IEICE Electronics Express

LETTER

IEICE Electronics Express, Vol.11, No.4, 1–8

A parallel arithmetic array for accelerating compute-intensive applications

Dong Wang[1a)], **Peng Cao**[2], **and Yang Xiao**[1]

[1] *Institute of Information Science, Beijing Jiaotong University*

[2] *National ASIC System Engineering Technology Research Center, Southeast University*

a) *wangdong@bjtu.edu.cn*

Abstract: A parallel arithmetic array processor for accelerating compute-intensive applications in low-power embedded systems is proposed in this study. The proposed flexible hardware architecture enables the fast execution of both control-dominated and compute-centric streaming computation tasks on the same array. Consequently, multiple levels of parallelism can be efficiently exploited. A test chip integrated with two 16×16 array processor cores was implemented in 65 nm CMOS technology. Multi-format video decoding algorithms were mapped on the chip as benchmarks. The proposed architecture achieved a notable 2.8× advantage on performance over an industrial coarse-grained array processor and a 66% performance boost over a state-of-the-art many-core processor. Meanwhile, the energy-efficiency was improved by 15.3× and 1.78×, respectively.

Keywords: arithmetic array, reconfigurable computing, multi-format video decoding

Classification: Electron devices, circuits, and systems

References

[1] H. Singh, M. H. Lee, G. Lu, F. J. Krdahi, N. Bagherzadeh and E. M. Filho: IEEE Trans. Comput. **49** [5] (2000) 465.

[2] J. Bae and J. Cho: IEICE Electron. Express **5** [8] (2008) 705.

[3] M. Ganesan, S. Singh, F. May and J. Becker: Proc. 17th Field Programmable Logic and Applications (2007) 467.

[4] F. Pescador, C. Sanz, M. J. Garrido, E. Juarez and D. Samper: IEEE Trans. Consum. Electron. **54** [2] (2008) 145.

[5] H. Xu, J. Tanabe, H. Usui, S. Hosoda, T. Sano, K. Yamamoto, T. Kodaka, N. Nonogaki, N. Ozaki and T. Miyamori: IEEE Symp. VLSI Circuits (2012) 150.

[6] Y. Ren, D. Wang and L. Liu: Proc. Asia-Pacific Signal and Information Processing Association Conf. (2011) 1.

[7] D. Wang, P. Ren and L. Liu: IEICE Electron. Express **10** [4] (2013) 1.

[8] D. Wang and M. D. Ercegovac: IEEE Trans. Comput. **61** [9] (2012) 1243.

[9] Y. Park, C. Yu, K. Lee, H. Kim, Y. Park, C. Kim, Y. Choi, J. Oh, C. Oh, G. Moon, S. Kim, H. Jang, J. Lee, C. Kim and S. Park: ISSCC Dig. Tech. Papers (2013) 160.

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014

IEICE Electronics Express, Vol.11, No.4, 1–8

[10] Audio Video Coding Standard (AVS): GB/T-200090.2 (2006) http://www.avs.org.cn.
[11] C. Yin, S. Yin, L. Liu and S. Wei: IEICE Trans. Electron. **E92-C** [10] (2009) 1284.

1 Introduction

The rapid evolution of portable, wireless devices today is pushing the boundaries of embedded microprocessor design and manufacturing technologies. Although silicon technologies would allow future designs to consume less power while delivering greater computational capacity and density, the traditional *von-Neumann* programmable architecture has already reached a performance and power limit known as the *power-wall*. Various new architectures are currently being studied in both academic and industrial fields. The reconfigurable array processor is a very promising architecture that can provide high energy efficiency of 10 MOPS/mW to 100 MOPS/mW [1] and notable software programmability.

