming	[53]	Stratix II EP2S30	123 MHz	158 mW	28.5 GOPS/W	14% of chip area	–	Better performance and low power consumption than FPGA and RaPiD architectures [59]
	[54]	XilinxXC6200	–	–	–	–	–	Reconfigurable data compression, the algorithm achieved compression ratio of 4 (approx)
	[55]	Zynq 7000	–	–	–	33% of LUTs	–	Implemented Moore's voting algorithm for reconfiguration
Wireless communication	[56]	Kintex XC7K70T	452.4MHz/ 3.6 GHz bandwidth	2.293 W	–	–	–	Reconfigurable digital down converter (DDC) is implemented at sampling rate of 1 kS/s–225 MS/s
	[57]	Virtex-6	50MHz	58.19mW	–	–	–	Reduced power consumption
	[58]	Kintex-7 Artix-7	–	1.8 W	–	5% for ZC706 and 20% on Zedboard	–	Direct feedthrough algorithms are implemented
	[60]	Virtex- 6	10MHz	–	–	–	–	The approach showed the reduced storage requirements to 25% with that of other PR approaches
Cryptography	[62]	Cyclone IV	–	–	–	–	28 ms for decryption	Ion irradiation fault injection on cryptographic integrated circuits with 80.5-MeV/n carbon ions
Network security	[64]	Zynq XC7Z020	–	–	–	5% of total area	–	Authenticated Encryption with Associated Data (AEAD) is implemented for 640x480 resolutions with 30 fps
	[66]	Altera FPGA	–	82.4 mW	233 nJ	–	20–40 μs	The objective is to degrade the processing delay and improve the system throughput
Software Defined Networking (SDN)	[67]	Zynq	156.25 MHz	–	–	2% LUTs 1% FFs 8% BRAMs	128 nsec	PR protocol for higher layer networks
Image and video processing	[69]	Intel Stratix II	220 MHz	98.5 mW	60.72 GOPS and 23.72 GOPS/mm^2	2.56 mm^2	–	Mathematical morphology at 200fps for 1024 $\times$ 1024 resolution
	[70]	Intel Stratix V	80–110 GHz	10 mW	–	–	30 fps	Overall accuracy is 98.65% for 64 $\times$ 64 THz image for or high GVF multiphase flow
Digital signal processing	[73]	Virtex 6	242 MHz	–	–	–	48% DSP blocks, 74% LUTs	Dynamic reconfigurable multipliers and DSP blocks is implemented for multi-pumping operations
	[74]	Stratix10	225 MHz	–	–	–	–	Introduced Spatial and temporal parallelism for FPGAs increasing stream processing elements (SPEs)

Table 4 continued

Applications	References	Device	Frequency	Power consumption	Energy	Area utilized	Processing time	Implications
Medical	[75]	Virtex-5 LX50	100 kHz	–	–	–	100 fps	Implemented FFT algorithm on FPGA for EIT system with 100 fps
	[76]	Virtex 7	350 MHz	27.135 mW	–	–	70 fps	Gabor filter is implemented on FPGA to meet space constraints and reduce power consumption
	[77]	Virtex-5 FPGA	347.92 MHz	1689.84 mW	–	30.06%	–	Dynamic PR for 3D HWT is implemented on limited resources
Consumer Electronics	[80]	XilinxXC4VLX25	–	–	–	–	19 ms	IMDCT and antialias blocks are implemented
	[81]	ATMEL AT40K40	12 MHz	–	–	57% of CLBs	26ms	ARDOISE architecture is designed for DPR
Automotive	[83]	Virtex-4 FX60	–	–	–	–	15 ms	DPR for multiple target tracking (MTT) was implemented using Kalman filters to reduce resource utilization and power consumption
	[84]	Virtex II Pro	106.724 MHz	–	–	–	9.370 ns	Partial run-time reconfiguration is introduced for driver assistance systems based on pixel-level operation
Bioinformatics	[87]	Cyclone V	50 MHz	–	–	–	–	Implementation of multifractal processor achieved 2.6x speed for complete DNA sequence
	[89]	XC4VSX35 FPGA	–	–	–	1474 CLB slices	24.12 ms	SVM and KNN classifiers are implemented on FPGA for cancer diagnosis.
Defense and space grade	[92]	Virtex-6 (ML605)	2.5 GBytes/s	400 mW	–	1474 LUTs, 3 DSPs, 101 BRAMs (for MicroBlaze)	–	The method is to implement fault tolerant equipment on reconfigurable FPGA with reduced cost and better area utilization
Deep Learning	[99]	ZYNQ XC7Z020	–	6 W	–	–	289.65 ms	YOLO application is used for detecting various objects and recognized using Movidius USB-GPU

