Materials Today: Proceedings xxx (xxxx) xxx

Contents lists available at ScienceDirect

Materials Today: Proceedings

journal homepage: www.elsevier.com/locate/matpr

Partial dynamic reconfiguration framework for FPGA: A survey with concepts, constraints and trends

T. Siva Sankar Phani [a], Anitha Arumalla [b], M. Durga Prakash [b,*]

[a] Department of ECE, Koneru Lakshmaiah Education Foundation, Guntur, Vaddeswaram 522502, India
[b] Department of ECE, V R Siddhartha Engineering College, Kanuru, Vijaywada 520007, India

ARTICLE INFO

Article history:
Received 11 December 2020
Received in revised form 22 January 2021
Accepted 26 January 2021
Available online xxxx

Keywords:
Dynamic reconfiguration
Reconfiguration controllers
Fine-grain reconfiguration

ABSTRACT

With demand for high performance and huge logic dense portable devices, the need for silicon area is increasing. A potential solution for the electronics industry to develop such huge logic demanding applications is the ability to reconfigure the system partially without altering the overall system operation. For more than two decades, reconfigurable computing has aided various applications and has seen tremendous technology transformation. The paper presents a survey of reconfigurable computing, its present state of existence, and a detailed report on state of art Partial Dynamic Reconfiguration Framework (PDRF) for reconfiguring FPGA designs partially and dynamically. A detailed analysis of the features, limitations, and performance of a wide range of PDRFs available in the literature are reported.

Selection and peer-review under responsibility of the scientific committee of the International Conference on Materials, Manufacturing and Mechanical Engineering for Sustainable Developments-2020.

1. Introduction

Reconfigurable computing was first introduced [1,2] during 1980 to accelerate the processing speed of General Purpose Processors (GPP). A GPP use generalized instruction set and hardware to execute a wide variety of algorithms with insignificant processing speed. Hence, a specialized Reconfigurable Hardware System (RHS) accelerator is complemented with GPP that can be customized by converting a part of the computation intensive code to hardware depending upon the task executed [3]. The RHS can be programmed to perform multiple tasks that can accelerate different portions of the software code running on a GPP. Depending on the code running on the GPP, one of multiple accelerators is programmed onto reconfigurable hardware. It usually takes an order of magnitude less time to evaluate a function when compared with the time required by assembly code executing on a GPP.

A RHS is divided into static region and dynamic region. Static region consists of primary functional unit of the system that doesn't need a reconfiguration and it continues to function even during reconfiguration. The region that can be reconfigured is Dynamic region / Dynamic Reconfiguration Region (DRR). Either static region or an external host computer will control the reconfiguration process of DRR. The RHS is classified as either Coarse Grain RHS (CGRHS) or Fine Grain RHS (FGRHS) depending on the granularity of reconfigurable datapath used in DRR. DRR with datapath greater than 1-bit granularity are Course Grain DDR (CGDRR) [4]. CGDDR is mostly used along with a GPP as a hardware accelerator for a given application. A CGDDR consists of Programmable Functional Units (PFU) such as ALUs, Multipliers, Look-Up-Tables, Shifters, digital macro units etc connected as linear [5,6], crossbar [7] or mesh arrays [8,9] along with programmable routing.

The arrangement of these PFU depend on the data transfer network, type of reconfiguration used and the application for which it is designed for. Mesh based arrays are more efficient than the other two architectures as its 2D array structure supports better routing with 8 PFUs surrounding each PFU for efficient configuration and parallelism. The CGDDR provide efficient reconfiguration due to higher granularity of datapath, regular structures, less congestion routing, less complex placement and small configuration memory requirement.

While the CGDDR is efficient for reconfiguration, it offers less flexibility in altering the RHS to suit wide range of applications. The architecture of CGRHS is fixed before fabrication, while majority of FGRHS use Field Programmable Gate Array (FPGA) that provides a flexibility to alter DRR after fabrication also. The reconfiguration of CGRHS computer systems is done by dynamic alteration of interconnects through software control, variation of level of parallelism by dynamic resource partition and dynamic

* Corresponding author.
E-mail address: mdprakash@vrsiddhartha.ac.in (M. Durga Prakash).

https://doi.org/10.1016/j.matpr.2021.01.851
2214-7853/
Selection and peer-review under responsibility of the scientific committee of the International Conference on Materials, Manufacturing and Mechanical Engineering for Sustainable Developments-2020.

adaptation of instruction set architecture to support large variety of applications like digital signal processing, multimedia, software radio, wireless communications, general purpose loop acceleration etc. As the CGDRRs are mostly reconfigurable data paths and routes with hardwired CFB, the flexibility of these architectures to get adapted to other applications is less. Research in CGRHS is mainly focused on standard and distributed CGRHS structures. A standard CGRHS structure has a GPP as the primary processing unit and multiple DRRs supporting GPP while a distributed CGRHS structure has multiple DRRs without a GPP providing a high level of resource regularity and configuration generality. Any one of the DRRs can be configured and used as a primary GPP. Each PFU of either structure is interfaced with a configuration memory that holds configuration information for each different configuration. The structure of standard CGRHS and distributed CGRHS are shown in Fig. 1.

The configuration mapping framework of a RHS is dependent on the architecture, granularity, level of interconnect and selection of compiler. Compiler should be designed for each different architectures. Higher the granularity less complicated is the mapping technology and routing algorithm.

Until the last decade, most of the designs were CGRHS owing to large routing overhead and huge resources required for routing configuration of FGRHS. However, multi-granular solutions are also developed to suit the application requirements. As the paper is mainly focused on the survey of research carried out on FGRHS using FPGA, the detailed discussion of CGRHS is beyond the scope of this paper. However few papers have contributed a detailed survey on CGRHS with substantial conclusions [4,10–12]. The rest of the paper is organized as follows: FGRHS introduction is presented in Section II, basic information required to perform partial reconfiguration is detailed in Section III, and research work reported for the development of PDRF is surveyed in Section IV followed by discussions and conclusion in subsequent sections.

