doi:10.3772/j.issn.1006-6748.2023.01.004

Design and implementation of instruction-driven and data-driven self-reconfigurable cell array①

SHAN Rui(山 蕊)②*, XIA Xinyuan*, YANG Kun**, CUI Xinyue*, LIAO Wang*, GAO Xu*
(*School of Electronic Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, P. R. China)
(**School of Safety Science and Engineering, Xi'an University of Science and Technology, Xi'an 710054, P. R. China)

Abstract

The reconfigurable chip, which integrates the advantages of high performance, high flexibility, high parallelism, low power consumption, and low cost, has achieved rapid development and wide application. Generally, the control part and the computing part of algorithm is accelerated based on different reconfigurable architectures, but it is difficult to obtain overall performance improvement. For improving efficiency of reconfigurable structure both for the control part and the computing part, a hybrid of instruction-driven and data-driven self-reconfigurable cell array is proposed. On instruction-driven mode, processing element (PE) works like a reduced instruction set computer (RSIC) machine, which is mainly for the control part of algorithm. On data-driven mode, data is calculated by flowing between the preconfigured PEs, which is mainly for the computing of algorithm. For verifying the efficiency of architecture, some high-efficiency video coding (HEVC) video compression algorithms are implemented on the proposed architecture. The proposed architecture has been implemented on Xilinx FPGA Virtex UltraScale VU440 develop board. The same circuitry is able to run at 75 MHz. Compared with the architecture that only supports instruction-driven, the proposed architecture has better calculation efficiency.

Key words: cell array, configurable computing, data-driven, instruction-driven

0 Introduction

At present, with the rise of big data applications such as machine learning, voice and video recognition, and artificial intelligence, the demand for real-time and intelligent user experience continues to increase, leading to a surge in computing and a sharp increase in bandwidth demand[1]. At the same time, the diversity and variability of application computing forces the hardware architecture must be flexible and customizable. Therefore, the reconfigurable computing structure has become an inevitable choice[2]. The reconfigurable chip, which integrates the advantages of high performance, high flexibility, high parallelism, low power consumption, and low cost, has achieved rapid development and wide application[3].

Various reconfigurable structures have been studied. Ref.[4] proposed an X-CGRA structure PE in X-CGRA has the ability to switch between precise OMs and different approximations. Ref.[5] proposed a 4D-CGRA structure that supports mutually exclusive data flows to be mapped to the same set of resources, allowing appropriate data flows to be executed at runtime based on branching results. Ref.[6] proposed a reconfigurable and low-complexity accelerator on application specific integrated circuit (ASIC) for both convolutional neural network (CNN) and generative adversarial networks (GAN) and described the methodology to determine the parameters of design and optimize the dataflow to obtain maximum performance. These reconfigurable structures are suit able for dataflow applications. Reconfigurable PE array is mainly used to realize the computation. Data is transferred between PEs through multiple interconnection methods. Data injection and recovery is usually completed by a dedicated controller. This type of structure is not very friendly to control intensive applications. Ref.[7] proposed three novel design methods to enable reconfigurable architecture to efficiently execute the control kernel. Ref.[8] proposed TLIA structure, which combined TIA technology with parallel conditions to effectively deal with control intensive kernels. Ref.[9] proposed a Blocks

① Supported by the National Natural Science Foundation of China (No. 61802304, 61834005, 61772417, 61634004), the Shaanxi Province Key R&D Plan (No. 2021GY-029).
② To whom correspondence should be addressed. E-mail: shanrui0112@163.com.
Received on Dec. 30, 2021

structure, which completed data communication between functional units through two separate circuit switched networks in the array structure. One network is used to configure control information, and the other network is used to build data path. These reconfigurable structures are friendly to control intensive kernels. The PE array is not only used for computing, but also for taking charge of managing data. Most of PEs adopt a RISC-like architecture. However, this kind of architecture has lower computational efficiency for dataflow applications.

For improving efficiency of reconfigurable architecture both for the control part and the computing part. A hybrid of instruction-driven and data-driven self-reconfigurable cell array is proposed. On instruction-driven mode, PE works like a RSIC machine, which is mainly for the control part of application. On data-driven mode, data is calculated by flowing between PEs, which is preconfigured. Meanwhile, in order to achieve zero delay in data transmission between adjacent PEs and improve data transmission efficiency, a double-buffered data-driven interface is used, which works on data-driven mode using {valid, ready} handshake mechanism. Only when the data arrives, the operation is fired and results are sent to neighboring PEs, at the same time ready signals are sent to upper PE.

This paper is organized as follows. Section 1 briefly states related work and motivation. The architecture of reconfigurable data-driven and instruction-driven cell array will be presented in Section 2. Section 3 discusses dynamically self-reconfigurable mechanism in detail. The results of stimulation and performance analysis are reported in Section 4. In Section 5, a conclusion is given.

1 Related work and motivation

Reconfigurable computing is considered a promising structure for continuously innovating applications[10]. For video compression processing chip, reconfigurable computing structure has been widely used to speed up calculation and improve performance. Ref. [11] proposed a reconfigurable design based on the 898 approximations transform in order to allow the simultaneous computation of eight 4-, four 8-, two 16-, or one 32-point approximate discrete cosine transform (DCTs). Ref. [12] proposed an architecture for interpolation filters, which is able to trade quality for energy and power efficiency by exploiting approximate interpolation filters and by halving the amount of required memory with respect to state-of-the-art implementations. Ref. [13] proposed a low power near memory sum of absolute difference (SAD) accelerator for motion estimation (ME). The accelerator is composed of 64 modular SADs. PEs on a reconfigurable fabric offer maximal parallelism. The PE arrays in those structures are RISC-like processor architecture. The task of video compression algorithm is distributed to execute on multiple PEs in parallel. Data processing in single PE is serialized, which seriously affects the performance improvement.

