PAPER Special Section on VLSI Design and CAD Algorithms

Area-Efficient Reconfigurable Architecture for Media Processing

Yukio MITSUYAMA[†a)], ***Member***, **Kazuma TAKAHASHI**[†], **Rintaro IMAI**[†], ***Nonmembers***, **Masanori HASHIMOTO**[†], **Takao ONOYE**[†], ***Members***, ***and*** **Isao SHIRAKAWA**[††], ***Fellow, Honorary Member***

SUMMARY An area-efficient dynamically reconfigurable architecture is proposed, which is dedicated to media processing. To implement a compact but high performance device, which can be used in consumer applications, the reconfigurable architecture distinctively performs 8-bit operations required for media processing whereas fine-grained operations are executed with the cooperation of a host processor. A heterogeneous reconfigurable array is composed of four types of cells, for which configuration data size is reduced by focusing application domain on media processing. Implementation results show that a multi-standard video decoding can be achieved by the proposed reconfigurable architecture with $1.1 \times 1.4\,\mathrm{mm}^2$ in a 90 nm CMOS technology.

key words: *reconfigurable, media processing, multi-standard, area-efficiency, dynamic reconfiguration*

1. Introduction

In recent years, various video compression algorithms are used in a variety of consumer applications, and for each application, e.g. video-phone, video recording, video streaming, and digital TV broadcasting, an appropriate algorithm must be selectively employed. Therefore, there are increasing demands for consumer devices to support more than one video compression algorithm.

Owing to their specialized organization, ASICs offer high performance, small area, and low power consumption, but lack flexibility and incur high development costs. On the other hand, microprocessors offer the maximum flexibility since many of applications can be programmed as softwares, but suffer from lower performance and higher power consumption.

To address these problems, various reconfigurable architecture have been proposed [1] aiming at high performance, high flexibility, and low power consumption. However, the conventional reconfigurable architectures usually suffer from large hardware cost [2] to be used in consumer audiovisual applications. On the other hand, there are some approaches of reconfigurable architecture [3], [4] achieving lower hardware cost than ordinary commercial FPGAs.

Motivated by this tendency, the present paper proposes a novel architecture of reconfigurable device dedicated to media processing. In order to achieve high performance and area-efficient implementation of multi-standard video decoding, our reconfigurable architecture is designated to perform distinctive operations of media processing. Since media processing incurs many of 8-bit and 16-bit operations, 8-bit coarse-grained reconfigurable architecture is employed. In order to execute calculation processes of video decoding efficiently, the reconfigurable architecture adopts a heterogeneous array structure composed of four type of cells, i.e. basic cell with an ALU and a shifter, multiplication cell with a multiplier, register cell with a 16×8-bit register file, and memory cell with a 256×8-bit memory. As for complicated and conditional operations including header analysis, address calculation, and decoder status control, an embedded host processor is cooperatively used.

Moreover, it is distinctive that our reconfigurable architecture achieves significant reduction of configuration data size and reconfiguration overhead owing to the specialized function cell organization for media processing. As a result, dynamic reconfiguration can be performed although the configuration data is stored in an on-chip memory, which is on the outside of reconfigurable array. Specifically, the proposed reconfigurable architecture can change its configuration in 32 cycles, and therefore achieving dynamic reconfiguration in nano-second order. This exclusion of configuration memory to outside of reconfigurable array contributes much for area efficiency.

VLSI implementation results claim that the proposed architecture attains 3.5 times as high performance as conventional reconfigurable architectures and the multi-standard video decoder based on the reconfigurable cell array occupies only $1.1 \times 1.4\,\mathrm{mm}^2$ in 90 nm CMOS technology.

The rest of the paper is organized as follows. Section 2 explains architecture exploration of domain specific reconfigurable device. Detailed architecture of function cells, array organization, and dynamic reconfiguration are described in Sect. 3. VLSI implementation results are shown in Sect. 4. Section 5 describes related works of heterogeneous coarse-grained reconfigurable devices, and Sect. 6 concludes this work.

Manuscript received March 24, 2008.
Manuscript revised July 3, 2008.

