Journal of Semiconductors
(2020) 41, 022402
doi: 10.1088/1674-4926/41/2/022402

ARTICLES

HRM: H-tree based reconfiguration mechanism in reconfigurable homogeneous PE array

Junyong Deng[1], Lin Jiang[2, †], Yun Zhu[1], Xiaoyan Xie[3], Xinchuang Liu[1], Feilong He[3], Shuang Song[4], and L. K. John[4]

[1]School of Electronic Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710121, China
[2]School of Communication and Information Engineering, Xi'an University of Science and Technology, Xi'an 710054, China
[3]School of Computer, Xi'an University of Posts & Telecommunications, Xi'an 710121, China
[4]The University of Texas at Austin, TX 78712, USA

Abstract: In order to accommodate the variety of algorithms with different performance in specific application and improve power efficiency, reconfigurable architecture has become an effective methodology in academia and industry. However, existing architectures suffer from performance bottleneck due to slow updating of contexts and inadequate flexibility. This paper presents an H-tree based reconfiguration mechanism (HRM) with Huffman-coding-like and mask addressing method in a homogeneous processing element (PE) array, which supports both programmable and data-driven modes. The proposed HRM can transfer reconfiguration instructions/contexts to a particular PE or associated PEs simultaneously in one clock cycle in unicast, multicast and broadcast mode, and shut down the unnecessary PE/PEs according to the current configuration. To verify the correctness and efficiency, we implement it in RTL synthesis and FPGA prototype. Compared to prior works, the experiment results show that the HRM has improved the work frequency by an average of 23.4%, increased the updating speed by 2×, and reduced the area by 36.9%; HRM can also power off the unnecessary PEs which reduced 51% of dynamic power dissipation in certain application configuration. Furthermore, in the data-driven mode, the system frequency can reach 214 MHz, which is 1.68× higher compared with the programmable mode.

Key words: H-tree based reconfiguration mechanism (HRM); Huffman-coding-like addressing; programmable mode; data-driven mode; homogeneous PE array

Citation: J Y Deng, L Jiang, Y Zhu, X Y Xie, X C Liu, F L He, S Song, and L K John, HRM: H-tree based reconfiguration mechanism in reconfigurable homogeneous PE array[J]. *J. Semicond.*, 2020, 41(2), 022402. http://doi.org/10.1088/1674-4926/41/2/022402

1. Introduction

With the increasingly complex, ever-changing and ever-increasing performance requirements of applications, the computing capability and flexibility of existing hardware platforms have been challenged, especially in video processing[1], graphics processing[2, 3], and artificial intelligence[4, 5] applications, etc. Video processing has been one of the most popular domains for multiple decades. Modern video processing technology aims to achieve high resolution, 3-dimensional representations while supporting multiple standards with low power consumption. To achieve this, software/system experts leverage multiple novel algorithms in a single processing framework to satisfy the requirements of various execution phases. The same situation has also been encountered in graphics processing, which has many different algorithms with different efficiencies for different rendering scenarios[6]. How to efficiently support multiple algorithms from the hardware perspective arises as a critical design challenge, and has gained a lot interest in both academia and industry community.

The traditional ASIC (application specific integrated circuit) platform possesses the highest performance but cannot support algorithm expendability[7]. The multi-core platform has the highest flexibility but do not possess the fast computing capability and low power dissipation[8]. With the increasing demand of flexibility, performance and application-specific optimization, the reconfigurable architecture paradigm, which yields high performance, programmability and low costs, has become a popular implementation platform[6, 9].

Reconfigurable architectures can be classified into fine-grained, coarse-grained or hybrid grained on the basis of the granularity at which reconfiguration is done[10]. Coarse grained reconfigurable architectures (CGRAs), compared to FPGAs, incur lower reconfiguration area (66% to 99.06%) and energy costs (88% to 98%)[11], while FPGA is easier in partial reconfiguration since it utilizes the LUT (look-up table) as the basic operating unit, but the large amount of LUTs leads to huge configuration files and hence a long time for reconfiguration due to the overhead of bit-level reconfigurability[12]. CGRA-based multi-core architecture then emerged to accelerate the entire application by running separate kernels or inter-dependent kernels in parallel[13], however, these architectures are not flexible enough to adaptively support various cases of the kernel-level parallelism (KLP) because the resources cannot be efficiently utilized under monotonous aggregation of multiple/many CGRAs, which results in either high power dissipation or performance bottleneck[14]. There-

Correspondence to: L Jiang, jianglin@xust.edu.cn

Received 27 OCTOBER 2019; Revised 13 DECEMBER 2019.

fore, boosting the efficiency of reconfigurable platform is considered as a critical concern.

To solve such practical challenges, we propose a new reconfiguration mechanism for modern video and graphics processing on the customized CGRA-like architectures or processing elements (PE) array. This reconfiguration mechanism, which is called H-tree based hierarchical reconfiguration mechanism (HRM)[15], focuses on fast updating and the simultaneous invoking of reconfiguration context and scalability of computing resource, which can also shut down the PE/PEs that are not necessary for the current configuration to reduce the power consumption. Originally, the H-tree is widely used to achieve simultaneous delivery of clock signal in large-scale IC (Integrated Circuit) design without skew. Compared to its original use of transferring clock signal, the amount of reconfiguration contexts needs to be delivered in the reconfigurable domain is much larger. Hence, achieving simultaneous update is difficult. Huffman coding is the best coding method for constructing variable length character based on the probability of character occurrence. At each encoding node, the character is determined as ‘0’ or ‘1’ according to the probability of the code appearing, and the final encoding is generated in this way. Similarly, The HRM generates the address code of the destination PE based on the principle of ‘0’ for turning left and ‘1’ for turning right. The HRM with Huffman-coding-like and mask addressing method can transfer reconfigurable contexts in unicast, multicast or broadcast mode and trigger them simultaneously. We also proposed a CGRA-like homogeneous PE array with HRM for video and graphics processing to verify the proposed approach. The PE can support both programmable mode and data-driven mode. The proposed PE array with HRM can achieve comparable performance with ASIC when work is done in data-driven mode.