At the same time, more and more reconfigurable structures for CNN calculations emerged. Ref. [14] implemented a generative network accelerator (GNA) based on intra-PE processing, inter-PE processing, and cross-layer scheduling techniques. Precision adaptive PEs and buffer bandwidth reconfiguration are used to support flexible bit widths for both inputs and weights in deep neural networks. Ref. [15] proposed a high energy efficient reconfigurable CNN accelerator with approximate computing named approximate computing based on reconfigurable architecture (ARA). Based on the approximate computing units, the convolution neural processing unit (CNPU) is proposed with the reconfigurable data path for mapping different tasks. Ref. [16] proposed DyHard-DNNs, where accelerator micro architectural parameters are dynamically reconfigured during deep neural network (DNN) execution to significantly improve metrics of interest. The PE arrays in those structures are mainly used to compute parallelly, a centralized controller attached it to realize the control of the calculation process. Data irrelevant calculation can be well parallelized. However, the control part can only be serialized.

In order to parallelize as much as possible to improve computational efficiency, the idea of this paper is that PE can follow the instruction flow driven mode, which is mainly used to realize the control part of algorithm. Unlike traditional centralized controller, it is more like a distributed controller with certain papalism and better flexibility.

It also can follow the dataflow driven, data processing like a dataflow graph with the adjacent interconnection between PEs. When the data arrives, the operation is fired. Meanwhile, in order to improve utilization of PEs, multiple configuration operations in one algorithm are supported in one PE. Among multiple configuration information, the operations are switched automatically once an operation finished. The detail switching process can be seen in subsection 3.3.

2 Instruction-driven and data-driven self-reconfigurable cell array

A hybrid of instruction-driven and data-driven structure is proposed, as shown in Fig. 1. It includes a

host interface, a global controller, and a PE array. The global controller is used to control and manage the computing resources of array. It is a key part to realize the self-reconfigurable mechanism.

The PE array connected by short interconnections is the core part of the entire system. Each PE can be set on two different work modes: instruction-driven and data-driven. When it works on instruction-driven mode, it is similar to the general-purpose processor, RISC-like structure is employed which consists of three stages, i. e., fetching, decoding, executing and writing back. When it works on data-driven mode, an adjacent interconnection interface is used to receive data to be sent by neighboring PEs, then the operation of PE is fired and the result is sent to neighboring PEs. On this mode, the operation remains unchanged for a long execution cycle (even up to thousands of execution cycles), which can reduce the bandwidth for loading configuration information and reduce overall power consumption. The instruction memory unit in PE is 512 × 44 bit, and the data memory unit is 512 × 16 bit. Meanwhile, twelve local registers and four shared registers (RE/R12, RS/R13, RW/R14, and RN/R15) are used.

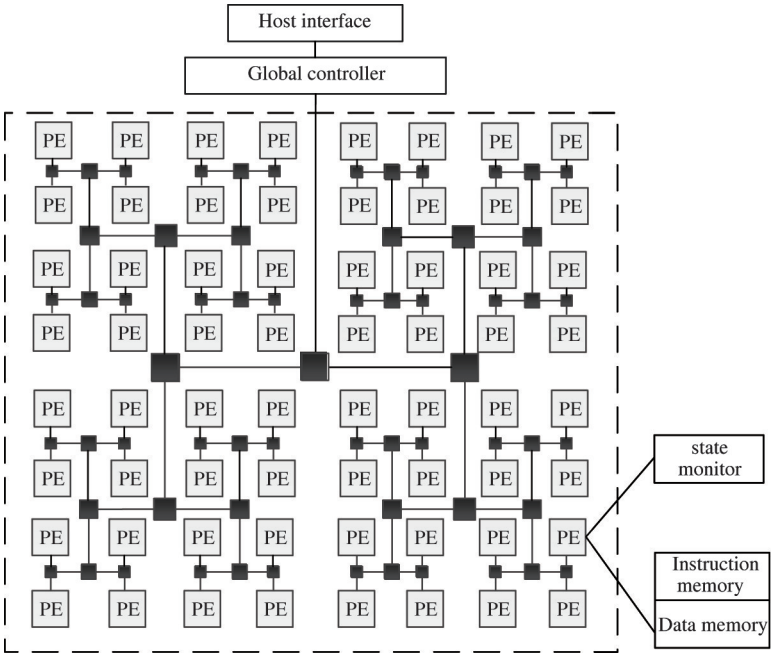


Fig. 1 Instruction-driven and data-driven self-reconfigurable cell array structure

An H-tree type hierarchical network between the host interface and the PE array is used to ensure that each instruction can reach PE in a shorter time, and realize the control and management for the array resources. When the host interface accesses the PE array, the global controller receives the information of bus from the host interface. The width of bus is 43 bits. The most 10 bits are used for address information, the flowing 3 bits are used for flag information, lower 30 bits are and used for command information. The flag information is used to determine whether to perform data feedback, instruction issuance, instruction multicast, or array boot, and the bit [30] is used to determine the delivery mode of destination PE. When it is reset, it means that the destination PE is on the instruction-driven mode. When it is set, it means that the destination PE is on the data-driven mode. Instruction information is used to determine the instructions executed by the PE.