[†]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.
[††]The author is with the Graduate School of Applied Informatics, University of Hyogo, Kobe-shi, 650-0044 Japan.
a) E-mail: mituyama@ist.osaka-u.ac.jp

DOI: 10.1093/ietfec/e91–a.12.3651

2. Architecture Exploration of Domain Specific Reconfigurable Device

2.1 Essential Evaluation Items for Reconfigurable Devices

In order to construct an efficient reconfigurable architecture, firstly it is necessary to consider the following items [5].

1. Cooperation with host processor:
   It must be decided for any reconfigurable device that target applications are achieved only by the device or by the cooperation with a host processor. Generally, when employing coarse-grained architecture the cooperation works effectively.
2. Granularity and functionality of reconfigurable cells:
   In accordance with the characteristics of target applications, the granularity and the functionality of reconfigurable cells should be carefully devised.
3. Reconfiguration mechanism:
   When enabling dynamic reconfiguration aiming at higher area-efficiency by time-multiplexed processing, the reconfiguration mechanism with less time and/or area overhead is indispensable.

Referring to the above evaluation items, the reconfigurable system architecture is constructed as shown in Fig. 1, which consists mainly of a reconfigurable array, a host processor, a configuration memory, and a data memory. The reconfigurable array is composed of several kinds of programmable cells and their programmable interconnects.

As shown in Fig. 1, configuration data and initialization data of the reconfigurable array are stored in the on-chip configuration memory and the data memory, respectively. On the configuration process of reconfigurable array, the configuration data and the initialization data are provided by these memories. User data for the reconfigurable array such as bitstream and frame data is transferred directly from the external memory to memory/register cells in the reconfigurable array. In addition, there are control data signals and data bus between the host processor and reconfigurable array. The host processor and the reconfigurable array can hence communicate with each other during runtime. Detailed interface architecture of the reconfigurable array is described in Sect. 3.9.

Each point for an area-efficient reconfigurable device is discussed in the followings.

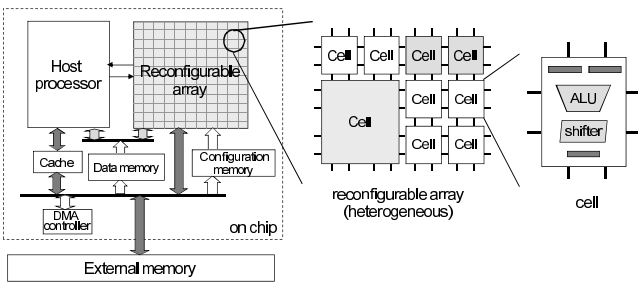


Fig. 1 Overall architecture.

2.2 Cooperation with Host Processor

There is a trade-off between the granularity of operation and the performance of reconfigurable device. A fine-grained reconfigurable architecture [5], [6] has a high flexibility to implement various applications. On the other hand, a coarse-grained reconfigurable architecture achieves high area efficiency if the granularity of reconfigurable device is suitable for the application.

Since major part of video applications has signal processing aspect, we adopt the coarse-grained architecture. However, complicated and conditional operations are also observed such as header analysis, address calculation for frame reference, and decoder status control. Therefore, the proposed reconfigurable architecture enables efficient cooperation with a host processor, where the reconfigurable array and the host processor are embedded in a single chip and work simultaneously by sharing the external memory.

2.3 Granularity and Functionality of Reconfigurable Cells

An area-efficient reconfigurable device must have suitable granularity and functionality for targeted applications [5], [6]. By reviewing processing features of standard video applications such as MPEG-2 decoder, MPEG-4 decoder, and H.263 decoder, requirements for reconfigurable cells are derived.

Variable length decoder (VLD) performs the decoding of a codeword into a quantized DCT coefficient, which is a series of codeword length detection, symbol decoding, bitstream shifting, and next codeword loading. Symbol decoding and bitstream shifting can be implemented by memories and shifters, respectively. Codeword length detection is to be done based on several small tables, whose word length is about 6 bits.

Inverse quantizer (IQ) performs the multiplication of a quantized DCT coefficient by quantization step size as well as saturation and IDCT mismatch control. Usually, addresses of inverse zig-zag scanning are stored in a memory. Calculation accuracies for address resolution and multiplication/saturation control are 6–8-bit and 8–24-bit, respectively.