A reconfigurable array processor consists of an array of homogeneous or heterogeneous processing elements (PEs). Unlike in traditional *von-Neumann*-like architectures (e.g., DSP processors) with limited number of arithmetic logic units (ALUs), the massive computational resources in array processor can exploit massive parallelism to accelerate the execution of the software program by dynamically adapting the hardware structure to meet application requirements. Moreover, reconfigurable processor can also reduce the control overhead for instruction decoding, sequencing and communication (e.g., data replication) between the computational blocks, thereby conserving the extra energy consumption. Therefore, array processors are particularly suited for boosting compute-intensive applications, such as video encoding/decoding, information security and big data processing in cloud environments. Take the widely studied multi-format video decoding [2] as an example. The compute-intensive tasks, including inverse transformation (IT), motion compensation (MC), intra-prediction (IP) and loop filtering (LF) may account for 60% to 80% of the total workload of the system (Fig. 1). Previous studies [3, 4, 5] have demonstrated that these tasks are mostly block-based word-level calculations that can potentially be processed in parallel. The design challenge lies in selecting the optimal hardware architecture that could efficiently exploit such parallelism. To address this issue, a flexible arithmetic array structure that executes both control-flow dominated and compute-centric streaming tasks on the same array is proposed in this study. This structure can achieve a significant speedup of target applications by utilizing multiple levels of parallelism.

2 PAAP architecture

The hardware architecture of the proposed parallel arithmetic array processor (PAAP) is illustrated in Fig. 2. PAAP consists of three major functional

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014

IEICE Electronics Express, Vol.11, No.4, 1–8

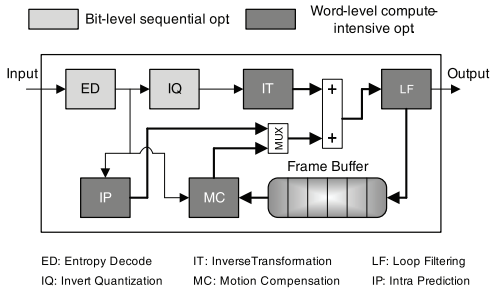

Fig. 1. A generic flow for multi-format video decoding.

parts, namely, computation arrays, configuration/control logic, and buffering memories. An array of $2M \times 2N$ arithmetic units (AUs) are organized in four $M \times N$ sub-arrays (SAs) at the center. Target applications are partitioned into multiple computation tasks and then mapped onto the SAs for parallel execution. Four 1024×256-bit multi-mode buffer memory blocks are associated with the SAs on the two sides. Each block can work independently either as a multi-port RAM or as a FIFO to support versatile memory access patterns. The operations of the computation and storage components are driven by the context (i.e., the configuration bit streams) fetched from the configuration interface.

In traditional reconfigurable array processors [1, 3], the streaming computations, which are characterized by predictable loop-based control flows with large iteration counts, large data sets, regular memory access patterns and high locality, are executed by a coarse-grained array. The irregular codes that contain a large amount of conditions and branches are mapped onto a RISC (Reduced Instruction Set Computer) or VLIW (Very long Instruction Word) processor. These types of architecture often result in frequent task switching and data transmitting between the two functional parts, thereby introducing extra workload and power consumption to the system.

In the proposed architecture, the AUs in each SA can be concatenated by a reduced-complexity intra-SA interconnection network to form one or several non-pipelined datapaths (Fig. 2). In each datapath, the output of one opera-