The function of status monitor is to feed back the status of PEs in the array to the global controller in real time. If it is detected that PE is on free, the global controller receives the feedback information and sends the instructions to the free PE in real time through the H-tree type hierarchical network.

3 Self-reconfigurable PE

As the basic part of the self-reconfigurable array processor, PE is the basic unit of the self-reconfigurable system and the key operation unit of data processing. It can work on different functions according to different configuration information, which is related to the function realization and overall performance, and plays a vital role in the entire accelerator system. Therefore, a PE structure based on the hybrid of instruction-driven

and data-driven is proposed as shown in Fig. 2. It is similar to the general RISC structure and a three-stage pipeline structure is designed. The first level is to read configuration information from the configuration memory. The second level is used to receive data from the 4 input ports or 16 general registers. The third level is used to receive the data to be sent by the previous level and write the result back to the destination after performing operation.

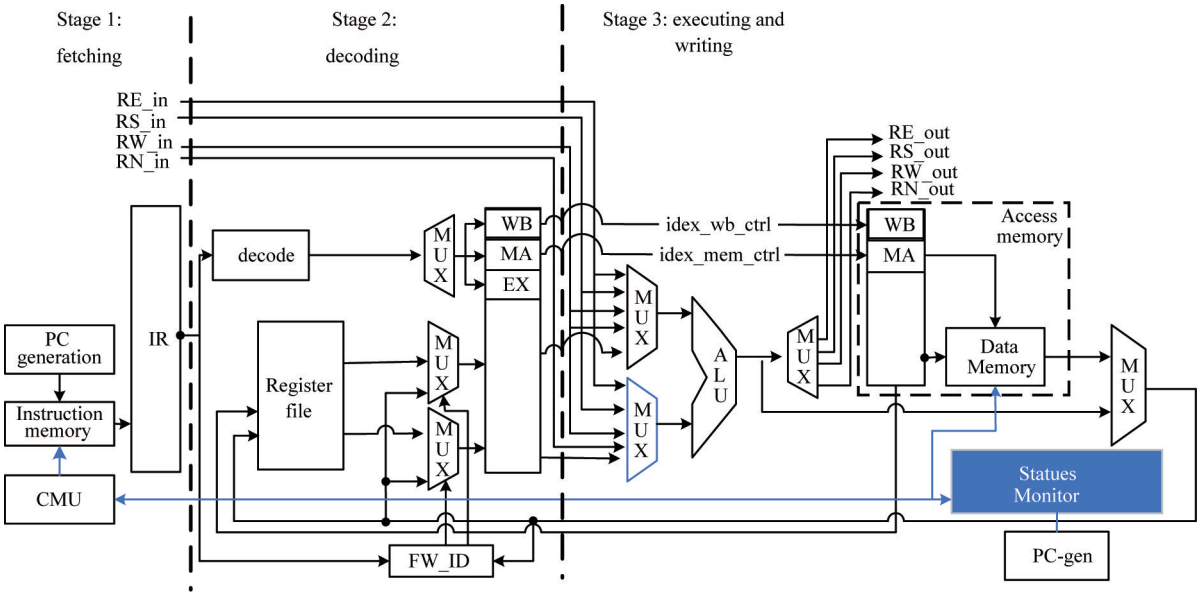


Fig. 2 PE structure

3.1 Instruction set

This section focuses on the research of the instruction set of the dual-mode dynamic self-reconfigurable PE. When the PE is configured on the instruction-driven mode, the configuration instruction memory size is 512 × 30 bits, and the data memory size is 512 × 16 bits. Each configuration instruction has to go through a three-stage pipeline of instruction fetching, decoding, executing and writing back. PE supports multiple operations such as logic and arithmetic operations, conditionally jumping immediately jumping, and load/store in the instruction-driven mode. The information of instruction includes 6 bits of opcode, 4 bits of destination register (RD), source register (RS), and target register (RT), and 16 bits of immediate (Imme) data. When the PE is configured on the data-driven mode, the size of configuration instruction memory is 16 × 30 bits, and the size of data memory size is 512 × 16 bits. For computationally intensive applications, after the controller issues the required configuration instructions, there is no need to repeat operations such as fetching and decoding, and directly to perform operation according to configuration information.

As shown in Fig. 3(a), when the PE is on the instruction-driven mode, the configuration instruction address is incremented by 1 by default, and when a branch or jump instruction is encountered, the configuration address is transferred to the specified address according to the state. When the PE is on the data-driven mode, only the data required by the current instruction arrives and triggers execution. Fig. 3(b) shows the instruction format of the 30-bit data transfer instruction. Fig. 3(c) shows the instruction format of the 30-bit jump instruction.

Opcode 1 bit+5 bits	RD 4 bits	RS 4 bits	RT/Imme 4 bits/16 bits

(a) 30 bits arithmetic operation shift instruction

Opcode 6 bits	RD 4 bits	RS 4 bits	

(b) 30 bits data transfer instructions

Opcode 6 bits	RD 4 bits	RS 4 bits	RT /Imme 4 bits /16 bits

(c) 30 bits jump instruction

Fig. 3 Instruction set type

3.2 Instruction flow operating mode

The data path of PE on the instruction-driven mode is shown in Fig. 4. The configuration information is sent into the configuration memory of the PE firstly through configuration information network based on H-tree type, then according to the update of PC, the corresponding configuration instruction is fetched from the

configuration memory and controls the operation of PE.