2. Fine grain reconfigurable hardware systems

Fine grain reconfiguration is a solution for high degree of flexibility to user. FGRHS use 1-bit granular Configurable Logic Block (CLB) as data path in DRR which can achieve vast functional transformation. The CGDDR are custom build for targeted application and therefore cannot be reused to suit a different application. But most of the early research is concentrated on CGRA due to high configuration overheads, low logic density and less support available for generic fine grain architectures like FPGAs. Now-a-days, leading FPGA vendors like Xilinx, Altera etc., are manufacturing high performance FPGAs at 16 and 20 nm technology nodes respectively. These FPGAs are provided with high end user friendly resources like hardwired embedded processor, memories, DSP block sets, and dynamic reconfiguration ports. As Xilinx is most widely used in reconfiguration computing, all the further discussions will be based on Xilinx FPGAs.

The FGRHS architecture consists of either external or on-chip DRRs as shown in Fig. 2. External FGRHS is built with multiple FPGAs and a controller [13,14]. The controller is either a GPP or another FPGA or CPLD, controlling the reconfiguration process. Dynamic reconfiguration is used to reconfigure one FPGA at a time in external FGRHS. No specialized tools are required to generate configuration information for external FGRHS as entire FPGA will be reconfigured. However, a reconfiguration controller should be designed for scheduling and reconfiguring the application.

Eldredge and Hutchings, [15] have proposed a Run-time Reconfigurable Artificial Neural Network (RRANN) that can be reconfigured at run-time and can enhance the FPGA's functional logic density. Three stages of back propagation algorithm that execute sequentially are reconfigured on common resources depending on the requirement. The system is developed on X12 board containing 12 XC3000 FPGAs with one FPGA used for global controller that interface with PC and control the reconfiguration process. Four FPGAs are used to implement neural processor for each stage of back propagation algorithm. The methodology has reported 500% area/resource saving for the implementation of neurons at the cost of performance. The system executed reconfiguration process for 80% of the time, while computation of results is done only for 20% of the total execution time. To decrease reconfiguration time of RRANN, [16] a careful physical partitioning of static and dynamic regions is performed. A reduction of resource aware RR has increased the neuron occupancy by 50% and reduced the reconfiguration time by 25% that of RRANN.

In [13], a virtual hardware manager is developed to interface between the run time system and computer application. The hardware manager is used to generate netlist of the reconfigurable circuits, place and route in the available FPGA resources, transform the circuit to new location on FPGA in case of relocation and can handle application interrupts. It is tested to accelerate postscript application running on PC and to reconfigure systolic FIR structure on XC6200 FPGA.

A reconfigurable development platform [14] was developed to support reconfigurable systems development on XC6200 development boards. Dynamic circuit switching modelling tool allows the reconfigurable design to be simulated. The reconfiguration process on a XC6216 device is controlled and monitored by XC4013 FPGA. An API is used to generate the configuration and transfer the configuration data to XC4013 through a PCI bus. External FGRHS are used when the logic density of available FPGA is very less when compared with the application requirement. The on-chip FGRHS are not much developed due to limited resources on FPGA, no sophisticated programming support to configure partial fine grain DRRs.

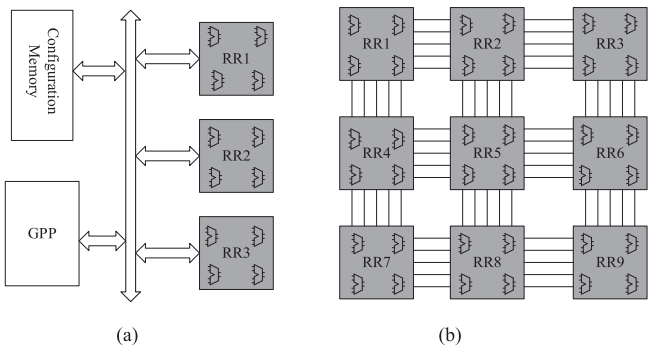

Fig. 1. (a) Standard CGRA (b) Distributed CGRA.

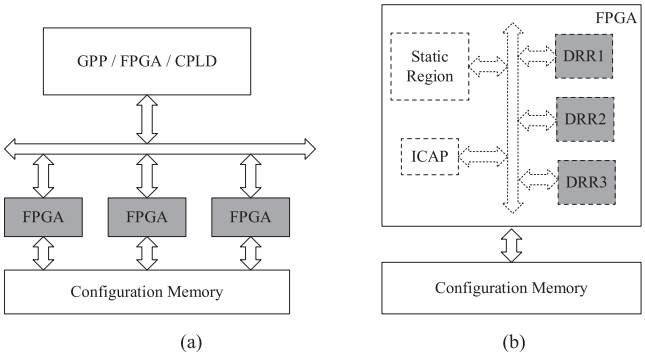

Fig. 2. (a) External FGRA (b) On-chip FGRA.

Xilinx has introduced partial reconfiguration internal port with Spartan6 and Virtex devices which are low cost and high performance FPGA families respectively. However the platform support was not fully incorporated in the software tool, Plan Ahead, until 2006. The vendor supported two different types of reconfiguration flows – namely difference based and module based. Module based flow is to support external FGRHS and difference based flow is for on-chip FGRHS. The approaches though well documented are very tedious procedures, cannot be customized and not flexible. Later, a much flexible tool set was developed for Plan Ahead to promote place and route of fine grain DRR [17]. This has shifted the research from external to on-chip FGRHS reconfiguration. In on-chip FGRHS reconfiguration, DRRs and reconfiguration controller are part of FPGA and a partial dynamic reconfiguration flow is used to reconfigure FGRHS.