FPGAs provide variety of applications in the field of medical research. Shadab Khan et al., developed an algorithm and implemented on the FPGA for Electrical impedance tomography (EIT) [75]. G. D. Licciardo et al., used Gabor filter for medical imaging applications implemented on Xilinx FPGA hardware [76]. For diagnostics, monitoring and therapeutic applications, Virtex FPGA and Spartan® FPGA families could employ for satisfying a variety of processing and interfacing demands. The combinations of three-dimensional (3D) medical imaging and partial reconfiguration have been evolving in recent years. A. Ahmad et al., proposed two architectures based on transpose computation and partial reconfiguration and implemented three-dimensional (3D) Haar wavelet transform (HWT) on Xilinx Virtex-5 FPGAs [77].

Over the past few years, the demand for consumer electronics is increasing exponentially [78]. The recent trends in the consumer electronics market are in extreme uprising period in which FPGAs deserve a major role in the industry [79]. Some of the results have shown that PR could be handy in audio and video processing applications, such as MP3 decoding [80] and JPEG encoding [81]. [82] proposed a new architecture for content security encryption and decryption that is dynamically configured on Xilinx FPGA (XC2VP100).

FPGAs are capable of performing multi-threading operations which allow them to implement in various applications, including automotive industries. Modern vehicles incorporate hundreds of electronic control units which directly influence their power consumption and durability. Hence, reconfigurable hardware components come across their pathway in addressing these challenges and provide flexibility. Some of the researchers have demonstrated the development of PR in driver assistance systems [83]. The real-time video processing based on PR on FPGAs is presented [84], and Autovision architecture for driver assistance systems was developed by [85]. Shanker Shreejith et al., explained the necessity of reconfigurable computing in the automotive industries for the next generation [86].

FPGA computing has removed all the obstacles to make their way in bioinformatics. J.E. Duarte-Sánchez et al., designed a low-cost SoC-FPGA to process human genome and also implemented a hardware accelerator for multifractal analysis of DNA sequences [87]. G Chrysos et al., surveyed the applications and challenges of reconfigurable

computing in the platform of bioinformatics [88]. [89] proposed a dynamic multi-classifier architecture of implementing dynamic partial reconfiguration combined with FPGAs that can be used in processing bioinformatics data .

[90] studied and explored the use of reconfiguration in military applications. FPGA will also find their use in space-based applications such as fault recovery and design of avionic equipment [91–94] proposed fine-grained dynamic reconfiguration method that could perform detection and recovering from configuration errors in SRAM FPGA-based Triple Modular Redundancy (TMR) devices for high radiation environment. Xilinx Microblaze TMR solution using Vivado Design Suite provides error detection and recovery for Xilinx devices [95]. PR has also been proposed in mitigating Single event upsets (SEUs) in SRAM-based FPGAs [96].

The need for technology improvements is more than before. With this acceleration, especially deep learning is yielding successful results along with FPGAs in the fields of object detection, voice recognition, medical analysis and so on. E.Wang et.al., [97] presented the first survey article for customized hardware accelerators and compared them with approximation for CNN and RNN (Recurrent Neural Networks). In [98], the authors have explained FPGA accelerators for convolution neural networks (CNNs) in order to increase the overall performance. Reconfigurable hardware accelerators can also be a different approach for implementing deep learning techniques. For object detection using CNN implemented on PYNQ board is presented [99]. In this study, Movidius USB-GPU is also experimented with PYNQ along with YOLO application is used for detecting various objects such as human, bicycle and a car. The experimental results in terms of calculation time and number of operations per image have shown that the performance of reconfigurable FPGA hardware is suitable for deep learning optimization.