It is similar to the pipeline processing of general-purpose processors. The configuration information can be switched during execution, and the address of next configuration information is determined by the configuration information itself according to the execution situation. It can be executed sequentially, or executed according to the address generated by the jump instruction. On the instruction-driven mode, data also can be transferred by adjacent interconnection between PEs.

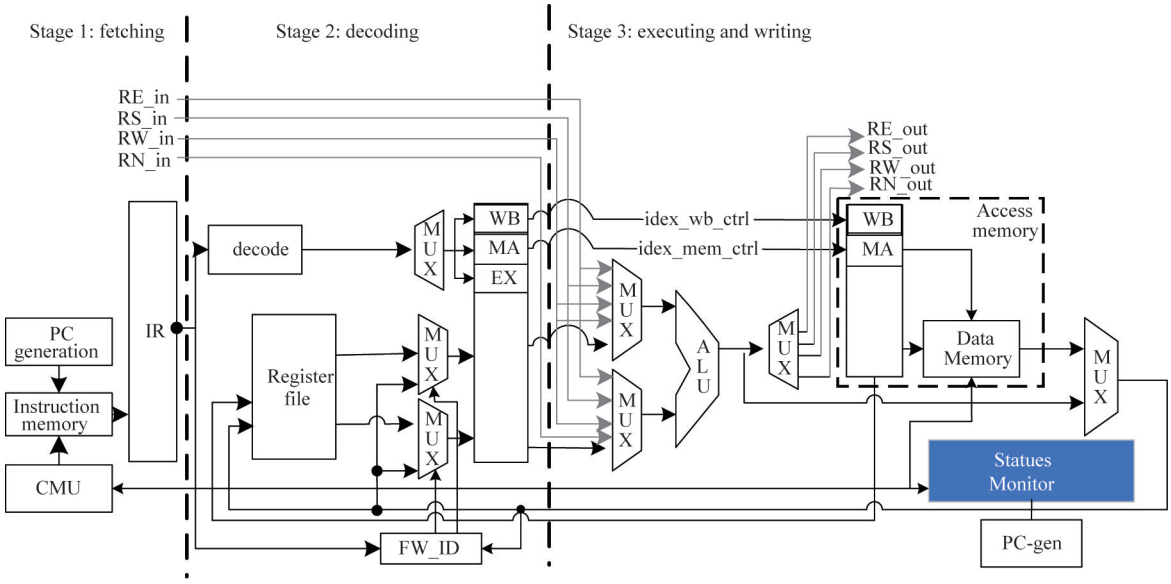


Fig. 4 The data-path of PE on the instruction-driven mode

3.3 Data flow operating mode

The data path of PE on the data-driven mode is shown in Fig. 5. No instruction fetching and decoding operations is required. The data from the upper PE is directly transferred to the execution unit and operating according to the configuration information. The execution of the operation is fired once the needed data arrives, and the result is delivered to the next level of PE. The operation will not be changed until the new configuration information arrives. It can reduce the bandwidth for loading configuration information and overall power consumption.

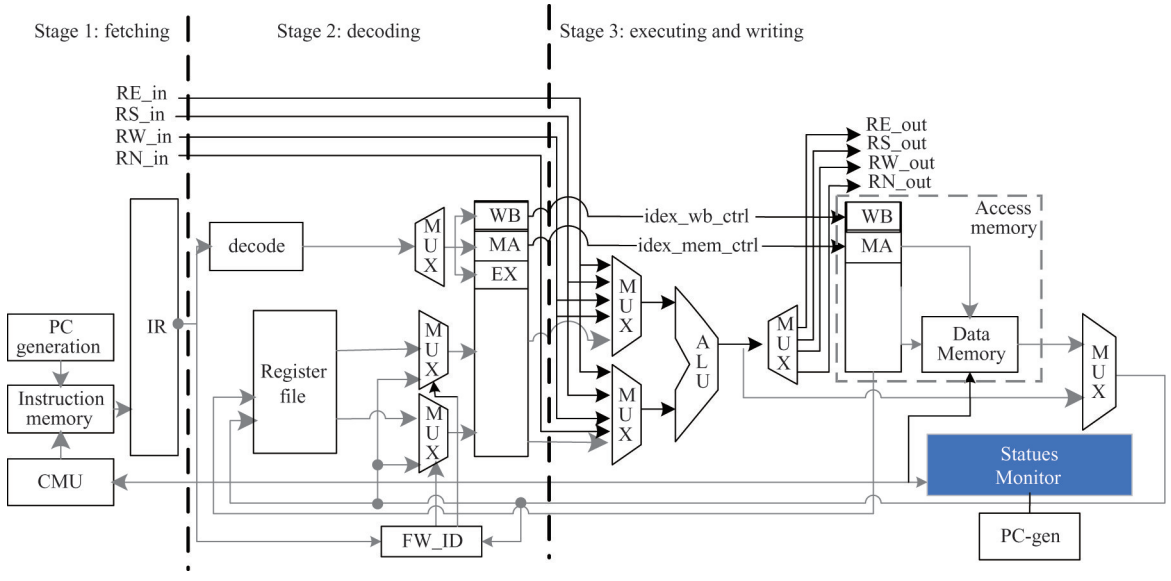


Fig. 5 The data-path of PE on the data-driven mode

First, take out the required configuration information from the configuration register. At the same time, the PC remains to be unchanged. When the required data arrives, the operation is executed and then the result is sent to neighboring PE. The operation type of PE will not be changed until a new configuration infor-

mation is received.