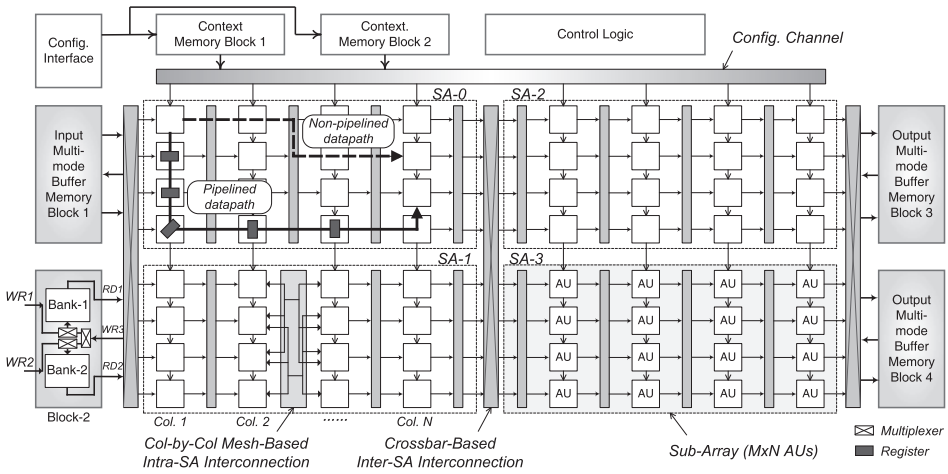

Fig. 2. Architecture of the proposed PAAP.

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014

IEICE Electronics Express, Vol.11, No.4, 1–8

tion is immediately fed to the input of the next operation in the chain. In this way, nested if-then-else statements can be executed in one clock cycle with high instruction-level parallelism (ILP). Through dynamic reconfiguration, AUs can also be pipelined to form a typical 2D array structure supporting streaming computation on the same SA. Massive data-level parallelism (DLP) can thus be utilized. When the SA is switched from control-flow dominated computation tasks to compute-centric stream processing tasks, the intermediate results can remain in the same buffer memory, thereby reducing the communication overhead and power consumption significantly. In general, N is usually selected to be between 2 and 6 to guarantee low latency on the non-pipelined data-path, whereas M can have a greater value (between 4 to 16) to enable higher data throughput.

On the top level, four SAs are designed to implement task-level parallelism (TLP). Sequential computation tasks, such as successively executing functions or loops that have data dependencies, are implemented as pipelined tasks by associating a buffer memory configured as a ping-pong FIFO between two SAs. Non-dependent parallel computation tasks are then mapped either on a shared SA or multiple SAs depending on the utilization of the memory and AU resources.

2.1 AU structure

Fig. 3 shows the internal structure of the AU. The fast arithmetic logic unit (FALU) performs the most fundamental operations (on 16-bit or 8-bit data), such as logic, comparison, barrel shift, and addition, which are covered by a typical RISC instruction set. Redundant arithmetic optimizations [6] are performed on the input operands and output results such that carry-propagate adders with long latencies are avoided in the middle of any non-pipelined datapath. The maximum number of FALUs that can be chained up is determined by the critical path delay. In the final implemented PAAP instance, this number is four.

The complex ALU (CALU) implements complicated operations required in domain-specific applications. As listed in Fig. 3, such operations include fixed-point multiplication (for video encoding/decoding and computer vi-

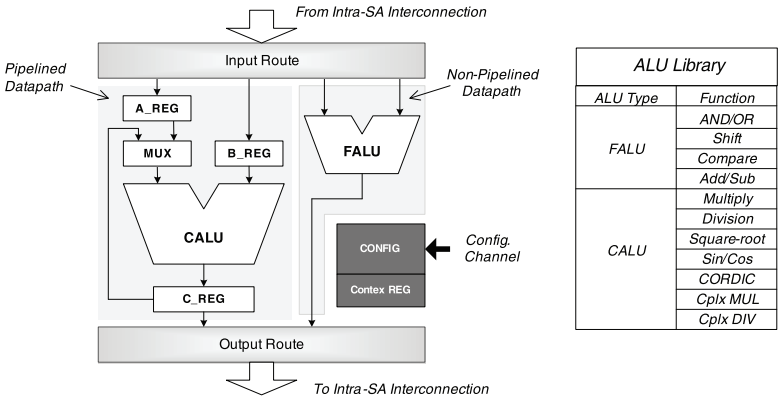

Fig. 3. The proposed AU architecture.