Recent Developments

AMAZON and Microsoft have launched FPGA cloud services [100]. The Asia-Pacific region seized the majority of market value in the global FPGA market and is expected to be the fastest increasing market during the estimated period. Growing applications in defense and aerospace, artificial intelligence, consumer electronics, deep learning and the automotive industry are a vital factor supporting their rapid growth in newer aspects and are presumed to steer the industries over the upcoming years. Intel® introduced 10 nm technology based Intel® Agilent® FPGAs and SOCs to deliver high performance for applications such as data centers, networking, and edge computing [101]. Xilinx team up with Samsung to deploy world's first 5G New Radio (NR) commercial deployment using the Xilinx® UltraScale+ platform [102].

Summary

Reconfigurable computing has been progressed significantly over the years. A concise study of FPGA architectures has been carried out and analyzes the architectures of reconfigurable devices and their applications. This survey will pave the way for researchers to find suitable applications for FPGAs. Some of the potential application domains such as automotive, communication, aeronautical, defense, deep learning and consumer electronics provide well suited to support PR system design. Reconfigurable architectures have been developed from basic FPGA architecture to high-performance architectures. The major issue in coarse-grained reconfigurable architectures is its flexibility need to be addressed for reconfigurable processors in the future. Among the types of FPGAs, SRAM based are widely used because of their volatility and reprogrammability. Programming for reconfigurable architectures and complete virtualization of FPGA resources for PR are the challenging tasks. The simulation of floor planning and I/O planning of PR device like Zynq® XC7Z020 is displayed with the help of Vivado® Design Suite. The various applications of PR are explored with a wide range of domains. The reconfigurable computing will find its way in the fields of nanotechnology shortly.

References

1. S.A. Fahmy, K. Vipin, FPGA dynamic and partial reconfiguration: a survey of architectures, methods, and applications. Comput. Surveys **51**, 1–39 (2018)
2. Xilinx, *FPGA*, 2019. Retrieved January 14, 2019 from https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html
3. G. Singh, Reconfigurable computing: a review of the technology and its architecture. IOSR J. VLSI Signal Process. IOSR-JVSP **3**, 8–13 (2013)
4. I. Kuon, J. Rose, Measuring the gap between FPGAs and ASICs. Trans. Compu. Aided Des. Integr. Circuits Syst. TCAD **26**(2), 203–215 (2007)
5. N. Alaraje, J.E. DeGroat, Evolution of reconfigurable architectures to SoFPGA, in *48th Midwest Symposium on Circuits and Systems*. IEEE, pp. 818–821 (2005)
6. C. Bobda, *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications* (Kluwer Academic Publishers, Dordrecht, 2007)
7. S. Brown, J. Rose, Architecture of FPGAs and CPLDs: a tutorial. IOSR J. VLSI Signal Process. IOSR-JVSP **13**(2), 42–57 (1996)
8. U. Farooq, *Tree-Based Heterogeneous FPGA Architectures* (Springer, Berlin, 2012)