In order to improve computation efficiency, it is permitted that multiple nodes in the dataflow graph of algorithm are mapped on a single PE. The operation is triggered to execute and autonomously switched to the next operation only when the required operation data arrives. When the last operation of the current configuration is completed, it is switched to the first operation and repeated until a new configuration command is received. For a variety of applications with complex calculations, it can speed up the execution time in a relatively small array scale.

In order to reduce the delay of data transmission between adjacent PEs and improve data transmission efficiency, a double-buffered data-driven interface is used, which works on data-driven mode using {valid, ready} handshake mechanism. When at least one of the two internal buffers is empty, the data to be sent from the upper-level PE is received and a response signal is sent to the upper-level processing unit to indicate that the data has been received normally. When the two internal buffers are all occupied, the data cannot be received until at least one of the internal two buffers is empty. The two buffers take turns to receive the data from the upper-level PE, and take turns to send.

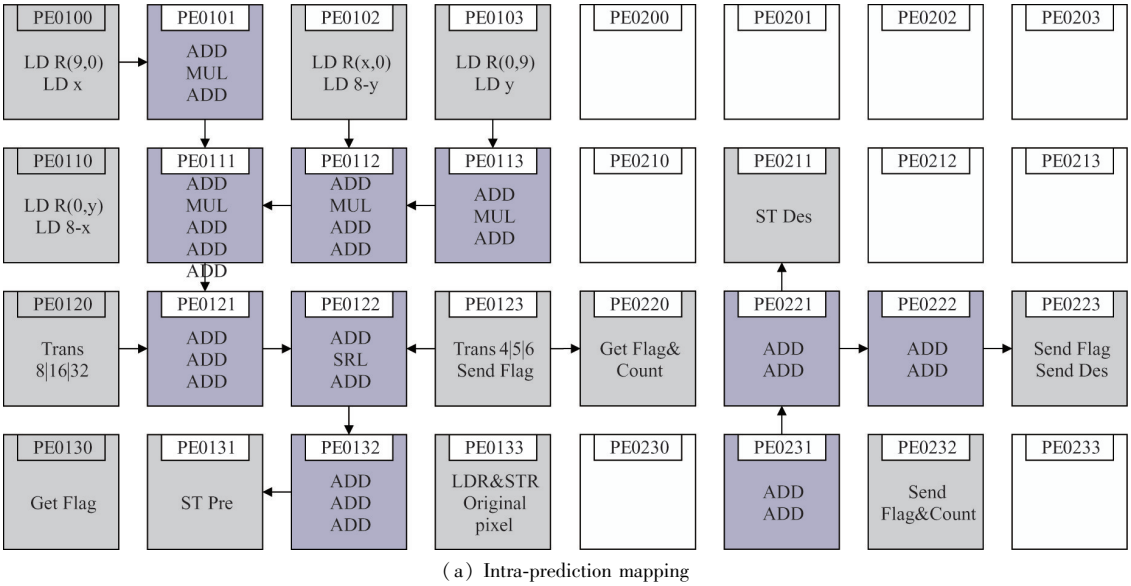
4 Implementation and performance

4.1 HEVC video compression algorithms mapping

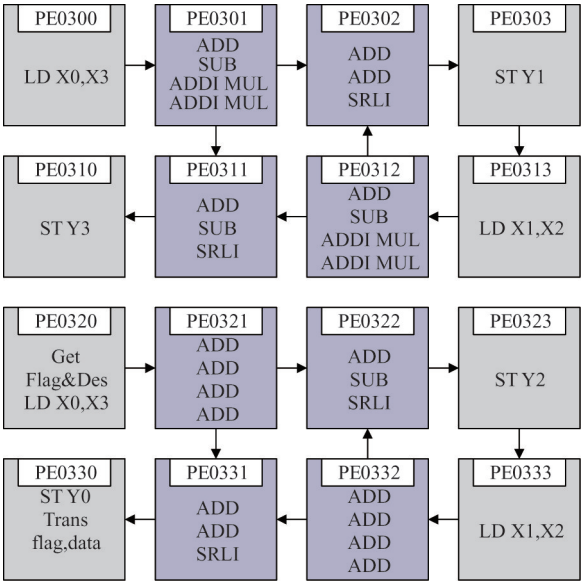
Some HEVC video compression algorithms are realized on the proposed structure. The mapping strategy can be seen in Fig. 6. The grey filled PEs work on the instruction-driven mode and the others work on the data-driven mode. For intra-prediction mapping, as shown in Fig. 6(a), total two clusters are used. PE0130 receives the flag bit of coding unit (CU) block division to be sent by PE0033. PE0133 remotely routes the original pixels in the DIM, and loads the reference pixels from PE0100, PE0102, PE0103 and PE0110 respectively. PE0131 stores the calculated prediction value in the local banks, and the other PEs are configured on the data-driven mode for calculating the prediction pixels. PE0220 receives the flag bit of CU block division to be sent by PE0123. PE0231 receives the predicted pixel, obtains the residual value, and writes the result to the local banks. PE0221, PE0222 and PE0223 are configured on the data-driven mode to transmit the residual value to the DCT module.

For DCT mapping, as shown in Fig. 6(b), only one cluster is used. PE0320 receives the residual value and the flag bit of CU block division and stores them in the local banks. PE0300, PE0313, PE0320 and PE0333 are working on the instruction-driven mode and load the required residual value. PE0303, PE0310, PE0323 and PE0330 are used to store the DCT calculation results. The rest PEs are configured on data-driven mode for calculation. Finally, PE0330 loads the results of DCT calculation and send the flags bits to the IDCT module.