The contributions of this paper can be summarized as follows:

We propose a novel H-tree based hierarchical reconfiguration mechanism to achieve high-speed reconfiguration on modern CGRA-like architectures or PE arrays, which achieves more than 20% improvement in working frequency and is 2× faster in context updating.

(1) We leverage the Huffman-coding-like addressing and mask addressing methods to achieve the simultaneous movement of reconfiguration contexts among particular/related PEs in unicast, multicast, and broadcast modes. In addition, such achievement provides the scalability for computing resources, and can power-off the unnecessary PEs in current scenario to decrease power consumption.

(2) We design a homogeneous PE array to verify the HRM, while the PE supports both programmable mode and data-driven mode, and it can achieve comparable performance with ASIC in data-driven mode.

The rest of this paper is organized as follows. Section 2 summarizes the related works of remapping context transmission and management strategies. Section 3 presents the motivation of the proposed reconfiguration mechanism. In Section 4, we elaborate the methodology in details, including both the HRM and also the associated PE array architecture. Section 5 depicts the architecture modeling, RTL design with Verilog and implementation of prototype system. Section 6 demonstrates the evaluation results and Section 7 concludes the paper.

2. Related works

In order to enhance the resource utilization (also the computing capability) and the flexibility of a reconfigurable system, the design of reconfiguration mechanism has gained tremendous amount of attention. Such mechanisms usually control reconfigurable context transmission and manage such contexts. In this section, we review context transmission technologies and context management strategies separately.

2.1. Context transmission

Run-time remapping tries to change the physical location of a task to decrease the requirements of memory bandwidth, communication costs[16]. Dynamic parallelism explores the parallelism of a task to induce speedup[17]. However, the existing re-mappers mostly target packet-switched NoC (network on chips) and are therefore not applicable to most CGRAs[14] or CGRA-based multi-core architecture. As for FPGA, the typical example of fine-grained reconfigurable architecture, which is claimed as supporting partial or dynamic reconfiguration, cannot achieve the goal actually because of the large size of configuration file which results long configuration time and leads to inadequate flexibility[18, 19].

2.2. Context management

The RSF[13] approach utilizes the pipelining of kernel-stream and intra/inter-CGRA co-reconfiguration to achieve the design objectives of minimal interconnection overhead and flexibility comparable to completely connected fabric (CCF). It accomplishes the objectives with the sharing of context memory (CMs) and data buffers (DBs). The TransMap[11] method stores only one implementation scheme in the configuration memory and applies a series of transformations to the stored bit-stream for remapping or parallelizing an application. In order to accelerate the transmission of contexts, some context compression methods have also been developed[20, 21]. The fat binary approach stores multiple task mapping schemes for each application at compile time and dynamically chooses the appropriate one at runtime[22]. The decision of choosing the appropriate scheme is application-dependent. And in HReA[10], where the whole PE array shares one CM with one data access port.

3. Motivation

In this section, we present the motivation of our approaches. The primary motivation comes from the actual speed requirements of switching among multimedia applications and the scalability analysis of current reconfigurable mechanisms.

3.1. Fast updating and invoking of reconfigurable context

Fast function switching is critical for the performance of reconfigurable platform. The representative of fine grained reconfigurable architecture, FPGA (field programmable gate array), consumes milliseconds of time to reorganize the function due to its bit-level reconfigurability, such as 534 ms for Stratix5-GXA3[19]. In the RSF approach mentioned in the Ref. [13], one CM can be shared by the two CGRAs next to it,

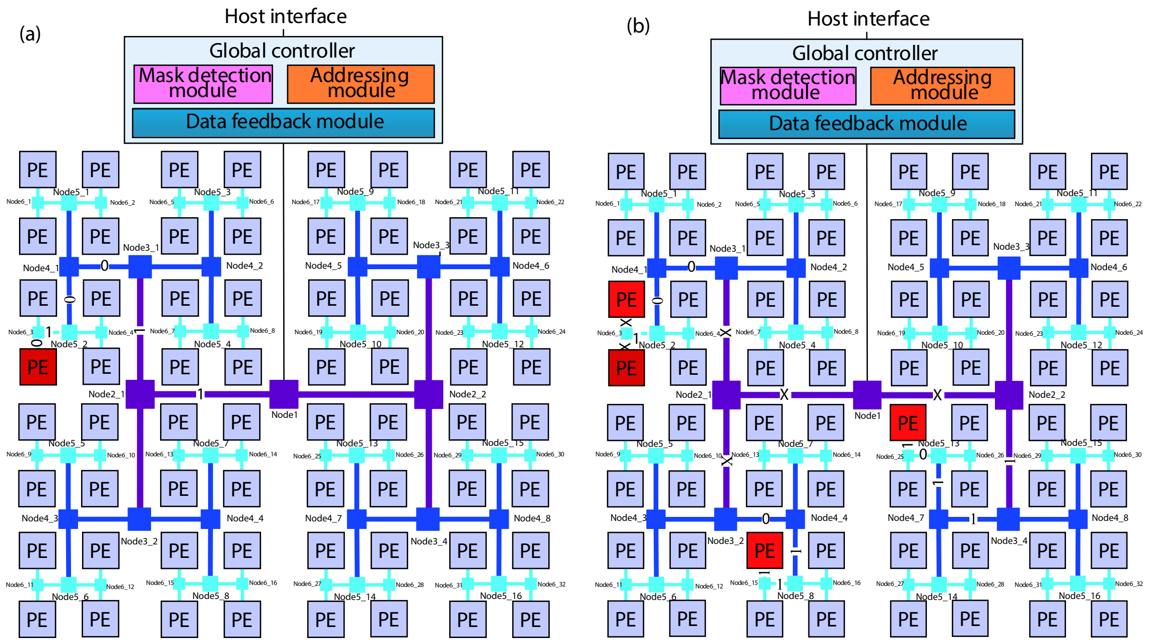

Fig. 1. (Color online) The topology of HRM. (a) Unicast. (b) Multicast/broadcast.

so one CGRA can be configured to execute the tasks following the contents of two different CMs which are located at the different directions of it. This means that one CGRA can work in at least two different configurations. However, the updating of the CM's content is not described in the RSF method. The CM in RSF is of 4 kB size, which needs a considerable amount of time to fill in, and it is also critical for the partial or dynamic reconfiguration of the computing resources. The context compression related approaches need decompression process, which will consume additional time for decompression and also increase the design overhead. We propose a H-tree based hierarchical reconfiguration mechanism which can propagate the context/instruction information from the controller or the host end in one cycle. When the computing resources (PEs in our approach) work in the SIMD or data-driven mode, the reconfiguration of a PE can be done in only a single clock cycle. The contexts or instructions stored in the memory can be invoked with a call-like instruction and the RAM part of the context memory can be updated prior to the call according to the performance model, which will hide the latency of updating process. The call-like instruction invokes parts or all the contents according to the instruction name which is similar to the API (application programming interface) function such as OpenGL commands. As for the start and stop of transmission, there should be a performance evaluation model which is application-dependent.