9. Microsemi Corp Tim Morin. Flash FPGAs give designers more flexibility. Retrieved January 14, 2019 from https://www.embedded.com/electronics-blogs/industry-comment/4438457/Flash-FPGAs-give-designers-more-flexibility (2019)
10. K. WeiB, C. Oetker, I. Katchan, T. Steckstor, W. Rosenstiel, Power estimation approach for SRAM-based FPGAs, in *Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays*. IEEE, Monterey, CA, USA (2000)
11. Ali Azarian, Mahmood Ahmadi, Reconfigurable Computing Architecture : Survey and introduction. In 2nd International Conference on Computer Science and Information Technology . IEEE, Beijing, China, pp.269–27. (2009)
12. W. Zhang, T.-J. Lin, N.K. Jha, A fine-grain dynamically reconfigurable architecture aimed at reducing the FPGA-ASIC Gaps. Trans. Very Large Scale Integr. VLSI Syst. **22**(12), 2607–2620 (2014)
13. M. Jain, M.P. Singh, A survey of reconfigurable architectures. Int. J. Comput. Appl. **98**(14), 36–40 (2014)
14. A. DeHon, R. Tessier, K. Pocek, Reconfigurable computing architectures, in *Proceedings of the IEEE*. IEEE, USA, pp. 332–354 (2015)
15. D. Soudris, G. Theodoridis, S. Vassiliadis, *Basic Definitions, Critical Design Issues and Existing Coarse-grain Reconfigurable Systems in A Survey of Coarse-Grain Reconfigurable Architectures and Cad Tools* (Springer, Berlin, 2007), pp. 89–149
16. A. De Hon. Reconfigurable Accelerators. Technical Report 1586. MIT Artificial Intelligence Laboratory (1996)
17. V. Tehre, R. Kshirsagar, Survey on coarse grained reconfigurable architectures. Int. J. Comput. Appl. **48**, 16 (2012)
18. P. Corsonello-Martin, M.M. Lanuzza, S. Perri, A new reconfigurable coarse-grain architecture for multimedia applications, in *Proceedings of the Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)* . IEEE, pp. 119–126 (2007)
19. C. Habermann-Manfred, G.J. Becker, T. Pionteck, Design and implementation of a coarse-grained dynamically reconfigurable hardware architecture, in *Proceeding WVLSI 01 Proceedings of the IEEE Computer Society Workshop on VLSI*. IEEE, pp. 41–46 (2001)
20. R. Tessier, I. Kuon, J. Rose, FPGA architecture: survey and challenges. Found. Trends Electron. Des. Autom. **2**(2), 135–253 (2008)
21. D. Cherepacha, D. Lewis, DP-FPGA: an FPGA architecture optimized for datapaths. Proc. FPGA **1994**, 329–343 (1994)
22. R. Kress, R.W. Hartenstein, A datapath synthesis system for the reconfigurable datapath architecture, in *Proceeding of ASP-DAC95*. pp. 329–343 (1995)
23. D.C. Cronquist, C. Ebeling, P. Franklin, RaPiD: reconfigurable pipelined datapath architecture, in *International Workshop on Field Programmable Logic and Applications, Field-Programmable Logic Smart Applications, New Paradigms and Compilers*. Springer, pp. 126–135 (1996)
24. M. Moe, M. Budiu, S. Cadambi, R.R. Taylor, S.C. Goldstein, H. Schmit, R. Laufer, PipeRench: a coprocessor for streaming multimedia acceleration, in *Proceedings of the 26th International Symposium on Computer Architecture* (1999)
25. D. Chen, J. Rabaey, A reconfigurable multiprocessor IC for rapid prototyping of algorithmic-specific high-speed DSP data paths. J. Solid-State Circuits **27**(12), 1895–1904 (1992)