JBits is one such Java application [18] that can be used to access internal resources of XC4000 and Virtex family devices. This API enabled dynamic reconfiguration of FPGA devices when there was not much software tool support for generating partial bitstream. However, the API has its limitations like glue logic cannot be implemented – only standard structured functions can be directly implemented, no proper documentation – therefore only proficient user can get maximum out of the tool and no debug support is available at configuration level. JBits is used for dynamic circuit specialization of DES algorithm by creating specific dynamic key and modify encrypt/decrypt mode dynamically [19]. This implementation has comparable improvement in the algorithm performance with respect to speed, area and power. Later various tools like JRTR [20], PARBIT [21], Xilinx EAPR Flow [22] were released for basic FPGA families supporting dynamic configuration loading. Currently, Xilinx PlanAhead and opensource GoAhead [23] were sophisticated tool flows including all the features of place and route for Xilinx Spartan and Vertex FPGAs that are enabled for Dynamic Reconfiguration

3. Basics of partial reconfiguration

Current FPGA advancements support adaptation and upgradation of the design running on the FPGA without affecting the integrity of the entire system. Reconfiguration of entire FPGA is called as reconfiguration and reconfiguring a portion of FPGA is called Partial Reconfiguration. Each design reconfiguration needs to load bitstream into SRAM configuration memory that can configure all the programmable resources of FPGA like programmable interconnects, CLBs, DSP blocks, BRAM etc. A full reconfiguration bitstream consists of configuration bits of entire device along with configuration commands. The size of the bitstream for a full reconfiguration depends on the device. i.e., high gate count device has large bitstream file size. The partial bitstream consists of configuration bits of selected area of FPGA to be reconfigured and the size of bitstream depends on the area selected for reconfiguration.

FPGA has different configuration ports with different data rate to load dynamic configuration data into the device as shown in Table 1 [24,25]. Select MAP and ICAP are high speed configuration choice depending on external and internal configuration respectively. In external reconfiguration, either a host computer will generate online configuration data or an external controller will load the configuration data from a pre-computed source. An internal/self-reconfiguration involves an internal controller load the configuration data from a pre-computed source.

Table 1
Dynamic configuration ports.

Configuration Port	Data Widths (bits)	Frequency (MHz)	External/ Internal
JTAG	1	66	External
Select Map	x8,x16,x32	100	External
ICAP	x8,x16,x32	100	Internal

To perform partial reconfiguration, the design should be partitioned into static and dynamic region. Static region consists of static part of the design that need not be reconfigured and control circuit that performs partial reconfiguration of DRR. Dynamic region consist of that part of the circuit that should be altered partially. Different adaptations of dynamic region are synthesized separately without adding I/O buffers, so that bitstream for each dynamic region partition can be generated individually. The static region is synthesized with an empty black box implementation of dynamic region.

A reconfiguration controller is responsible in loading partial bitstream to the static system without losing its integrity. The bitstream comprises of configuration commands and configuration bits. The configuration commands are used to load the configuration bits in to the addressed configuration memory above the FPGA fabric. If external editing of bitstream is required, then a sequence of command should be used to write configuration bits into the configuration memory.

The bitstreams are transferred to the FPGA fabric through an internal configuration access port (ICAP). ICAP is a configuration port inside the fabric that allows the user to access the configuration information of FPGA from within the fabric. It can be programmed in any one of the allowed data widths (x8, x16, x32). The ICAP primitive is as shown in Fig. 3. The O data bus can be used to debug the configuration process through different status information.

The netlist generated are exported along with constraints to generate netlist into PlanAhead tool. A partition region is selected to accommodate the largest dynamic partition with a minimum selectable area as a frame. The minimum area of FPGA that can be addressed in configuration memory is a frame. The size of partial bitstream depends on the selection of partition area.

4. Evolution of partial reconfiguration platforms

With the introduction of partial reconfiguration technology features into FPGAs, the research in reconfigurable computing have shifted from coarse grain to fine grain owing to the advantages of flexibility of adapting to any application, readily available platform to customize. These in-chip FGRHS were developed to support adaptive signal processing application, fault tolerant systems, software defined radio and other reconfigurable systems. Most of the research is concentrated on development of reconfigurable computing framework using FPGAs, automation framework for configuration management and task scheduling for compiling configuration data.

Many researchers have proposed dynamic reconfiguration frameworks build around Xilinx FPGAs, which are leading SRAM technology based FPGAs, supporting PDR through built-in internal ICAP port. A general architecture of on-chip FGRHS developed and used by most of the researchers is shown in Fig. 4.

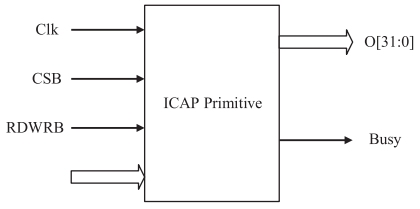

Fig. 3. ICAP primitive.

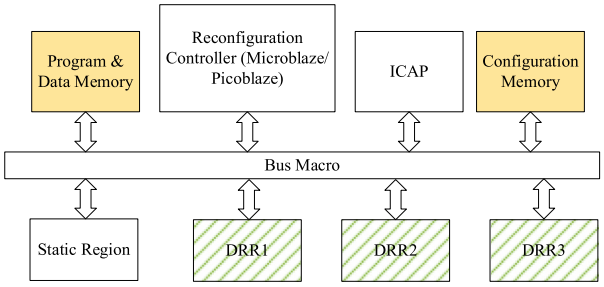

Fig. 4. General architecture of on-chip FGRHS.

It consists of a reconfiguration controller built upon a soft-core Microblaze/hardcore Picoblaze processor. The processor executes the software part of the application while making soft decision on scheduling and loading configuration data from configuration memory to FPGA SRAM memory through ICAP. Program memory stores the software routines for application and reconfiguration controller, while data memory holds processing data. If sufficient memory capacity is available, then configuration, program and data can be placed in a single memory unit. Multiple DRRs can be reconfigured through ICAP port without affecting the operation of static region of the system including reconfiguration controller, ICAP and any other static region required by the application. To evaluate the performance of such frameworks, a cost model is developed by papadimitriou et al. [26]. The reconfiguration time and throughput of PDR systems are measured by analyzing the parameters that affect reconfiguration speed.

$$\text{Reconfiguration Time (RT)} = T_{Mem-Proc} + T_{Proc-ICAP} + T_{ICAP-ConfigMem}$$

The reconfiguration time is depended on the time required to transfer configuration data from external memory to processor ($T_{Mem\text{-}Proc}$), processor to ICAP ($T_{Proc\text{-}ICAP}$) and ICAP to FPGA configuration memory ($T_{ICAP\text{-}ConfigMem}$). The major bottleneck in reconfiguration time are the $T_{Mem\text{-}Proc}$ &$T_{Proc\text{-}ICAP}$ as the processor has to execute software routines to perform both data transfers on a shared data bus. Also the same bus is used to perform regular static system tasks, which further increases the reconfiguration time.