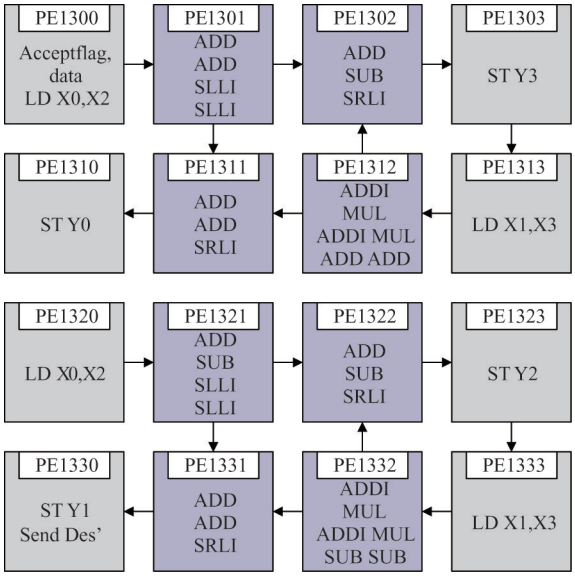
For IDCT mapping, as shown in Fig. 6(c), one cluster is used. PE1300 receives the result from the DCT calculation and the flag bit of CU block division from IDCT, and stores them in local banks. PE1300, PE1313, PE1320 and PE1333 are working on the instruction-driven mode and load the required data. PE1303, PE1310, PE1323 and PE1330 store the results



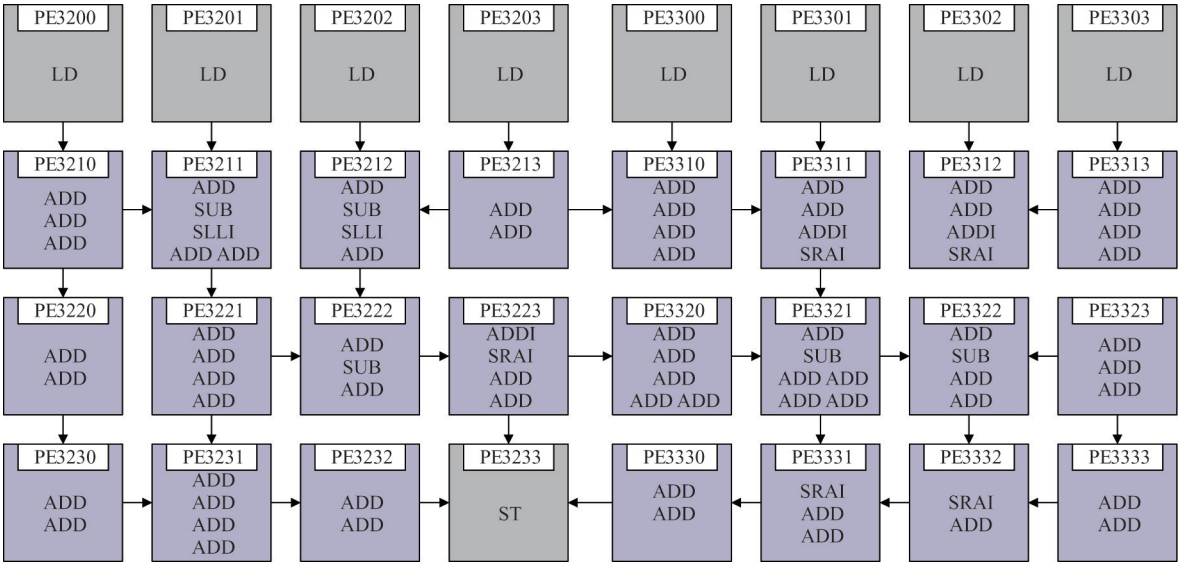
(a) Intra-prediction mapping



(b) DCT mapping



(c) Inversediscretecosine transform (IDCT) mapping



(d) deblocking filter mapping

Fig. 6 Some HEVC video compression algorithms mapping

of IDCT calculation. The rest PEs are configured on data-driven mode for calculation. Finally, PE1330 loads the results of IDCT calculation and sends the flag bit to the reconstruction module.

For deblocking filtering mapping, as shown in Fig. 6(d), two clusters are used. If filtering is required, PE3202 receives the pixel value of the 8×8 block boundary and stores it in a local bank. PE00, PE01, PE02 of PEG32 and PEG33, PE03 load the required pixels. All PEs except PE3233 are configured on the data-driven mode for filtering calculations. After the calculation is completed, the PE3233 stores the data, and then PE3202 sends the data to complete the replacement of the block boundary pixels.

A simple testbench is built to simulate the proposed architecture. Firstly, a translator is used to translate assembly to machine language. And then the test bench sends the machine language to destination PE through H-tree type hierarchical network.

The statistic of PE working mode can be seen in Table 1. The computing time for different algorithms can be seen in Table 2. For dataflow applications, such as deblocking filter, most PEs work on data-driven mode. For control intensive applications, such as intra-prediction, most PEs work on instruction-driven. Also, it can be seen that PEs working on instruction-driven take charge of accessing data. PEs working on data-driven take charge of computing.