26. Altera. Partial Reconfiguration IP Core, UG-PARTRECON datasheet. Retrieved January 14, 2019 from https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-partrecon.pdf (2017)
27. Xilinx Inc. 2011b. Virtex-II Pro and Virtex-II Pro-X Platform FPGAs, DS083 (v5.0).Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/data_sheets/ds083.pdf
28. Xilinx Inc. 2015a.UG360: Virtex 6 FPGA Configuration user guide, UG360 (v3.9).Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/user_guides/ug360.pdf
29. K. Compton, S. Hauck, Reconfigurable computing: a survey of systems and software. Comput. Surveys **34**(2), 171–210 (2002)
30. Xilinx Inc. 2003. Two flows for partial reconfiguration: module based or difference based, Xilinx Application Note XAPP290, Version 1.1. Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/application_notes/xapp290.pdf
31. Xilinx Inc. 2004b. Two flows for partial reconfiguration: module based or difference based, Xilinx Application Note XAPP290, Version 1.2. Retrieved January 14, 2019 from https://pdfhall.com/xilinx-xapp290-two-flows-for-partial-xun-zhang-page_5bd91d8a097c47647d8b4574.html (2004)
32. Xilinx Inc. 2011a. Partial Reconfiguration User Guide, UG702 (v13.1). Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/ug702.pdf (2011)
33. S. McMillan, B. Blodget, P. Lysaght, A lightweight approach for embedded reconfiguration of FPGAs. Proc. Conf. Des. Autom. Test Europe **1**, 399–400 (2003)
34. S. Bayar, A. Yurdakul, Dynamic partial self reconfiguration on Spartan-III FPGAs via a parallel configuration access port (PCAP), in *Proceedings of the 2nd HiPEAC Workshop on Reconfigurable Computing*, pp. 1–10 (2008)
35. J. Mason, J. Young, P. Lysaght, B. Blodget, B. Bridgford, Enhanced architectures, design methodologies and CAD tools for dynamic reconfiguration of Xilinx FPGAS, in *Proceedings of the International Conference on Field Programmable Logic and Applications*, pp. 1–6 (2006)
36. Xilinx Inc. 2017c. Vivado Design Suite Tutorial, UG947 (v2017.3). Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug947-vivado-partial-reconfiguration-tutorial.pdf
37. D. Koch, Z. Xiao, M. Lujan, A partial reconfiguration controller for Altera Stratix V FPGAs, in *Proceedings of the International Conference on Field Programmable Logic and Applications* (2016)
38. S.A. Fahmy, K. Vipin, ZyCAP: efficient partial reconfiguration management on the Xilinx Zynq. IEEE Embedded Syst. Lett. **6**(3), 41–44 (2014)
39. K. Rajesham, M. Majer, A. Niyonkuru, C. Bobda, A. Ahmadinia, Partial configuration design and implementation challenges on Xilinx Virtex FPGAs, in *Proceedings of 18th International Conference on Architecture of Computing Systems*, pp. 61–66 (2005)
40. Xilinx Inc. Programmable Logic Data Book, 1996. Retrieved January 14, 2019 fromhttp://noel.feld.cvut.cz/hw/xilinx/Xilinx96.pdf
41. Xilinx Inc.2014b. Virtex-II Platform FPGAs, DS031 (v4.0). Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/data_sheets/ds031.pdf
42. Xilinx Inc. 2004c.Virtex Series Configuration Architecture User Guide, XAPP151 (v1.7). Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/application_notes/xapp151.pdf
43. Xilinx Inc. 2007b. Xilinx Device Drivers Programmer Guide, (v1.4), Retrieved January 14, 2019 from https://xilinx.github.io/embeddedsw.github.io/doc/xilinx_drivers_guide.pdf (2007)