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014

sion), triangular functions (for information encryption), CORDIC (for software-defined radio) and complex number division/square-root (for multiple-input multiple-output antenna detection). The circuit designs of these operators were previously optimized for area and power consumption [6, 7, 8]. With these dedicated functions, PAAP can efficiently support a wider range of compute-intensive applications compared with previous designs [1, 3]. In the final design, 16×16 constant-correction truncated (CCT) multipliers [6] are implemented as CALUs in Col.2 AUs. Compared with using convention multipliers, using CCT multipliers reduce the SA area by 8%.

2.2 Reduced-complexity interconnection

In traditional coarse-grained array processor designs [1], 2D mesh-based interconnection networks are usually used to connect the PEs in a single large array. Although capable of enabling sophisticated and flexible routing strategies, 2D mesh interconnections may consume a large portion (usually from 20% to 40%) of the array area. The use of a column-to-column mesh interconnection structure in the PAAP architecture is thus proposed in this work. As shown in Fig. 2, each AU can be directly connected to two AUs within the same column and to three AUs in the adjacent columns with very short delays. These interconnections can be used to build the non-pipelined datapath that contains FALUs (Fig. 3). Multi-step mesh routes are only allowed to be used in pipelined datapaths. The width of each link is 32-bit, supporting 16-bit and 8-bit operands. Col.1 AUs are connected to Col.N AUs to support iterative operations. To guarantee architecture flexibility and efficiency, a 256-bit crossbar-based inter-SA connection network is designed to support high-throughput data exchanges between SAs and buffer memories. When a 16×16 PAAP is finally implemented, the results show that the proposed scheme reduces the interconnection area by a factor of 3.6 compared with using 2D full mesh interconnection scheme.

2.3 Multi-mode buffer memory structure

As shown in Fig. 2, each buffer memory block comprises two 512×256-bit two-port RAM banks. When configured in the double buffering mode (Fig. 4-a), different blocks of external data can be simultaneously written into Bank-1 through port $WR1$ and read from Bank-2 through $RD2$ to overlap the computation with data transfers. By dynamically switching the multiplexer circuit, the intermediate results are written back (usually by the same SA) into a reserved memory space of Bank-2. Consequently, each parallel computation task owns a dedicated 256-bit memory bus that can support a high data throughput with low memory access conflict rates. When configured in ping-pong FIFO mode (Fig. 4-b), the data of sequential computation tasks are alternatively written into or read from Bank-1 and Bank-2 by two SAs through the inter-SA connection networks. The proposed multi-mode buffer memory structure guarantees an efficient utilization of memory bandwidth and supports the flexible mapping of the target algorithm.

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014

IEICE Electronics Express, Vol.11, No.4, 1–8

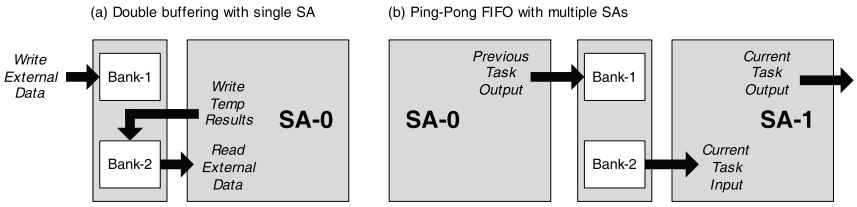

Fig. 4. Multi-mode buffering memory structure.

2.4 Algorithm mapping

Fig. 5 depicts the algorithm mapping schemes for H.264 decoding on the proposed architecture. Eight SAs (two PAAPs) are used to exploit multiple levels of parallelism to accelerate both compute-intensive (Fig. 5-a) and bit-level sequential tasks (Fig. 5-c). The efficient dataflow (Fig. 5-b) guarantees a very high data throughput and minimizes the energy consumption caused by frequent data replication or external memory access. Yi and Ui/Vi ($i = 1, 2, 3, 4$) refer to the four 8×8 luminance and chrominance sub-MBs in a 16×16 macro-block (MB), respectively. In general, one MB can be decoded in every 800 cycles, thereby achieving a decoding performance of 30 fps (frame per second) given the target working frequency of 200 MHz with 8160 MBs to be processed in one high-definition frame (i.e., $200 \times 10^6/(8160 \times 30)$). Algorithm partitioning (into parallel tasks) is performed by manually inserting directives in the source code, whereas task mapping, scheduling and memory allocation are automatically performed by a compiler tool [11] we developed.