3.2. Scalability of reconfiguration mechanism

With the new transistor process and tri-gate fin design, a single chip will accommodate more computing resources[23]. So, the reconfiguration mechanism should facilitate this reality with minimal interconnection overheads increasing. In order to achieve this goal, the reconfiguration mechanism should support the scalability of computing resources. The reconfiguration mechanism we proposed can access each PE with 10-bit address bus at the root and 1-bit bus at the PE end in one clock cycle in our 32 × 32 PE array architecture. This mechanism possesses the characteristic of scalability, which can be expended easily to larger scale PE array. For example, with 1-bit more address bus at the root of H-tree, it can access 2048-PE array with little performance degradation.

Motivated by the above description, and also the power consumption requirement, which is a key challenge in large scale system on chip (SoC), we propose HRM to increase reconfiguration efficiency, minimize interconnection costs while supporting high-throughput operating modes such as SIMD (single instruction multiple data) and data-driven for the entire system to achieve, and control the power dissipation by shutting down unused computing resources in specific scenarios. The proposed method is one kind of system solution for video processing and graphics processing applications.

4. Methodology

This section discusses the HRM with Huffman-coding-like addressing and mask addressing firstly, and then demonstrates the HRM applied reconfigurable PE array architecture.

4.1. HRM

In order to improve the efficiency of reconfiguration (including the fast updating and invoking of reconfigurable context), minimize interconnection costs, and also control the power consumption, we propose HRM — a novel reconfiguration mechanism. The HRM leverages Huffman-coding-like addressing and mask addressing, which can be easily applied on a homogeneous thin-core PE array (for example, consisting with 1024 PEs). The HRM structure is shown in Fig. 1. In order to show it clearly, Fig. 1 only presents 8 × 8 PEs.

HRM is designed from the idea of clock-tree with H-type, which can guarantee the minimal time skew so that the information can arrive at every destination at the same time. The video processing and graphics processing need some associated PEs to work together as a certain function unit. For example, the reconfigurable video processing system requires multiple PEs to execute the same task, such as intra-prediction (Intra), integer motion estimation (IME), fractional motion estimation (FME), motion compensation (MC), DeBlocking filter (DB), etc. And in graphics processing field, the vertex shading, the plane-clip, projection, 3D-Clip, pixel shading, primitive assemble unit, back-face culling, viewport transformation, fragment operating unit and so on are also need several PEs to work together[24–26]. The associated PEs should also start simultaneously to process in parallel.

The global controller determines the operating mode and selects the appropriate algorithm for PEs. HRM is used to transfer the instructions or configuration messages to a specific PE or associated PEs, and collect status information of each PE to the global controller. When the global controller transfers data to or from the PEs via HRM, it utilizes Huffman-coding-like code to address each PE. Using the 1024-PE array as an example, HRM addressing bus is of 10-bit width at the root (the bus end which connects to the controller). HRM will take 10-level turns to access the destination end (each PE). When the bit of the bus is logic ‘1’, HRM will take a right turn; If the bit[9] is logic ‘0’, HRM will take a left turn. The bus width will be 9-bit width after the 1st turn, 8-bit width after the 2nd turn, and soon. When the information delivered by HRM reaches the PE, the bus is only 1-bit width. The 10-bit address can also be used as mask information transmission, which is used for multicast or broadcast of reconfiguration context. This kind of addressing is easy to expand to facilitate to the larger scale PE array with low cost. For example, if the PE array is of 2048 PEs, the bus width is only 1-bit wider at the root. In addition to the addressing bus, there is 1-bit power control signal, which is leveraged to realize the clock gating. This bit is logic ‘1’ by default. After the configuration, HRM will set this bit ‘0’ and disable the clock of each unused PE in current scenario.

With the Huffman-coding-like addressing and mask addressing method, HRM can transmit/receive four kinds of instruction/context information with 32-bit width:

(1) The operation instruction with issuing format of: 2-bit flag of ‘01’ + 30-bit RISC-like instruction.

(2) The ‘call’ instruction with issuing format of: 2-bit flag of ‘11’ + 16-bit invoking address + 14-bit extensible space.

(3) The broadcast SIMD/data-driven instruction with issuing format of: 2-bit flag of ‘10’ + 30-bit RISC-like instruction. In this format, the broadcast mask should be transmitted first with the 2-bit flag of ‘10’, and at each turn, the switch node will mark the transmitting direction according to the bit value. If that bit is logic ‘1’, the instruction sent next will be transmitted in both directions; as for logic ‘0’, the instruction sent next will be transmitted according to the address bit value.

(4) The status feedback with issuing format: 2-bit flag of ‘00’+30-bit extensible space. This is used for collecting the status of each computing resource for performance evaluation, and it is running parallel with the computing resource without disturbing the normal operation.

The global controller monitors the operating status of each functional unit, determines the operating mode, and selects a particular algorithm based on the performance evaluation model in real time for PEs. When the host interface accesses the array processor, the global controller receives the bus information from the host interface and judges whether the instruction is unicasted, multicasted, broadcasted, or fed back according to the flag.