44. Xilinx Inc. 2004a.DS280: OPB HWICAP, DS 280 (v1.3). Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/ip_documentation/opb_hwicap.pdf (2004)
45. Xilinx Inc.UG070:Virtex-4 FPGA User Guide, UG070 (v2.6). Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/user_guides/ug070.pdf (2008)
46. Xilinx Inc. 2010. DS586: XPS HWICAP, LogiCORE IP XPS HWICAP(v5.00a). Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/ip_documentation/xps_hwicap.pdf
47. Xilinx Inc. 2007a. Virtex-II Pro and Virtex-II Pro-X FPGA User guide, UG012(v4.2), Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/user_guides/ug012.pdf
48. Xilinx Inc. 2015b. UG360: Virtex 6 FPGA Configuration User Guide,UG360 (v3.9).Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/user_guides/ug360.pdf
49. Xilinx Inc. 2014a. UG910:Vivado Design Suite User Guide, UG910 (v2014.1).Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug910-vivado-getting-started.pdf
50. Xilinx Inc. 2012. UG893:Vivado Design Suite User Guide, UG893 (v2012.2).Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug893-vivado-ide.pdf
51. Xilinx Inc. 2017a.UG909:Vivado Design Suite User Guide Partial Reconfiguration, UG909 (v2017.1). Retrieved Jan-uary 14, 2019 from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug909-vivado-partial-reconfiguration.pdf
52. Xilinx Inc. 2017b. Vivado Design Suite HLx Editions User Guide, UG893 (v2017.3).Retrieved January 14, 2019 from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug893-vivado-ide.pdf
53. C. Liang, X. Huang, SMARTCELL: a power-efficient reconfigurable architecture for data streaming applications, in *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 257–262 (2008)
54. Z. Li, S. Hauck, E. Schwabe, Configuration compression for the Xilinx XC6200 FPGA. Trans. Comput.-Aided Des. Integr. Circuits Syst. **18**(8), 1107–1112 (2008)
55. U. Algemili, Investigation of reconfigurable FPGA design for processing big data streams, in *2nd International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing; IEEE International Conference on Intelligent Data and Security (IDS)* (2016)
56. Z.-K. Wang, X. Liu, X.-X. Yan, Q.-X. Deng, Design and FPGA implementation of a reconfigurable digital down converter for wideband applications. Trans. Very Large Scale Integr. VLSI Syst. **25**(12), 3548–3552 (2017)
57. J.-C. Prvotet, J. Lorandel, M. Helard, Fast power and performance evaluation of FPGA-based wireless communication systems. IEEE Access **4**, 2005–2018 (2016)
58. T. Dao, K. Chowdhury, M. Leeser, B. Drozdenko, M. Zimmermann, Hardware-software codesign of wireless transceivers on Zynq heterogeneous systems. Trans. Emerg. Topics Comput. **6**(4), 566–578 (2018)
59. G. Xing, M. Shen, C. Ebeling, C. Fisher, H. Liu, Implementing an OFDM receiver on the RaPiD reconfigurable architecture. Trans. Comput. **53**(11), 1436–1448 (2004)
60. S.A. Fahmy, T.H. Pham, I.V. Mcloughlin, An end-to-end multi-standard OFDM transceiver architecture using FPGA partial reconfiguration. IEEE Access **5**, 21002–21015 (2017)
61. X. Chen-Xiaojun, T.Y. Yao, Z. Wang, Q. Luo, A dynamic reconfigurable design of multiple cryptographic algorithms based on FPGA, in *International Conference on Smart Internet of Things*. IEEE, pp. 105–110 (2018)
62. C. Shao, L. Dai, X. Guoqing, H. Li, D. Guanghua, J. Guo, Heavy-ion microbeam fault injection into SRAM-based FPGA implementations of cryptographic circuits. Trans. Nucl. Sci. **62**(3), 1341–1348 (2015)