To avoid this lag in reconfiguration time, two-bus architecture was used by few other researchers. This architecture, as shown in Fig. 5, has one bus to carry out reconfiguration process and the other to carryout data transfer for static system. Xilinx provided two different open-source PDRFs that are controlled by an embedded processor. XPS HWICAP [27] controller with PLB bus interface and AXI HWICAP [28] controller with AXI4 bus interface. These frameworks are meant for demonstrating a prototype for RHS, though they are inefficient in terms of reconfiguration data-rate.

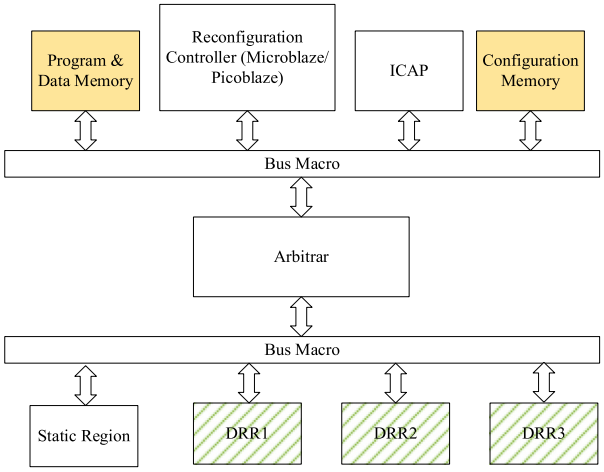

Fig. 5. Two bus architecture of On-chip FGRHS.

A simple and customizable ICAP-I [29], an ICAP interface, is designed with an ICAP interface module, a memory controller and a memory scheduler. Memory controller is an interface to external memory like flash memory or SD RAM that support read, write, and other device specific operations. ICAP interface module controls read-write operations from memory to ICAP through the memory controller. The memory scheduler will share the external memory access between ICAP and the user application without conflicts. In this design the reconfiguration throughput greatly depends upon the memory holding configuration data. The ICAP-I is tested with external SD Flash card and BRAM. The reconfiguration throughput is 29 MB/s with SD card and 180 MB/s with BRAM interface.

A high speed RHS is developed and evaluated the performance using ICAP controllers with three different interface variations [30]. AC_ICAP is a hardware variation with generic interface, PLB_AC_ICAP is a PLB interfaced controller and AXI_AC_ICAP is an AXI interfaced controller. The standalone AC_ICAP has highest reconfiguration rate (381 MB/s) and smaller resource utilization when compared with processor attached controllers. The framework also made it possible to reconfigure at LUT level, while the standard smallest reconfiguration area is considered to be 1 frame. The reconfiguration of LUT is performed without pre-computed configuration bits, while frame reconfiguration is done with stored pre-computed configuration bits.

An intelligent ICAP controller with bitstream compressions [31] is used to improve reconfiguration throughput. The compression algorithm is developed to gain advantage from unused long sequence of zeros in the bitstream. This algorithm is effective only if the DRR is a small part of reconfiguration frame. If the utilization of DRR within frame is 100%, then the possibility of sequence of zeros is less and the compression ratio will also be less.

In [32], a DDR SDRAM based PDRF is designed to achieve highest reconfiguration throughput by increasing the operating frequency of ICAP controller to 370 MHz while Xilinx datasheets claim that ICAP can run at a maximum of 200 MHz only. The configuration bits should be stored in external memory and loaded into volatile DDR SDRAM through MicroBlaze or PowerPC processor core and customized Memory Interface Generator (MIG) core.

A generic reconfiguration manager [33] is designed that can be used with either an processor or any control logic. As the major hold back of reconfiguration speed is the source of configuration data, different memory sources like BPI flash, BRAM, SD flash are considered to evaluate the core. A configuration throughput of 253.04 MB/s is achieved by using BRAM and SD flash together.

In [34], Dobai et al. considered the use of PDR for the implementation of Virtual Reconfigurable Circuits (VRC). Evolvable hardware design for adaptable image filter is implemented on Xilinx Zynq SOC platform. ARM processor is used to run software for evolutionary algorithm and to perform reconfiguration from DDR memory. It is stated that the reconfiguration time has increased in comparison with other Virtex architectures as the size of minimum block that can be configured, called frame, in Zynq platform is more than that in Virtex platform. It is also observed that FPGA based implementation proved faster when compared with ARM based implementation of VRC.

PDR is demonstrated using applications like pixel processing system and FIR filtering system [35,36] on Virtex 4 platform. A compact flash card is used for storing partial bitstream and input data to the application. The partial bitstream from compact flash

are transferred to DDR RAM. Direct Memory Access (DMA) core is used to facilitate fast data transfers from DDR RAM to reconfiguration modules. Ethernet interface is provided to control PDR from PC. The static region resource utilization in this system is nearly equivalent to dynamic region.

Custom hardware ICAP controllers are suggested in [33,37] without using hard or soft processor core. Tarrillo Jimmy et al. [33] evaluated the reconfiguration speed with BPI Flash and SD Flash on Virtex 5 and Virtex 6 FPGA platforms. Reconfiguration speed of this design is half the maximum possible theoretical throughtput of ICAP, which is achieved by using Block RAM. Though the design was designed with less hardware resources, the reconfiguration throughput is low. Also the capacity of on-chip BRAM should increase with the increase in bitstream size. Vipin et al. [38] proposed a high speed partial reconfiguration controller that have attained the near maximum reconfiguration throughput. This is one of the best designs to support PDR. The configuration bitstream are loaded from PC to DDR RAM through a UART port using a DMA controller. The configuration bitstream are then transferred to ICAP through this DMA controller. As the data transfers from a high speed DDR RAM, whose data rate is larger than that of ICAP, the design has best reconfiguration time. But the hardware resources and power dissipation of DDR controller are huge.

Pierre et al. [39] proposed a PDR architecture that relies on PowerPC operating at 100Mhz. PCBs are transferred from ethernet port to ICAP through PowerPC providing remote access reconfiguration. The use of PowerPC increases the physical resource requirement of the system. Hence in this paper, we propose a system independent of a processor that can provide a remote reconfiguration control to the user.