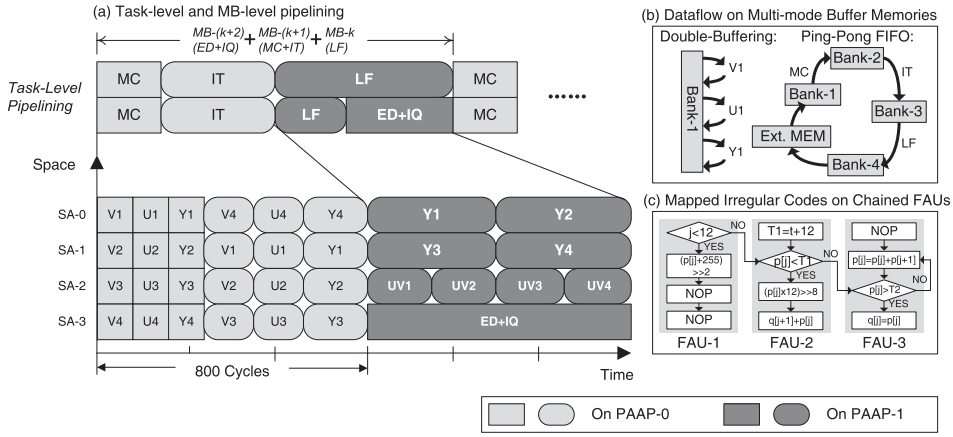

Fig. 5. Mapping H.264 decoding algorithm on the proposed architecture.

3 System integration and architecture

To fully test and verify the performance and energy efficiency of the proposed architecture, two 16×16 PAAPs are integrated into an embedded SoC (System-on-a-Chip) as shown in Fig. 6-a. Multi-format high-definition (HD) video decoding (H.264/MPEG-2/AVS [10]) is selected as the benchmark application because of its heavy computation-load. The two PAAPs are con-

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014

IEICE Electronics Express, Vol.11, No.4, 1–8

nected to the system bus through two 64-bit data and configuration busses. The 2D-DMA (Direct Memory Access) can provide fast 2D block-based data fetching and storing. The extra configuration controller is designed to recode the fetched contexts (because the PAAP configuration interface is 128-bit wide) and to synchronize the computation task mapped on the two PAAPs. The host ARM processor is only used for peripheral device control, video stream fetching and storing. A total of 54 encoded video streams (20 for H.264, 18 for MPEG-2 and 16 for AVS) are tested to obtain the accurate performance data as shown in Fig. 6-b.

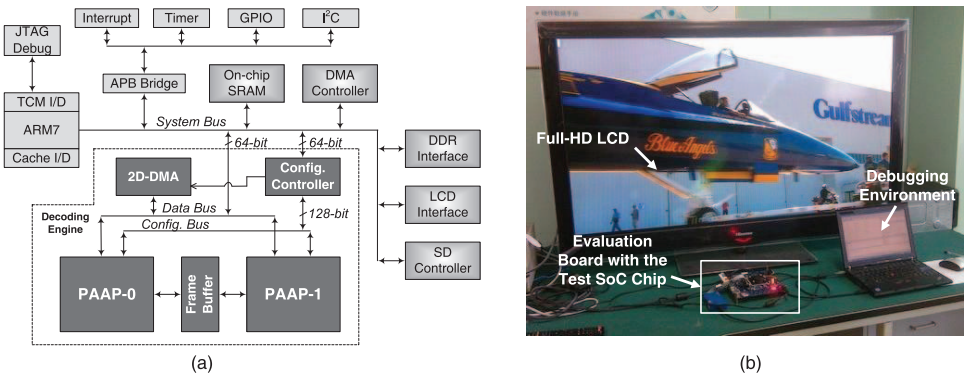

Fig. 6. (a) Hardware architecture of the test SoC chip, and (b) Testing environment.