4.2. PE array structure

The HRM is applied on the homogeneous thin-core PE array. In order to verify the availability of HRM, we also design a reconfigurable PE array. The proposed PE array is different from the CGRA-based multi-core architecture[13], but with similar functionalities and the same 2D mesh structure. Such similarities guarantee that the HRM can be adopted by other state-of-the-art CGRA-based architectures directly. The CGRA-based multi-core architecture typically includes general purpose processors, multi-CGRA, and on-chip communication architecture such as NoCs or on-chip bus. The general purpose processor mainly executes control instructions and irregular code segments. The multi-CGRA performs data-intensive kernel code segments. The proposed PE array composes of 1024 homogeneous thin-core RISC-like (reduced instruction-set computer) processing elements. The 1024 PEs is coupled directly with adjacent interconnect, in which each 4 × 4 PEs form a logical PE cluster (PEC), as shown in Fig. 2(b).

The 1024 PE array consists of 64 clusters organized in 8 × 8 structure. In many applications, there exists remote data communication between different PEs. The adjacent interconnect and distributed memory cannot efficiently and flexibly handle such communications. We develop an inter-cluster communication structure which connects the PEC with a router (R) through network adapter (NA). The topology is shown in Fig. 2(a).

The PE communicates with the adjacent PE through four shared registers (RN, RE, RW,RS, which represents the four directions, north, east, west, and south respectively), which can be called the NEWS adjacent interconnect[27]. These registers can be accessed directly by each adjacent PE. Each PE has 12 local registers and 4 shared registers. Totally, each PE can manipulate 16 registers. Since the PE is RISC-like processor, it can burden the role of general purpose processor to process intensive control instructions or irregular tasks. In addition, since there are many PEs, they can handle the ILP (instruction level parallelism) computing mode.

The PE can also be configured as data-driven mode for improving the performance and energy efficiency, which can be enabled or disabled by the control commands transmitted by HRM. As shown in Fig. 2(c), the bold lines indicate the control flow and data flow after the data-driven mode is enabled, and the control flow mainly includes the operation code and the data path reconstruction signal, where the dataflow comes from the last-level PE. When data-driven mode is enabled, the fetch stage, the decode stage, and the write back stage are stalled by the gated clock. The data from the last-level PE is directly transmitted to the execution unit, and the corresponding operation is performed according to the operation code received from the HRM, then the result is sent to the adjacent PE through four shared registers directly.

The instruction memory (IM) of each PE used to buffer the operating instructions/contexts. The operations of a particular application are stored in the IM as default configuration when power on the system. As the operating instruction is stored in the memory, the proposed reconfiguration mechanism may sound no different than fat binary. However, the IM also can receive run-time reconfiguration contexts from the global controller through the HRM based on the selected per-

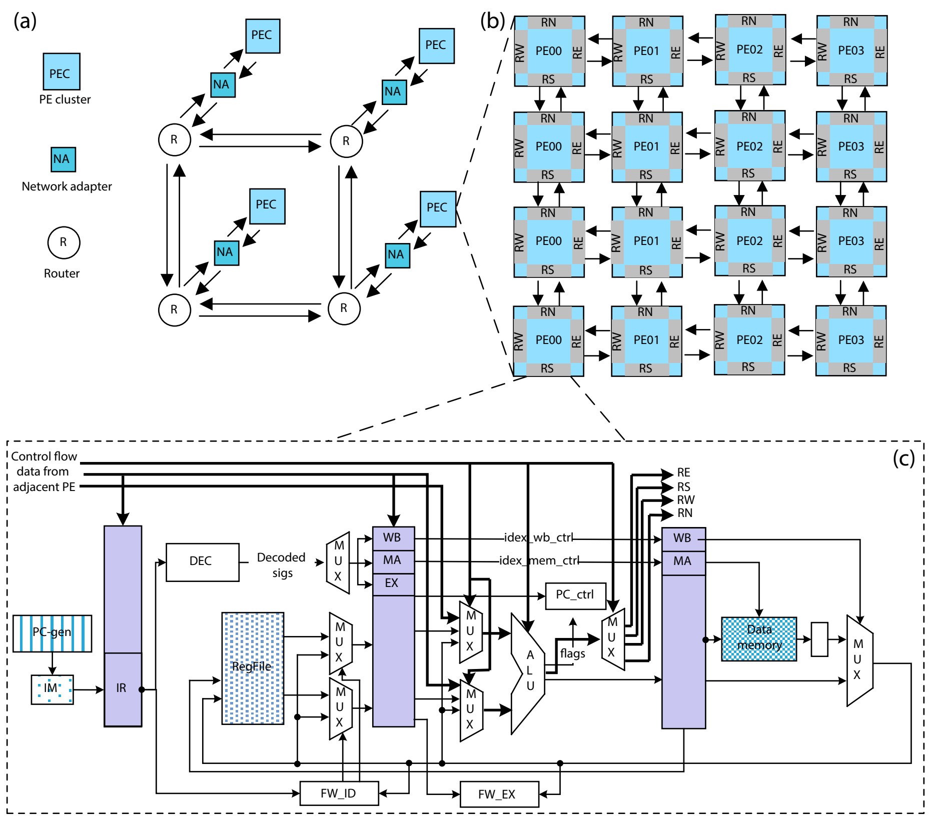

Fig. 2. (Color online) Homogeneous thin-core PE array. (a) Inter –cluster communicate. (b) PE cluster. (c) Micro-architecture of PE.

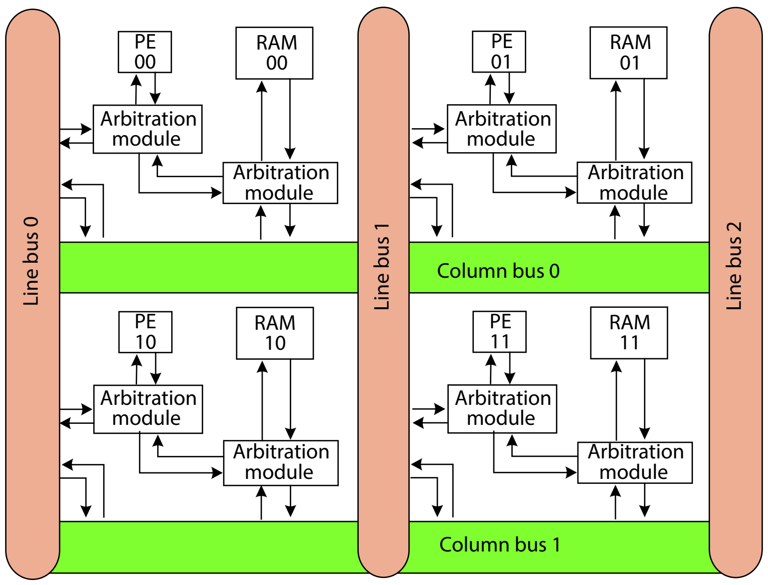

Fig. 3. (Color online) Distributed shared memory structure.

formance model. Under the help of HRM, the reorganized function running on one PE or several associated PEs can start simultaneously, which is very different from fat binary. With the instructions stored in default, or the real-time instructions passed from the global controller through HRM, the PEs can act as a specific function unit (FU) and operate at data-driven mode to improve performance while reducing power dissipation.