Becker, J., et al. [40] presented a new adaptive software/hardware reconfigurable system, using a real application in the automotive domain implemented on a Xilinx Virtex-II 3000 FPGA. The benefits of adaptive self-reconfiguration in future automobiles are discussed in detail. The system has exploited partial dynamic reconfiguration to reduce power dissipation and increase the adaptiveness of embedded systems. Two-bus architecture is employed so that the application run on the system is not effected by reconfiguration process.

Conger, C., et al. [41] proposed and analyzed partial reconfiguration design methodologies to perform self-reconfiguration on Virtex-4 and Virtex-5 FPGAs. The paper investigated the partial reconfiguration design metrics like bitstream size, clock frequency, and % resource utilization with respect to the aspect ratio of the DRR. The authors, with experimental work, have suggested guidelines in choosing a good placement of DRR for efficient speed performance, and optimal bitstream file size.

A multiple target track system for automotive applications is designed in [42] and evaluated both in terms of performance and FPGA resource utilization to develop reconfigurable platform for a wide range of automotive and multimedia applications. The dynamically reconfigurable MTT system is implemented using the Xilinx ICAP and bus macro technologies. The developing and validating the proposed new hardware platform, is focused on developing the software mechanisms for managing the dynamic reconfiguration of hardware accelerators and the heuristics for maximizing run-time performance.

T. Raikovich and B. Fehr. [43], introduced a Xilinx Virtex-5 FPGA based reconfigurable hardware accelerator system, which has been created mainly for image processing of biological samples. The work aimed at accelerating the implementation of the basic Cell Profiler modules in a reconfigurable FPGA fabric, and built up the processing pipeline from reconfigurable modules. The test system is implemented on the Xilinx ML506 board, which utilizes an XC5VSX50T FPGA optimized for memory-intensive and DSP applications. The partial reconfigurable filtering times are observed to be 70 times faster for median filtering and 30 times faster for FIR filtering when compared with software solution.

Manel Hentati, Yassine Aoudni et al. [27], studied the use of the Reconfigurable Video Coding (RVC) technology for the specification of an application and the design of a system based on the Dynamic Partial Reconfiguration (DPR) functionality. The paper details the study of Inverse Quantization (IQ) algorithm of an MPEG- 4 decoder and an implementation of MPEG-2 and the H263 IQ algorithms using RVC and DPR. The reconfiguration times are found to be very short (about 157 us) compared to the time of an image decoding (typically 40 ms). Hence the authors concluded that that dynamic reconfiguration of FPGAs is a promising approach for saving resources and increasing performance.

Li Wancai; Chen Jianyong; Li Zhenyu [28], presented a general-purpose FPGA-based dynamic reconfigurable platform for image processing. The hardware efficiency and software flexibility are demonstrated by the functional verification of the edge detection and image filtering algorithm on the Virtex-4 development platform.

5. Discussions on reconfiguration platforms

Though CGRHS are mainly used for GPP acceleration, FGRHS are being used for hardware acceleration [46,47], run time design adaptability, evolvable hardware systems [34,48,49], fault tolerant designs [50,51], hardware update and maintenance without suspending the system operation.

From the above reviewed systems and Table 2, it is evident that the RHS using embedded processor [30,31,33,34,36,37,45] couldn't achieve the theoretical throughput of ICAP port. This is due to the execution of long software routines running on a embedded processor to control the reconfiguration process. The use of processor gives greater flexibility to the designer in terms of soft IP cores and software routines but at the cost of large physical resources and slower reconfiguration rates. However few systems have approached the theoretical throughput by using a DMA transfer [34,54,55] between configuration source and ICAP, or by using BRAM/FIFO [30–32,45] to pre-fetch complete partial bitstream before sending to ICAP.

There are custom build reconfiguration controllers that don't require embedded processor to perform reconfiguration process. Few of the designs [44] proved that ICAP can run at higher frequencies than the Xilinx Specification of 100 MHz. It is reported that ICAP can operate functionally correct upto 550 MHz. However, a working RHS platform is not demonstrated. Another PDRF [45], has developed a reconfigurable driver assistant system with an embedded processor and fast ICAP reconfiguration process by increasing the operating frequency of ICAP till 300 MHz. Custom ICAP controllers [31–33,38,53] are developed with a target to increase the reconfiguration throughput and reduce the physical requirements of the reconfiguration controller. However, these PDRFs still have limitation on the reconfiguration speed due to the fact that the source of configuration data has smaller data rate when compared with ICAP data rate. Most of the configuration data sources are on-board memories which require pre-computed reconfiguration bitstream to be preloaded. The platform flash memory, SD card/Compact Flash card has data transfer rate ranging from 50 to 200 Mbps, while ICAP can operate at a maximum throughput of 32 bits @ 100 MHz i.e., 3.2Gbps. Therefore, the designs with any flash memory acting as configuration source will require BRAM/FIFO [31–33] to temporarily store the configuration data during reconfiguration. The memory requirement for such systems increases with the increase in reconfiguration area. Custom reconfiguration controllers with DDR SDRAM as configuration

Table 2
Reconfiguration throughput of different reconfiguration controllers.