Table 1 Statistic of PE working mode

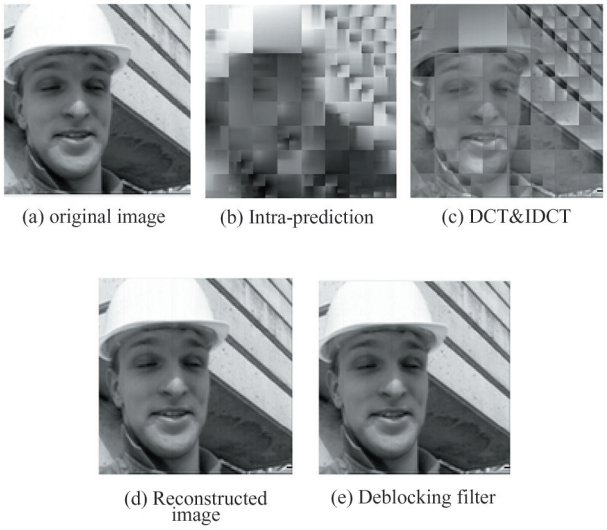
algorithm	# PEs on instruction-driven	# PEs on data-driven	#PEs	Percentage of PEs on data-driven
CTU	16	0	16	0%
Intra-prediction	13	10	23	43.5%
DCT	8	8	16	50%
IDCT	8	8	16	50%
Deblocking filter	13	23	36	63.9%

Table 2 Computing time (unit: cycles)

Algorithm	Image size	Computing time
CTU division	64×64	7228
Intra-prediction	8×8	620
	16×16	2642
	32×32	10 436
DCT	4×4/8×8	161/645
	16×16/32×32	2258/8497
IDCT	4×4/8×8	150/600
	16×16/32×32	2181/8505
Deblocking filter	64×64	19 084

4.2 Performance analysis

The computing time comparison with other structures is shown in Table 3, such as field programmable gate array (FPGA), central processing unit (CPU), ASIC and dynamic programmable reconfigurable array processor(DPRAP)[17]. DPRAP is similar to the proposed architecture, except that all PEs in it are only working on the instruction-driven mode. Compared with DPRAP, the proposed architecture can reach a speed-up of more than 5 times. Compared with FPGA, the computing time of the proposed architecture is close to that of FPGA. Compared with CPU, the proposed architecture is faster. However, it is lower than ASIC. For realized on ASIC, the computing speed is fastest, but it occupied most look-up-table (LUT) and register resources.



(a) original image (b) Intra-prediction (c) DCT&IDCT

(d) Reconstructed image (e) Deblocking filter

Fig. 7 The running results of some HEVC algorithms

Table 3 The comparison of computing time (unit: cycles)

algorithm	Image size	FPGA[18-20]	CPU[21-22]	ASIC[23-25]	DPRAP[17]	Proposed
Intra-prediction	8×8	4232	1300	47	2181	620
DCT	8×8	500	1130	64	7374	645
IDCT	8×8	-	1027	64	4882	600
Deblocking filter	8×8	128(32×32)	-	8	1388(16×16)	234

The proposed architecture has been implemented on Xilinx FPGAVirtexUltraScale VU440 chip. The synthesis result is shown in Table 4 for different scales. For 32×32 PE array, it can work at the frequency of over than 75 MHz, and occupies 272 200 register resources and 646 981 LUTs totally for 1024 PEs. The synthesis result of 1024 PEs and the result compared with other implementation architectures can be also shown in Table 5. Compared with DPRAP, resource occupation and frequency is relatively close. However, the proposed architecture is more functional and has higher calculation efficiency. Compared with Ref. [26], the proposed architecture has more PE resources in unit area. Compared with Ref. [27], the proposed architecture has more PEs and can support more operations per clock cycle. Meanwhile, compared with Ref. [17] and Ref. [26], in the same number of PEs, the proposed architecture occupies lower register and LUT resources.

Table 4 The synthesis result

Scale	PEs	Clock/MHz	LUT	FF	BRAM
4×4	16	100	25 445	16 180	16
8×8	64	100	103 400	65 479	64
32×32	1024	75	646 981	272 200	173

Table 5 Performance and resource usage

	DPRAP[17]	Ref. [26]	Ref. [27]	Proposed
Develop board	Vertix-6	ZC706	Vertix-6	Vertix UltraScale
Clock/MHz	120	100	150	75
LUT	33 662	232 252	134 520	646 981
FF	106 443	120 184	148 323	272 200
BRAM	-	-	139	173
PEs	16	64	128	1024
Power/W	-	-	5.056	6.368

5 Conclusion

In this paper, a hybrid of instruction-driven and data-driven self-reconfigurable cell array is proposed. For the computing part of algorithm, PE works on the data-driven mode through adjacent interconnection interface realizing data transfer between PEs. At the same time, PE can be fired only when the needed data arrives. Those PEs work like ASIC circuits, so the calculation efficiency can be promoted. For the control part of algorithm, PE works on the instruction-driven mode and is mainly used to realize the control of the calculation process and data access through programming. Flexible working methods effectively improve the performance.

Some HEVC video compression algorithms are realized on the proposed architecture. The computing time is statistic. Simultaneously, it has been implemented on Xilinx FPGA Virtex UltraScale VU440 develop board based on 1024 PEs. The circuitry can be run at 75 MHz and occupies 646 981 LUTs and 272 200 register resources.

References

[1] CEZE L, HILL M D, WENISCH T F. Arch2030: a vision of computer architecture research over the next 15 years[J]. Arch2030 Workshop, 2016:1-12.

[2] CHEN N, WANG Z, HE R, et al. Efficient scheduling mapping algorithm for row parallel coarse-grained reconfigurable architecture[J]. Tsinghua Science and Technology, 2021, 26(5): 724-735.

[3] LU Y, LIU L, ZHU J, et al. Architecture, challenges and applications of dynamic reconfigurable computing [J]. Journal of Semiconductors, 2020, 41(2): 021401.