The data memory of the PE array is of distributed shared structure. Each PE has its own data memory with size of 512 B. In order to achieve the large memory size and low accessing latency, the RAMs in a PE cluster are shared logically through the column and row buses as shown in Fig. 3. In order to show it clearly, Fig. 3 only presents the memory shared structure of 4 PEs. As for the intra-cluster data access, a high-speed switching structure is realized, which can meet 16-channel data parallel read/write access.

5. Prototype development and experimental setup

This section describes the architecture modeling software platform, RTL synthesis configuration, and FPGA prototype system.

5.1. Architecture modeling

In order to evaluate the feasibility of the proposed reconfigurable mechanism, we developed an architecture model using SystemC[28], which describes the behavioral model of the 1024 PE array. The reconfigurable interface is developed with Qt[29], which communicates the reconfiguration contexts and video sequence/graphics data between the user and PE array. The user can type the operation into each PE and assemble it into binary automatically.

5.2. Design of HRM

According to the above analysis, the key to realize fast and dynamic reconstruction of reconfigurable array is HRM. Therefore, in this part, we further explain the hardware architecture of HRM in combination with Section 4 of HRM design ideas and methods. According to Fig. 1, the hardware architecture consists of two parts: a global controller and an H-type reconfiguration network.

Global controller: It is used to complete management of configuration information, monitoring of PE status, and data interaction with the host. The configuration information management module includes two parts: a configuration storage management module and a configuration network address generation module. The configuration storage manager generates a corresponding address according to the request sent by the host, accesses the corresponding configuration storage, and configures the network address generation module. The network address generator configures the addressing node according to the requirements. The data interaction on the host side mainly receives commands from the upper layer host and provides real-time feedback status information of the PE array.

H-type reconfiguration network: This network completes the delivery of configuration information and the collection of status information. In Fig. 1, each node in the H-type has two configurable registers, one for directional and one for mask. The directional controls the transmission in a single direction on the node, with left ‘0’ and right ‘1’. The mask controls the simultaneous transmission in both directions on the node. The mask control has a higher priority than the direction control. The entire data path is established by configuring these two registers on each node to achieve unicast, multicast and broadcast of the array. As shown in Fig. 1(a) (which consists of 8 × 8 PEs, inferring 6-bit addressing bus), after configuration the unicast mode is adopted, and the configuration information is sent to the destination PE through Node1_1, Node2_1, Node3_1, Node4_1, Node5_2, and Node6_3 in sequential order. Since the addressing of the PE (filled with red) takes six turns of right, right, left, left, right, and left, the address should be 110010. The multicast or broadcast mode is shown in Fig. 1(b). The configuration information arrives at Node2_1 and Node2_2 at the same time. The subsequent configuration process is similar to the unicast mode. When transfer a single configuration to a PE, the address and operation code are transferred in one pass[6, 30]; When a series of configuration information required by a PE, we use the transfer method with link establishment, data transfer, and link termination. During the data transfer, the following configuration information does not need to have address information. The network is also responsible for collecting data from PE/PEs to the host for performance evaluation.

Table 1. PE cluster RTL implementation.

Component	Parameter	This paper	RSF
PE	Bit-width of regs in a PE	16	16
	# of regs in a PE	16	4
	# of PEs	16	16
CM(4 kB)/IM(1 kB)	Bit-width of a CM/IM	32	32
	# of CMs/IMs	16	16
CB(1.5 kB)/DRAM(512 B)	# of sets	1	2
	# of banks in a set	1	3
	Bit-width of a bank	16	32

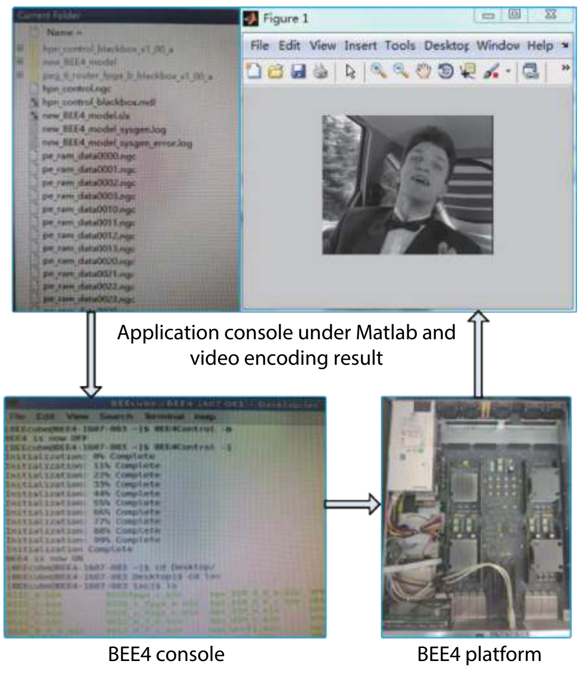
Figure 1
File Edit View Insert Tools Desktop Window Help
Application console under Matlab and video encoding result
BEE4 console
BEE4 platform

```
BEEcube@BEE4-1607-003 ~]$ BEE4Control -0
BEE4 is now OFF
BEEcube@BEE4-1607-003 ~]$ BEE4Control -1
Initialization: 0% Complete
Initialization: 11% Complete
Initialization: 22% Complete
Initialization: 33% Complete
Initialization: 44% Complete
Initialization: 55% Complete
Initialization: 66% Complete
Initialization: 77% Complete
Initialization: 88% Complete
Initialization: 99% Complete
Initialization Complete
BEE4 is now ON
BEEcube@BEE4-1607-003 ~]$ cd Desktop/
BEEcube@BEE4-1607-003 Desktop]$ cd [illegible]
BEEcube@BEE4-1607-003 [illegible]]$ ls
```

Fig. 4. (Color online) The prototype system of the proposed approach.