Design	PCB transfer	ICAP Frequency (MHz)	ICAP Throughput (MBps)
[Vipin et al. 2012] [38] V6_Custom	PC-UART-DDR3-ICAP	100	399.80
[Tarrillo et al. 2014] [33] V5_Custom	SD Flash-BRAM-ICAP	100	192.15
[Tarrillo et al. 2014] [33] V6_Custom	SD Flash-BRAM-ICAP	100	253.04
[Anitha et al. 2012] [53] V5_Custom	PC - Ethernet - ICAP	100	118.79
[Anitha et al. 2012] [53] V6_Custom	PC - Ethernet - BRAM - ICAP	100	400
[Andres et al. 2015] [30] V5_PLB	SD Flash-ICAP	100	14.66
[Andres et al. 2015] [30] V5_PLB	SD Flash-BRAM-ICAP	100	378.73
[Andres et al. 2015] [30] V5_FSL	SD Flash-ICAP	100	14.67
[Andres et al. 2015] [30] V5_FSL	SD Flash-BRAM-ICAP	100	378.85
[Pham et al. 2012] [32] V5_Custom	DDR3SDRAM-FIFO-ICAP	370	1480*
[Hansen et al. 2011] [44] V5	ICAP	500	2000*
[Hansen et al. 2011] [44] V5	ICAP	533	2132*
[Hansen et al. 2011] [44] V5	ICAP	550	2200*
[Claus et al. 2010] [45]V4_PLB	DDR SDRAM-MPMC-ICAP	140	560
[Claus et al. 2010] [45]V5_PLB	DDR SDRAM-MPMC-ICAP	300	1200
[Liu et al. 2010] [31] V4_Custom (MIPS)	SRAM-FIFO-ICAP	100	392.74
[Dobai et al. 2013][34] Zynq_AXI	DDR3SDRAM-DMA-ICAP	100	380
[Llamocca et al. 2013][36] V4_PLB	SD Flash-DMA-ICAP	100	16.28
[Kulkarni et al. 2013] [37] Zynq_AXI	DDR3SDRAM-MiCAP	100	23

* Throughput is of ICAP IP core only

Table 3
Resource Utilization of Different PDRF.

Design		Slice	FlipFlop	LUT	BRAM
[Vipin et al. 2012] [38] V6_Custom	ICAP contr.	NR	74	38	8
	DMA contr.	NR	598	548	0
	Cmpl. Sys.	NR	NR	NR	NR
[Anitha et al. 2012] [53] V5_Custom	ICAP contr.	10	15	28	4
	Cmpl. Sys.	201	580	535	3
[Anitha et al. 2012] [53] V6_Custom	ICAP contr.	12	14	32	4
	Cmpl. Sys.	190	547	451	3
[Tarrillo et al. 2014] [33] V5_Custom	ICAP contr.	NR	7	13	0
	Mem. contr.	NR	246	752	5
	Cmpl. Sys.	NR	NR	NR	NR
[Tarrillo et al. 2014] [33] V6_Custom	ICAP contr.	NR	7	11	0
	Mem. contr.	NR	247	662	5
	Cmpl. Sys.	NR	NR	NR	NR
[Andres et al. 2015] [30] V5_PLB	ICAP contr.	952	1609	2375	7
	Cmpl. Sys.	2084	3516	4556	23
[Andres et al. 2015] [30] V5_FSL	ICAP contr.	903	1484	2329	7
	Cmpl. Sys.	2094	3450	4643	23
[Pham et al. 2012] [32] V5_Custom	ICAP contr.	59	NR	NR	NR
	Cmpl. Sys.	1759	NR	NR	NR
[Pham et al. 2012] [32] V6_Custom	ICAP contr.	37	NR	NR	NR
	Cmpl. Sys.	1308	NR	NR	NR
[Hansen et al. 2011] [44] V5*	ICAP contr.	27	104	99	0
	Cmpl. Sys.*	NR	NR	NR	NR
[Claus et al. 2010] [45]V4_PLB	ICAP contr.	NR	197	354	2
	Cmpl. Sys.	NR	551	1054	2
[Liu et al. 2010] [31] V4_Custom (MIPS)	ICAP contr.	274	336	367	NR
	Cmpl. Sys.	10,690	10,734	19,702	NR
[Dobai et al. 2013][34] Zynq_AXI	ICAP contr.	NR	NR	NR	NR
	Cmpl. Sys.	NR	644	2325	58
[Llamocca et al. 2013][36] V4_PLB	ICAP contr.	NR	NR	NR	NR
	Cmpl. Sys.	5308	5519	6517	NR
[Kulkarni et al. 2013] [37] Zynq_AXI	ICAP contr.	NR	675	500	1
	Cmpl. Sys.	NR	NR	NR	NR

*Throughput is of ICAP IP core only

source can contest with the theoretical ICAP throughput. But DDR SDRAM is a volatile memory and therefore, a procedure should be devised to load partial configuration data into DDRSDRAM before configuration. Though SDRAM can support data rates upto 6.4 Gbps, using it intern leads to large area overhead.

Table 2 gives a listing of physical resources used by each RHS. This listing is not made for comparison, as it is not appropriate to compare physical resource requirement of various designs developed on different platforms. The RHS are developed on Virtex 4 (V4), Virtex 5 (V5), Virtex 6 (V6), and Xilinx SOC device Zynq with ARM Embedded processor. Majority of the research work focused on improving the reconfiguration throughput at a cost of increased hardware resources. The advantage of reconfigurable computing will be considered only if the PDRF occupies a fractional part of the whole system. The PDRF includes ICAP controller, a memory controller/a peripheral controller that delivers configuration data

to ICAP, bus infrastructure, and may include DMA controller. Use of high data-rate memories like DDR SDRAM needs controllers with huge physical resources. As an alternative, one can use an host computer to hold the configuration data and send it to FPGA on demand through peripheral ports like USB, UART, Ethernet etc [38,39,52,53]. However, the reconfiguration throughput again depends on the throughput of peripheral (See Table 3).

Though FGRHS can provide reconfiguration at fine granularity making it possible to change minor functional details of DRR, the minimum area that can be reconfigured is limited by one reconfiguration frame region of the chosen platform. For example, Virtex 4, Virtex 5 and Virtex 6 have a reconfiguration frame of a CLB column consisting of 16 CLBs, 20 CLBs, and 40 CLBs respectively. PDRF that can reconfigure much smaller regions like LUT are also developed [28,30]. The flexibility of LUT reconfiguration is achieved at a cost of enormous complexity of PDRF. Hence the flexibility of reconfiguration, reconfiguration throughput and complexity of PDRF have always been a challenging constraints to each other.

6. Conclusions

An overview of reconfigurable computing is presented, while describing the architectural evolution in CGRHS and FGRHS, with an emphasis on FGRHS and PDRF for FPGA. From the above discussions, it is observed that a good amount of work is done in improving reconfiguration throughput of a FGRHS. To use reconfiguration to all dimensions of applications, high flexibility in choosing the DRR and generating configuration data is essential while using reasonable physical resources in comparison with DRR. To extend the use of RHS to service and maintenance sector, remote tool development is required to create on-line place and route, validate configuration, and provide security from unauthorized access.