4 Implementation results and comparisons

The proposed architecture (Fig. 6-a) is coded in Verilog HDL and implemented in a 9.28 mm ×9.28 mm die using TSMC 65 nm CMOS process. The typical working frequency is 200 MHz at 1.2 V. The equivalent gate count of one PAAP is 5.5 M, which accounts for 36% of total chip area. The performance and power consumption (measured on Agilent 93 K platform) of the fabricated SoC for HD H.264 decoding (because this format is the most complex one) are compared with those of four state-of-the-art designs (Table I). The reference designs include a coarse-grained array processor [3], a industrial DSP processor [4], a many-core processor (with two 32-core clusters) [5] and a dedicated hardwired multi-format video codec [9] (ASIC design). All these processors are domain-specific architectures developed to accelerate multi-format video encoding/decoding. The performance and power data are normalized to make a fair comparison.

Compared to the DSP processor, the proposed architecture possesses multiple processing units and very wide (256-bit) data paths. Therefore, performance growth can be achieved by efficiently exploiting TLP and DLP instead of scaling up the clock frequency, which has great impact on system power dissipation. As shown in Table I, the proposed architecture gains an 18× improvement on performance and consumes 85% less power than the DSP processor. When compared to the array processor of [3] and the many-core

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014

processor, the proposed special AU structure and multi-mode buffering memory can further increase the ILP of control-flow dominated computation tasks and the data throughput with low memory access overhead as discussed in Section 2.4. Consequently, a 1.8× and a 66% performance boosts are achieved by the proposed architecture, respectively. With regard to the energy efficiency, the proposed architecture outperforms the one with the best result by a factor of 1.78. The proposed architecture even has a close performance with the custom designed circuit [9] (only 17% slower). Although the energy efficiency is 1.8× lower, the proposed architecture has the advantage that it could always be reprogrammed to satisfy future application demands, such as the upcoming high efficiency video coding (HEVC) standard.

Table I. Performance and energy efficiency comparison with reference designs for HD H.264 decoding.

	[3]	[4]	[5]	[9]	Proposed
Technology (nm)	90	130	40	32	65
Area (mm^2)	75	529	210	N/A	48.9
Frequency (MHz)	450	600	333	166	200
Performance (fps)	24	25	30	30	30
Power (mW)	3420	1900	500	< 100	280
Normalized Performance (MBs/s/MHz)	435	67.5	735	1474	1224
Energy Efficiency (MBs/s/mW)	57	21.3	490	> 2448	874

5 Conclusion

An arithmetic array processor for accelerating compute-intensive applications is proposed in this study. With flexible array structure, reduced-complexity interconnection, and multi-mode buffer design, the proposed architecture can efficiently exploit TLP, DLP and ILP to speed up the execution of both control-flow dominated and compute-centric streaming tasks. Two 16×16 PAAPs are integrated into an SoC to implement multi-format video decoding algorithms (H.264/MPEG-2/AVS) as benchmarks. Measurements of performance and power data on the fabricated chip show that the proposed design achieves 1.8× faster speed and consumes 14.3× less energy than a typical industrial coarse-grained array processor. The proposed architecture also has a 1.66× advantage on performance and a 1.78× advantage on energy efficiency over a state-of-the-art many-core processor. By selecting the proper CALU functions, the proposed array processor can be used to enhance the performances of low-power embedded systems for various compute-intensive applications.

Acknowledgments

This work was supported by the National Natural Science Foundation of China Grant No. 61106022.

DOI: 10.1587/elex.11.20130981
Received December 12, 2013
Accepted January 06, 2014
Publicized January 31, 2014
Copyedited February 25, 2014