63. D. Hulton, Accelerating cryptography with FPGA clusters. Retrieved April 10, 2019 from http://mil-embedded.com/articles/accelerating-cryptography-fpga-clusters (2010)
64. A. Baksi, A. Chattopadhyay, V. Pudi, T. Srikantha, FPGA based cyber security protocol for automated traffic monitoring systems: proposal and implementation, in *Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, pp. 18–23 (2016)
65. D.H. Summerville, H. Chen, Y. Chen, A survey on the application of FPGAs for network infrastructure security. Commun. Surveys Tutor. **13**(4), 541–561 (2011)
66. B.J. Mohd, T. Hayajneh, S. Ullah, K.S. Balagani, An enhanced WLAN security system with FPGA implementation for multimedia applications. IEEE Syst. J. **11**(4), 2536–3545 (2017)
67. G. Zervas, V. Mishra, Q. Chen, REoN: a protocol for reliable software-defined FPGA partial reconfiguration over network, in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. IEEE (2016)
68. C. Rotsos, N. Zilberman, P.M. Watts, A.W. Moore, Reconfigurable network systems and software-defined networking, in *IEEE Proceedings*, vol. 103. IEEE (2015)
69. K. Mei, B. Zhang, N. Zheng, Reconfigurable processor for binary image processing. Trans. Circuits Syst. Video Technol. **25**(3), 823–831 (2013)
70. I.M. Saied, M. Meribout, E. Alosani, A new FPGA-based terahertz imaging device for multiphasecircuits and systems II: express briefs. IEEE Trans. Terahertz Sci. Technol. **64**(3), 319–323 (2017)
71. M. Garca-Valderas, A. Lindoso, L. Entrena, L. Parra, A hybrid fault-tolerant LEON3 soft core processor implemented in low-end SRAM FPGA. Trans. Nucl. Sci. **64**, 1 (2017)
72. J. Hormigo, J. Villalba, HUB floating point for improving FPGA implementations of DSP applications. Trans. Circuits Syst. II Express Briefs **64**(3), 319–323 (2017)
73. B. Ronak, S.A. Fahmy, Multipumping flexible DSP blocks for resource reduction on Xilinx FPGAs. Trans. Comput.-Aided Des. Integr. Circuits Syst. **36**(9), 1474–1482 (2017)
74. K. Sano, S. Yamamoto, FPGA-based scalable and power-efficient fluid simulation using floating-point DSP blocks. Trans. Parallel Distrib. Syst. **28**(10), 2823–2837 (2017)
75. A. Borsic, S. Khan, P. Manwaring, R.J. Halter, FPGA-based voltage and current dual drive system for high frame rate electrical impedance tomography. Trans. Med. Imaging **34**(4), 888–901 (2015)
76. C. Cappetta, G.D. Licciardo, L.D. Benedetto, Design of a gabor filter HW accelerator for applications in medical imaging. Trans. Compon. Packag. Manuf. Technol. 1187–1194 (2018)
77. A. Amira, H. Rabah, A. Ahmad, B. Krill, Efficient architectures for 3D HWT using dynamic partial reconfiguration. J. Syst. Architect. **56**, 305–316 (2010)
78. G. Roos, Consumer Electronics Drive FPGA Growth (2015), Retrieved April 10, 2019 fromhttps://epsnews.com/2015/07/15/consumer-electronics-drive-fpga-growth
79. Xilinx, 2010, *Xilinx in Consumer Electronics: Accelerating Differentiation and Innovation*, Retrieved April 16, 2019 from https://www.xilinx.com/publications/prod_mktg/CS527_Consumer_SellSheet.pdf
80. M.H. Zarifi, H. Taghipour, J. Frounchi, Design and implementation of MP3 decoder using partial dynamic reconfiguration on Virtex-4 FPGAs, in *Proceedings of the International Conference on Computer and Communication Engineering*. IEEE (2008)
81. E.B. Bourennane, S. Bouchoux, M. Paindavoine, Implementation of JPEG2000 arithmetic decoder using dynamic