CRediT authorship contribution statement

T. Siva Sankar Phani: Investigation, Resources, Visualization. **Anitha Arumalla:** Conceptualization, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Supervision. **M. Du rga Prakash:** Conceptualization, Investigation, Resources, Writing - original draft, Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Mehra, Wong, Majithia, A Comparative Study of Some Two-Processor Organizations, IEEE Trans. Comput.C-29 (1980) 44–49.
[2] S.K. Mehra, J.C. Majithia, Reconfigurable computer architectures, IEE Proc. Comput. Digit. Tech. (1982) 156–164.
[3] P.M. Athanas, H.F. Silverman, Processor reconfiguration through instruction-set metamorphosis, Computer. 26 (1993) 11–18.
[4] R. Hartenstein, A Decade of Reconfigurable Computing: A Visionary Retrospective, in: Proc. Des. Autom. Test, IEEE Comput. Soc, 2001: pp. 642–649.
[5] C. Ebeling, D.C. Cronquist, P. Franklin, RaPiD — Reconfigurable Pipelined Datapath, in: 6th Int. Work. Field-Programmable Log. Appl., Springer Berlin Heidelberg, Germany, 1996: pp. 126–135.
[6] S.C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, R.R. Taylor, PipeRench: A reconfigurable architecture and compiler, Computer. 33 (2000) 70–77.
[7] D. Chen. Conrad, Programmable Arithmetic Devices for High Speed Digital Signal Processing, University of California at Berkeley, 1992.
[8] E. Mirsky, A. DeHon, MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources, in: Proc. FPGAs Cust. Comput. Mach., IEEE, 1996: pp. 157–166.
[9] J. Babb, M. Frank, V. Lee, E. Waingold, R. Barua, M. Taylor, J. Kim, S. Devabhaktuni, A. Agarwal, The RAW Benchmark Suite: Computation Structures for General Purpose Computing, in: Proc. 5th Annu. Symp. F. Program. Cust. Comput. Mach., IEEE Comput. Soc, 1997: pp. 134–143.
[10] R. Hartenstein, Trends in Reconfigurable Logic and Reconfigurable Computing, in: Proc. 9th Int. Conf. Electron. Circuits Syst., IEEE, 2002: pp. 801–808.
[11] T. Cervero, S. Lopez, R. Sarmiento, Dynamically Reconfigurable Architectures for Multimedia Applications, in: Proc. Des. Circuits Integr. Syst., 2009: pp. 29–34.
[12] Zain-ul-Abdin, B. Svensson, Evolution in architectures and programming methodologies of coarse-grained reconfigurable computing, Microprocess. Microsyst. 33 (2009) 161–178.
[13] J. Burns, A. Donlin, J. Hogg, S. Singh, M. De Wit, A Dynamic Reconfiguration Run-time System, in: Proc. 5th Symp. Field-Programmable Cust. Comput. Mach., IEEE Comput. Soc, 1997: pp. 66–75.
[14] G. McGregor, D. Robinson, P. Lysaght, A Hardware/Software Co-design Environment for Reconfigurable Logic Systems, in: Field-Programmable Log. Appl. From FPGAs to Comput. Paradig., Springer Berlin Heidelberg, 1998: pp. 258–267.
[15] J.G. Eldredge, B.L. Hutchings, Density Enhancement of a Neural Network using FPGAs and Run-time Reconfiguration, in: Proc. Work. FPGA's Cust. Comput. Mach., IEEE Comput. Soc. Press, 1994: pp. 180–188.
[16] J. Hadley, B. Hutchings, Designing a Partially Reconfigured System, in: F. Program. Gate Arrays Fast Board Dev. Reconfigurable Comput., 1995.
[17] N. Dorairaj, E. Shiflet, M. Goosman, PlanAhead software as a platform for partial reconfiguration, Xcell J. (2005).
[18] S. Guccione, D. Levi, P. Sundararajan, S. Jose, JBits: A Java-based Interface for Reconfigurable Computing, in: 2nd Annu. Mil. Aerosp. Appl. Program. Devices Technol. Conf., 1999: pp. 253–261.
[19] C. Patterson, High Performance DES Encryption in VirtexTM FPGAs using JBitsTM, in: Proc. FPGAs Cust. Comput. Mach., IEEE, 2000: pp. 113–121.
[20] S. McMillan, S.A. Guccione, Partial Run-Time Reconfiguration using JRTR, in: Field-Programmable Log. Appl, in: Roadmap to Reconfigurable Comput, Springer, Berlin Heidelberg, 2000, pp. 352–360.
[21] E. Horta, J. Lockwood, PARBIT: A Tool to Transform Bitfiles to Implement Partial Reconfiguration of Field Programmable Gate Arrays (FPGAs), 2001.
[22] D. Lim, M. Peattie, Two Flows for Partial Reconfiguration: Module Based or Small Bit Manipulations, San Jose, CA, 2002.
[23] D. Koch, J. Torresen, C. Beckhoff, D. Ziener, C. Dennl, V. Breuer, J. Teich, M. Feilen, W. Stechele, Partial Reconfiguration on FPGAs in Practice – Tools and Applications, in: Proc. ARCS Work. (ARCS), 2012, 2012: pp. 1–12.
[24] Xilinx, Inc, Virtex-6 Family Overview, 2015.
[25] Xilinx, Inc, Virtex-6 FPGA Configuration User Guide (UG360).
[26] K. Papadimitriou, A. Dollas, S. Hauck, Performance of Partial Reconfiguration in FPGA Systems, ACM Trans. Reconfigurable Technol. Syst. 4 (2011) 1–24.
[27] Xilinx, Inc, Xilinx DS586 LogiCORE IP XPS HWICAP (v5.01a), 2011.
[28] Xilinx, Inc, AXI HWICAP v3.0 LogiCORE IP Product Guide (PG134), 2016.
[29] V. Lai, O. Diessel, ICAP-I: A Reusable Interface for the Internal Reconfiguration of Xilinx FPGAs, in: Proc. Int. Conf. Field-Programmable Technol. 2009: pp. 357–360.
[30] L.A. Cardona, C. Ferrer, AC-ICAP: A Flexible High Speed ICAP Controller, Int. J. Reconfigurable Comput. 2015 (2015) 1–15.
[31] S. Liu, R.N. Pittman, A. Forin, Minimizing Partial Reconfiguration Overhead with Fully Streaming DMA Engines and Intelligent ICAP Controller, in: Proc. 18th Annu. ACM/SIGDA Int. Symp. F. Program. Gate Arrays - FPGA '10, 2010: p. 292.
[32] H.-M. Pham, V.-C. Nguyen, T.-T. Nguyen, DDR2/DDR3-based Ultra Rapid Reconfiguration Controller, in: 2012 Fourth Int. Conf. Commun. Electron., IEEE, 2012: pp. 453–458.
[33] J. Tarrillo, F.A. Escobar, F.L. Kastensmidt, C. Valderrama, Dynamic Partial Reconfiguration Manager, in: Proc. 5th Symp. Circuits Syst., IEEE, 2014: pp. 1–4.
[34] R. Dobai, L. Sekanina, Image Filter Evolution on the Xilinx Zynq Platform, in: Proc. NASA/ESA Conf. Adapt. Hardw. Syst., IEEE, 2013: pp. 164–171.
[35] D. Llamocca, M. Pattichis, G.A. Vera, D. Llamocca, M. Pattichis, G.A. Vera, Partial reconfigurable FIR filtering system using Distributed arithmetic, Int. J. Reconfigurable Comput. 2010 (2010) 1–14.
[36] D. Llamocca, M. Pattichis, A dynamically reconfigurable pixel processor system based on power/energy-performance-accuracy optimization, IEEE Trans. Circuits Syst. Video Technol. 23 (2013) 488–502.
[37] A. Kulkarni, V. Kizheppatt, D. Stroobandt, MiCAP: A Custom Reconfiguration Controller for Dynamic Circuit Specialization, in: Proc. Int. Conf. ReConFigurable Comput. FPGAs, 2015: pp. 1–6.
[38] K. Vipin, S.A. Fahmy, A High Speed Open Source Controller for FPGA Partial Reconfiguration, in: Proc. Int. Conf. Field-Programmable Technol, IEEE: pp. 61–66.
[39] P. Bomel, J.P. Diguet, A. Networked, Lightweight and partially reconfigurable platform, Reconfigurable Comput. Archit. Tools Appl. (2008) 318–323.
[40] J. Becker, M. Hübner, G. Hettich, R. Constapel, J. Eisenmann, J. Luka, Dynamic and partial FPGA exploitation, Proc. IEEE (2007) 438–452.
[41] C. Conger, A. Gordon-Ross, A.D. George, Design Framework for Partial Run-Time FPGA Reconfiguration., in: Proc. Int. Conf. Eng. Reconfigurable Syst. Algorithms, ERSA 2008, Las Vegas, Nevada, USA, 2008: pp. 122–128.
[42] N. Harb, S. Niar, J. Khan, M. Saghir, A reconfigurable platform architecture for an automotive multiple target tracking system, ACM SIGBED Rev. (2009).
[43] T. Raikovich, B. Fehér, Application of Partial Reconfiguration of FPGAs in Image Processing, in: PRIME 2010 6th Conf. Ph. D. Res. Microelectron. & Electron. 18-21 July 2010, Berlin Inst. Technol., IEEE, Germany, 2010: pp. 1–4.