5.3. Implementation of prototype system

We developed the hardware of proposed HRM and also the PE array using Verilog HDL. The circuit specification of the proposed architecture and the RSF architecture is shown in Table 1. The prototype system of the proposed approach is implemented in the BEE4 FPGA development board of BeeCube with the FPGA chip of XC6VLX550T, as shown in Fig. 4, which takes video processing as an example. The top left part shows the console on host machine, where the user can input the video sequence and reconfiguration contexts. The bottom left part shows the console of the BEE4, where user can configure the FPGA and enable it to execute. The bottom right part is BEE4 platform, where the proposed HRM and PE array are mapped into and running on. The top right part is the reconstruction image, where the user can observe the results intuitively. The prototype system can work at speed of up to 150 MHz, and the HRM can work at 440 MHz, which means that the proposed reconfigurable mechanism won’t be the bottleneck of the whole system. In Fig. 4, the program running on each PE is stored by default, so it is the power-on configuration of the PE array. In the next part, we will describe the reconfiguration process.

6. Evaluation

In this section, we examine the reconfigurable process

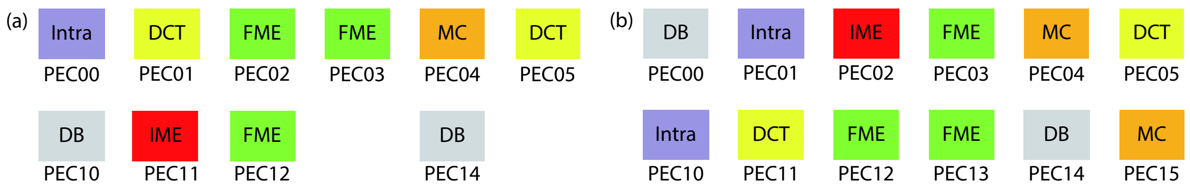

Fig. 5. (Color online) Organizations of video processing system. (a) Original organization of video processing system. (b) Video processing system with higher parallelism.

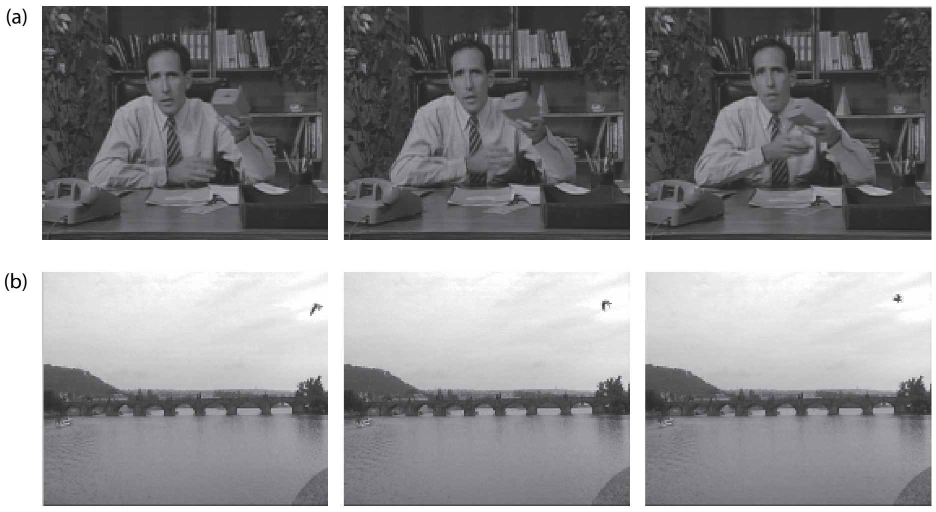

Fig. 6. Results of different sequences in video processing. (a) Salesman. (b) Bridge.

Table 2. The statistics of reconfiguration instructions.

FU changing	PE	Operation instruction	Call instruction
DCT→Intra	16→2	2856	8
DB→Intra	15→8	2087	9
Intra→DB	9→12	2584	8
IME→DCT	11→16	3136	7
FME→IME	2→11	39	16
FME	2	0	2
MC	5	0	5

on the prototype platform. We check the whole system structure reorganization on video processing, graphics processing, the data-driven mode operation to achieve high performance and low power consumption, and then the comparison of reconfigurable process.

6.1. Application in video processing

Under the proposed HRM and associated PE array, we implemented one kind of video processing system, as shown in Fig. 5(a). While testing, we found that the FUs, such as Intra and MC, are the bottlenecks. Then we reorganized it to Fig. 5(b) in run-time. In Fig. 5, each module with different color presents different FU, and each FU occupies a PE cluster.

In order to realize this reconfiguration, the contexts should be transferred are shown in Table 2. As shown in Fig. 5, when the array is initialized, PEC01 performs DCT algorithm and occupies 16 PEs. After reconfiguration, the cluster performs Intra prediction algorithm and occupies 2 PEs. We use 16→2 to indicate this change, so are the 15→8, 9→12, etc. The operation instructions can be transferred during the execution of last reconfiguration, so the reorganization of PE array only spend the time of call instructions' transmission, which means the reconfiguration time is 55 clock cycles. In HReA, the contexts' transmission is finished through router. And technically the PE's reconfiguration will spend at least 2 cycles for each call-like context, i.e., it will spend 2× more time for reconfiguration. Fig. 6 shows the video processing results under two sequences.

Meanwhile, we compared the power consumption of the new configuration of video processing with and without the power control. If left the unused PEs running in idle, the dynamic power dissipation is 286 mW; while in the HRM verison, we leverage the power control signal to disable the unnecessary PEs in this context, and the dynamic power dissipation is 139 mW. The power consumption reduced about 51%.