[4] AKBARI O, KAMAL M, AFZALI-KUSHA A, et al. X-CGRA: an energy-efficient approximate coarse-grained reconfigurable architecture [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(10): 2558-2571.

[5] KARUNARATNE M, WIJERATHNE D, MITRA T, et al. 4D-CGRA: introducing branch dimension to spatio-temporal application mapping on CGRAS [C] // 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). Westminster: IEEE, 2019: 1-8.

[6] XU W, ZHANG Z, YOU X, et al. Reconfigurable and low-complexity accelerator for convolutional and generative networks over finite fields[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(12): 4894-4907.

[7] ZHU J, LIU L, YIN S, et al. A hybrid reconfigurable architecture and design methods aiming at control-intensive kernels[J]. IEEE Transactions on Very Large Scale Integration Systems, 2019, 23(9): 1700-1709.

[8] LIU L, WANG J, ZHU J, et al. TLIA: efficient reconfigurable architecture for control-intensive kernels with triggered-long-instructions[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 27(7): 1-1.

[9] WIJTVLIET M, J HUISKEN, WAEIJEN L, et al. Blocks: redesigning coarse grained reconfigurable architectures for energy efficiency[C] // 2019 29th International Conference on Field Programmable Logic and Applications (FPL). Barcelona: IEEE, 2019:17-23.

[10] LIU L, ZHU J, LI Z, et al. A survey of coarse-grained reconfigurable architecture and design: taxonomy, challenges, and applications[J]. ACM Computing Surveys (CSUR), 2019, 52(6): 1-39.

[11] JRIDI M, ALFALOU A, MEHER P K. Efficient approximate core transform and its reconfigurable architectures for HEVC[J]. Journal of Real-Time Image Processing, 2020, 17(2): 329-339.

[12] PREATTO S, GIANNINI A, VALENTE L, et al. Optimized VLSI architecture of HEVC fractional pixel interpolators with approximate computing[J]. Journal of Low Power Electronics and Applications, 2020, 10(3): 24.

[13] SUNDARAM J, SRINIVASA S R, KURIAN D, et al. A 93 TOPS/Watt near-memory reconfigurable SAD accelerator for HEVC/AV1/JEM encoding[C] // 2021 Design, Automation and Test in Europe Conference and Exhibition (DATE). Grenoble: IEEE, 2021: 1400-1403.

[14] YAN J, YIN S, TU F, et al. GNA: reconfigurable and efficient architecture for generative network acceleration [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(11): 2519-2529.

[15] GONG Y, LIU B, GE W, et al. ARA: cross-layer approximate computing framework based reconfigurable architecture for CNNs[J]. Microelectronics Journal, 2019, 87: 33-44.

[16] PUTIC M, VENKATARAMANI S, ELDRIDGE S, et al. DyHard-DNN: even more DNN acceleration with dynamic

hardware reconfiguration [C] // Proceedings of the 55th Annual Design Automation Conference. San Francisco: IEEE, 2018: 1-6.
[17] XIE X Y, DU Z L, HU C Z, et al. A parallelization method of inception architecture based on array processor [C] //2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). Auckland: IEEE, 2020: 92-99.
[18] ZHU Y, JIANG L, SHI P F, et al. Parallelization of intra prediction algorithm based on array processor [J]. High Technology Letters, 2019, 25(1): 74-80.
[19] CHEN M, ZHANG Y, LU C. Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms [J]. AEU-International Journal of Electronics and Communications, 2017, 73: 1-8.
[20] AYADI L A, BOUBAKRI W, LOUKIL H, et al. A hardware-efficient parallel architecture for HEVC deblocking filter [C] // 2019 16th International Multi-Conference on Systems, Signals & Devices (SSD). Istanbul: IEEE, 2019: 669-673.
[21] JIANG W, CHI Y, JIN H, et al. A fine-grained parallel intra prediction for HEVC based on GPU [C] // 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS). Wuhan: IEEE, 2016: 778-784.
[22] MASOUMI M, AHMADIFAR H R. Performance of HEVC discrete cosine and sine transforms on GPU using CUDA [C] //2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI). Tehran: IEEE, 2017: 857-861.
[23] AMISH F, BOURENNANE E B. Fully pipelined real time hardware solution for high efficiency video coding (HEVC) intra prediction [J]. Journal of Systems Architecture, 2016, 64: 133-147.
[24] SINGHADIA A, MAMILLAPALLI M, CHAKRABARTI I. Hardware-efficient 2D-DCT/IDCT architecture for portable HEVC-compliant devices [J]. IEEE Transactions on Consumer Electronics, 2020, 66(3): 203-212.
[25] BALDEV S, ANUMANDLA K K, PEESAPATI R. Scalable wavefront parallel streaming deblocking filter hardware for HEVC decoder [J]. IEEE Transactions on Consumer Electronics, 2019, 66(1): 41-50.
[26] SHAN R, DENG J, JIANG L, et al. Design of a clustered data-driven array processor for computer vision [J]. High Technology Letters, 2020, 26(4): 424-434.
[27] NIETO A, DAVID L. VILARIÑO, et al. PRECISION: a reconfigurable SIMD/MIMD coprocessor for computer vision systems-on-chip [J]. IEEE Transactions on Computers, 2016, 65(8):2548-2561.

SHAN Rui, born in 1986. She received her Ph. D degree from Xidian University in 2018, and also received her M. S. degree from Xi'an University of Posts & Telecommunications in 2011. She is an associate professor at Xi'an University of Posts & Telecommunications. Her research focuses on integrated circuit design.