reconfiguration of FPGA, in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design*, pp. 2841–2844 (2004)
82. H. Yokoyama, K. Toda, FPGA-based content protection system for embedded consumer electronics, in *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA.05)*. IEEE (2005)
83. M.A.R. Saghir, Y.E. Hillali, N. Harb, S. Niar, R.B. Atitallah, Dynamically reconfigurable architecture for a driver assistant system, in *Proceedings of the 9th Symposium on Application Specific Processors (SASP)*. IEEE, pp. 62–65 (2011)
84. F. Muller, C. Claus, J. Zeppenfeld, W. Stechele, Using partial-run-time reconfigurable hardware to accelerate video processing in driver assistance system, in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*. IEEE, pp. 1–6 (2007)
85. W. Stechele, C. Claus, A. Herkersdorf, Autovision: a run-time reconfigurable MPSoC architecture for future driver assistance systems. Inf. Technol. **49**(8), 181–186 (2007)
86. S.A. Fahmy, S. Shreejith, M. Lukasiewycz, Reconfigurable computing in next-generation automotive networks. Embed. Syst. Lett. **5**(1), 12–15 (2013)
87. P.A. Moreno, J.E. Duarte-Snchez, J. Velasco-Medina, Hardware accelerator for the multifractal analysis of DNA sequences. Trans. Compon. Packag. Manuf. Technol. **15**(5), 1611–1624 (2018)
88. C. Rousopoulos, K. Pramataris, I. Papaefstathiou, A. Dollas, A. Papadopoulos-I, V.J. Kirmitzoglou, T. Promponas, G. Theocharides, G. Petihakis, E. Chrysos, J.L. Sotiriades, Reconfiguring the bioinformatics computational spectrum: challenges and opportunities of FPGA-based bioinformatics acceleration platforms. Des. Test **31**(1), 62–73 (2014)
89. K. Benkrid, H.M. Hussain, H. Seker, Dynamic partial reconfiguration implementation of the SVM/KNN multi-classifier on FPGA for bioinformatics application, in *Proceedings of The 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, pp. 7667–7670 (2007)
90. L. Tosi, M.D. Ciano, O. Mulertt, Y. Gabriel, J.-D. Legat, D. Aulagnier, C. Gamrat, R. Liberati, P. Manet, D. Maufroid, V.L. Barba, RECOPS: reconfiguring programmable devices for military hardware electronics, in *Proceedings of 2007 Design, Automation Test in Europe Conference Exhibition*. IEEE, pp. 1–6 (2018)
91. A. Fernndez, N. Montealegre, D. Merodio, P. Armbruster, In-flight reconfigurable FPGA-based space systems, in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, pp. 1–8 (2015)
92. N. Belanger, R.B. Atitallah, V. Viswanathan, J.-L. Dekeyser, FPGA-centric design process for avionic simulation and test. Trans. Aerospace Electron. Syst. **54**(3), 1047–1065 (2018)
93. N. Marzwell, W.H. Zheng, S. Chau, In-system partial run-time reconfiguration for fault recovery applications on spacecrafts, in *International Conference on Systems, Man and Cybernetics*, Vol. 4. IEEE, pp. 3952–3957 (2005)
94. T.H. Nguyen, E. Cetin, Z. Zhao, D. Agiakatsikas, O. Diessel, Fine-grained module-based error recovery in FPGA-based TMR systems. Trans. Reconfig. Technol. Syst. **11**(1), 1047–1065 (2018)
95. Xilinx, 2008, XAPP987: Single-Event Upset Mitigation Selection Guide, Retrieved April 16, 2019 fromhttps://www.xilinx.com/support/documentation/application_notes/xapp987.pdf
96. Xilinx 2018. PG268: MicroBlaze Triple Modular Redundancy (TMR) Subsystem(v1.0). Retrieved April 16, 2019 from https://www.xilinx.com/support/documentation/ip_documentation/tmr/v1_0/pg268-tmr.pdf
97. E. Wang, J.J. Davis, R. Zhao, H.-C. Ng, X. Niu, W. Luk, P.Y.K. Cheung, G.A. Constantinides, Deep neural network approximation for custom hardware: where we've been, where we're going. ACM Comput. Surv. (2019). https://doi.org/10.1145/3309551
98. A. Shawahna, S.M. Sait, A. El-Maleh, FPGA-based accelerators of deep learning networks for learning and classification: a review. IEEE Access **7**, 7823–7859 (2019). https://doi.org/10.1109/ACCESS.2018.2890150
99. V.Y. Cambay, A. Uar, M.A. Arserim, Object detection on FPGAs and GPUs by using accelerated deep learning, in *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, Turkey, pp. 1–5 (2019). https://doi.org/10.1109/IDAP.2019.8875870
100. K. Freund, *Amazon And Xilinx Deliver New FPGA Solutions*, 2017. Retrieved April 16, 2019 fromhttps://www.forbes.com/sites/moorinsights/2017/09/27/amazon-and-xilinx-deliver-new-fpga-solutions/#751292392370
101. Intel, 2019, *Intel Agilex FPGA Advanced Information Brief: (Device Overview), AG overview*, Retrieved April 10, 2019 from https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/agilex/ag-overview.pdf
102. Xilinx, 2019, *Xilinx and Samsung Jointly Enable the Worlds First 5G NR Commercial Deployment*, Retrieved February 25, 2019 fromhttps://www.xilinx.com/news/press/2019/xilinx-and-samsung-jointly-enable-the-world-s-first-5g-nr-commercial-deployment.html