6.2. Application in graphics processing

We also apply the proposed HRM and associated PE array to graphics processing. Here we take a scene of viewport transform as an example. The scheme is mapped the graphics processing algorithm on 1 PE cluster. By reconfiguring the function of PE, the viewport transformation can be flexibly realized with lower computation cost[6]. Fig. 7 shows the overall mapping process of viewport transformation. The final output is consistent with the results derived from the software VS2013 platform, proving the correctly implementation of graphics processing algorithm on the proposed HRM and PE array. Fig. 8 shows the output from the prototype system, where the two mouse marks are used for select and feedback of OpenGL.

6.3. Realization of data-driven mode

Data-driven mode is used for the data intensive computation. We select the six-tap filter in fractional motion estima-

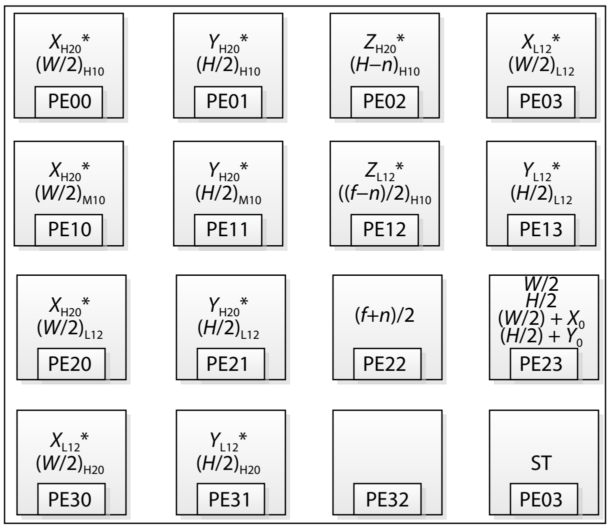

Fig. 7. Mapping of viewport transportation.

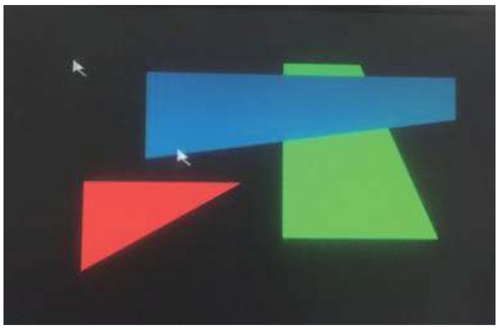

Fig. 8. (Color online) Rendering scene by proposed scheme.

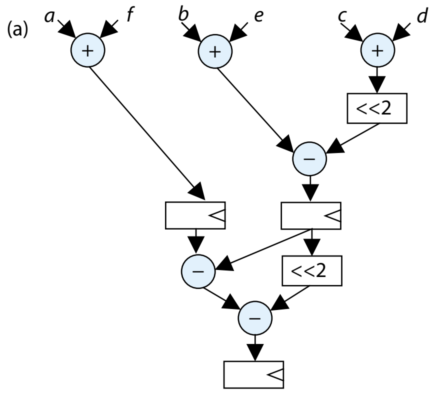
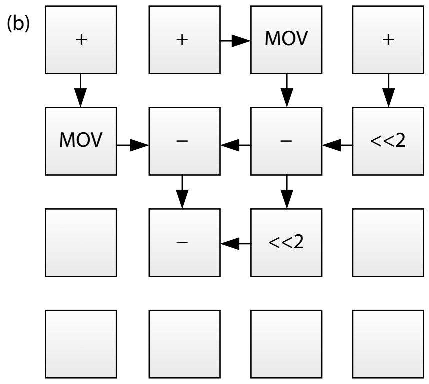

Fig. 9. Data-driven mode mapping of six-tap filter. (a) Six-tap filter. (b) Data-driven mode mapping.

tion (FME) algorithm of video encoding. FME performs motion search around the refinement center pointed to by integer motion estimation (IMV) and further refines the integer motion vectors (IMVs) into fractional MVs (FMVs) of quarter-pixel precision. FME interpolates half-pixels using a six-tap filter. The six-tap filter was expressed and realized in ASIC structure as shown in Fig. 9(a). In order to realize it in proposed structure, we map it on the PE array as shown in Fig. 9(b). All the configuration contexts are transferred from global controller through HRM, and each associated PE latch its own configuration information. Then the PE array works in this configuration. In such scenario, the working speed can reach 214 MHz in this FPGA. And if we utilize Verilog HDL to describe the six-tap filter directly and implemented it on the same FPGA, its operating frequency is 500 MHz. The results show that the performance of proposed reconfigurable structure is comparable to the ASIC circuit.

6.4. RTL-synthesis results

In order to evaluate the performance of the proposed architecture and the HRM, we synthesize the circuit using design compiler under the process of 90 nm CMOS technology. The results of area, critical delay, and power dissipation are shown in Table 3. The synthesis results show that the proposed approach has some advantages in speed, and area of 25.1%, 36.9%, respectively when compared with RSF, however, the power dissipation is higher, but still very low, about 1.96 mW. When compared with HReA, the speed advantage is about 21.6%, however, the HReA is implemented under the 65 nm CMOS technology. The HRM maximum operating frequency can reach 494.16 MHz. The instruction issuing operation is completed in one cycle, the instruction broadcasting operation is completed in 2 cycles, and the data feedback operation is completed in one cycle. Hence, we can predict that HRM will have a comparable speedup under the same technology. From Table 3, we can observe that the HRM only occupies 0.1% area of one PE cluster. The area cost can be negligible.

Table 3. Synthesis results of PE cluster.

Comparison aspect	PE cluster				HRM
	This paper	RSF	HReA	Comparison	
Process (nm)	90	90	65	—	90
Area (gate equivalent)	4155696	6586491	—	36.9%↓	4331
Critical delay (ns)	2.80	3.74	3.57	25.1%↓, 21.6%↓	2.02
Power (mW)	1.96	0.8	—	145%↑	0.48

7. Conclusion

Prior reconfigurable architectures suffer from performance bottleneck due to the inadequate flexibility and low context switching speed. This paper proposes a fast reconfigurable mechanism —HRM and applies it in a massive homogeneous PE array to achieve the simultaneous movement of reconfiguration contexts among particular/related PEs in unicast, multicast, or broadcast modes and also control the power. We develop the HRM architecture model using SystemC, implement the HRM design in Verilog, simulate with Questasim, synthesize using Design Compiler, and implement on a FPGA board. Compared to the RSF and HReA, HRM can speedup critical delay by an average of 23.4%, and improve the context/algorithm switching performance by 2× while saving 36.9% area. In the context switching, HRM can shut down the unnecessary PEs which reduced 51% of dynamic power dissipation in one video processing configuration. Moreover, in the data-driven mode, the proposed PE array with HRM can achieve performance comparable to ASICs.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 61834005, 61602377, 61772417, 61802304, 61874087) and the Shaanxi International Science and Technology Cooperation Program No. 2018KW-006, Shaanxi Provincial Key R&D Plan under Grant No. 2017GY-060, and Shaanxi Province Co-ordination Innovation Project of Science and Technology under Grant No. 2016KTZDGY02-04-02.