[44] S.G. Hansen, D. Koch, J. Torresen, High Speed Partial Run-Time Reconfiguration Using Enhanced ICAP Hard Macro, in: Proc. Int. Symp. Parallel Distrib. Process. Work. Phd Forum, IEEE, 2011: pp. 174–180.

[45] C. Claus, R. Ahmed, F. Altenried, W. Stechele, Towards Rapid Dynamic Partial Reconfiguration in Video-Based Driver Assistance Systems, Springer, Berlin Heidelberg, 2010, pp. 55–67.

[46] C. Claus, F.H. Müller, J. Zeppenfeld, W. Stechele, A New Framework to Accelerate Virtex-II Pro Dynamic Partial Self-Reconfiguration, Proc. 21st Int. Parallel Distrib. Process. Symp. (2007) 1–7.

[47] A. Sudarsanam, R. Barnes, J. Carver, R. Kallam, A. Dasu, Dynamically reconfigurable systolic array accelerators: a case study with extended Kalman filter and discrete wavelet transform algorithms, IET Comput. Digit. Tech. 4 (2010) 126.

[48] S. Soleymani, A. Noore, Dynamically reconfigurable evolutionary multi-context robust cellular array design, Int. J. Circuits Archit. Des. 2 (2016) 1.

[49] C. Valderrama, L. Jojczyk, P. Possa, Convergence in reconfigurable embedded systems, IEEE Int. Conf. Electron. Circuits, Syst. ICECS (2010, 2010,) 1144–1147.

[50] F. Kastensmidt, P. Rech, Fault Tolerant Manager Core for Dynamic Partial Reconfiguration in FPGAs, in: F. Kastensmidt, P. Rech (Eds.), FPGAs Parallel Archit, Springer International Publishing, Cham, Aerosp. Appl., 2016, p. 319.

[51] S.S. Erdogan, T. Shaneyfelt, G.S. Ng, A. Wahab, Dynamically Reconfigurable FPGA for Robotics Control, in: Proc. 10th Int. Conf. Control. Autom. Robot. Vis., 2008: pp. 2277–2282.

[52] D. Llamocca, M. Pattichis, A. Vera, J. Lyke, Dynamic Partial Reconfiguration Through Ethernet Link, in: AIAA Infotech@aerosp. 2010, American Institute of Aeronautics and Astronautics, Reston, Virigina, 2010: pp. 1–4.

[53] A. Anitha, M. Madhavi Latha, Partial dynamic reconfiguration framework for FPGAs through remote access, Int. J. High Perform. Syst. Archit. 7 (2) (2017) 98–104.

[54] Nagham Samir et al., Energy-adaptive lightweight hardware security module using partial dynamic reconfiguration for energy limited internet of things applications, 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019.

[55] Giacomo Valente et al., Dynamic partial reconfiguration profitability for real-time systems, IEEE Embedded Syst. Lett. (2020).