References

[1] Yun Z, Jiang L, Wang S, et al. Design of reconfigurable array processor for multimedia application. Multimed Tools Appl, 2018, 77(3), 3639
[2] Shi X, Luo X, Liang J, et al. Frog: Asynchronous graph processing on GPU with hybrid coloring model. IEEE Trans Knowl Data Eng, 2017, 30(1), 29
[3] Wang Y, Davidson A, Pan Y, et al. Gunrock: A high-performance graph processing library on the GPU. ACM SIGPLAN Notices, 2016, 51(8), 11
[4] Cao S, Zhang C, Yao Z, et al. Efficient and effective sparse LSTM on FPGA with bank-balanced sparsity. Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2019, 63
[5] Wu E, Zhang X, Berman D, et al. Compute-efficient neural-network acceleration. Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2019, 191
[6] Tian R J, Jiang L, Deng J Y, et al. Design and implementation of reconfigurable viewport transformation unit in embedded GPU. Mini-Micro Syst, 2018, 39(05), 1074
[7] Vestias M P. High-performance reconfigurable computing. In: Advanced Methodologies and Technologies in Network Architecture, Mobile Computing, and Data Analytics. IGI Global, 2019, 731
[8] Yao P, Zheng L, Liao X, et al. An efficient graph accelerator with parallel data conflict management. Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques, 2018, 8
[9] Yang C, Wang Y, Wang X, et al. WRA: A 2.2-to-6.3 TOPS highly unified dynamically reconfigurable accelerator using a novel Winograd decomposition algorithm for convolutional neural networks. IEEE Trans Circuits Syst I, 2019, 66(9), 3480
[10] Liu L, Li Z, Yang C, et al. Hrea: An energy-efficient embedded dynamically reconfigurable fabric for 13-dwarfs processing. IEEE Trans Circuits Syst II, 2017, 65(3), 381
[11] Jafri S M A H, Daneshtalab M, Abbas N, et al. Transmap: Transformation based remapping and parallelism for high utilization and energy efficiency in CGRAs. IEEE Trans Comput, 2016, 65(11), 3456
[12] Karunaratne M, Mohite A K, Mitra T, et al. Hycube: A CGRA with reconfigurable single-cycle multi-hop interconnect. 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), 2017, 1
[13] Kim Y, Joo H, Yoon S. Inter-coarse-grained reconfigurable architecture reconfiguration technique for efficient pipelining of kernel-stream on coarse-grained reconfigurable architecture-based multi-core architecture. IET Circuits, Devices Syst, 2016, 10(4), 251
[14] Tajammul M A, Jafri S M A H, Hemani A, et al. Private configuration environments (PCE) for efficient reconfiguration, in CGRAs. 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors, 2013, 227
[15] Jiang L, Deng J Y, Song S, et al. HRM: H-tree based reconfiguration mechanism in homogeneous PE array for video processing. Poster in the 55th Annual Design Automation Conference (DAC'18), 2018
[16] Huang J, Raabe A, Buckl C, et al. A workflow for runtime adaptive task allocation on heterogeneous mpsocs. 2011 Design, Automation & Test in Europe, 2011, 1
[17] Jafri S M A H, Tajammul M A, Hemani A, et al. Energy-aware-task-parallelism for efficient dynamic voltage, and frequency scaling, in CGRAs. 2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2013, 104
[18] Kirischian L. Reconfigurable computing systems engineering: virtualization of computing architecture. CRC Press, 2017
[19] Wei S J, Liu L B, Yin S Y. Reconfigurable computing. Science Press, 2014 (in Chinese)
[20] Wang Y S, Liu L B, Yin S Y, et al. Hierarchical representation of on-chip context to reduce reconfiguration time and implementation area for coarse-grained reconfigurable architecture. Sci Chin Inform Sci, 2013, 56(11), 1
[21] Kim Y, Mahapatra R N. Dynamic context compression for low-power coarse-grained reconfigurable architecture. IEEE Trans Very Large Scale Integr Syst, 2009, 18(1), 15
[22] Venkat A, Tullsen D M. Harnessing ISA diversity: Design of a heterogeneous-ISA chip multiprocessor. ACM SIGARCH Comput Architect News, 2014, 42(3), 121
[23] Hu C. Why FinFET and what next. Keynote in Shanghai Tech Workshop on Emerging Devices, Circuits and Systems, 2016
[24] Deng J Y, Li T, Jiang L, et al. Design and optimization for multiprocessor interactive GPU. The Journal of China Universities of Posts and Telecommunications, 2014, 21(3), 85
[25] Deng J Y, Li T, Jiang L, et al. Design and implementation of the graphics accelerator oriented to OpenGL. Journal of Xidian University, 2015, 42(6), 124
[26] Deng J Y, Li T, Jiang L, et al. The design of multiprocessor interactive GPU MIGPU-9. J Comput Aid Des Comput Graph, 2014, 26(9), 1468
[27] Shen X B, Liu Z X, Wang R, et al. The unified model of computer architectures. Chin J Computs, 2007, 30(5), 729
[28] Black D C, Donovan J, Bunton B, et al. SystemC: From the ground up. Springer Science & Business Media, 2009
[29] Eng L Z. Qt5 C++GUI programming cookbook. Packt Publishing Ltd, 2016
[30] Zhang X T, Jiang L, Deng J Y, et al. Design and Implementation of global controller in reconfigurable video array processor. Microelectron Comput, 2017, 34(11), 75