An 8×8 two-dimensional inverse discrete cosine transformer (IDCT) is generally employed in various video applications. First, 8×1 1-D IDCT of each row is calculated, and then 8×1 1-D DCT of each column is calculated. Instead of using a transposition memory, addresses for access pattern of intermediate data are stored in a memory. The required bit width depends on the calculation accuracy, in general 16–28-bit is required.

Motion compensator (MC) performs motion vector decoding, half-pel manipulation, and frequent accessing to the frame memory, including picture reference and calculation result write-back. The required bit width is about 10-bit for calculating the average of pixels in the half-pel manipulator,

and about 24-bit for calculating frame memory addresses.

As mentioned above, the multi-standard video decoder, which is the targeted application domain of our reconfigurable architecture, consists mainly of 8–32-bit operations, particularly 8–16-bit operations. Thus the granularity of our reconfigurable device is set to 8 bits. In order to implement multibyte operations, neighboring cells are used simultaneously.

As for functionality of reconfigurable cells, our architecture adopts a heterogeneous structure of four types of cells, i.e. basic cell with an ALU and a shifter, multiplication cell with a multiplier, register cell with a 16×8-bit register file, and memory cell with a 256×8-bit memory. Since application domain is specific in our reconfigurable device, this heterogeneity can be supposed to be more effective. The ratio of four types of cells and the detailed array structure are determined to be suited for the mapping of multi-standard decoding.

2.4 Dynamic Reconfigurability

Dynamic reconfigurability of devices has also a fundamental trade-off between area and performance. An excessive numbers of reconfiguration during operations leads to the decrease in performance due to reconfiguration overhead. Therefore, in order to achieve effective architecture it is very important to determine the appropriate dynamic reconfigurability in consideration of the reconfiguration overhead.

Some of the conventional dynamically reconfigurable devices [7], [8] have a multi-context architecture and can perform the runtime reconfiguration without any overhead. However, a set of configuration data and required multiplexers for context control occupy considerable area in reconfigurable cells. As a result, area-efficiency of the device is decreased in this style.

Since reconfiguration overhead and hardware costs are increased relative to the size of configuration data, every effort must be made for the reduction of configuration data size aiming at higher area-efficiency. Each of function cells in our reconfigurable architecture is specialized to handle calculations in media processing applications so that configuration data in each cell can also be reduced to required minimum. Therefore, our reconfigurable architecture can perform dynamic reconfiguration with low overhead, where configuration data can even be stored on the outside of the array.

As an example of reconfiguration overhead, when suppressing the overhead to 5% of whole decode process, the runtime reconfiguration has to be performed in about 30 cycles at the clock frequency of 100 MHz, in case a macroblock decoding is executed with four contexts.

3. Details of Reconfigurable Architecture

3.1 Basic Structure of Reconfigurable Cell

Figure 2 shows basic structure of our reconfigurable cell.

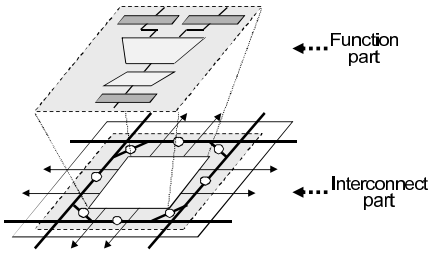


Fig. 2 Basic structure of reconfigurable cell.

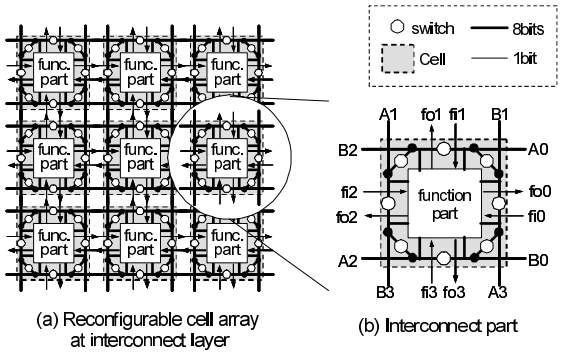


Fig. 3 Interconnect part of reconfigurable cell.

Each of the cells consists of an interconnect part, which has almost the same structure for all kinds of cells, and a function part, that has a different structure for each kind. Detailed organization of the interconnect part and the function part in each cell are described in what follows.

3.2 Interconnection

When providing fully flexible interconnection architecture, the serious increase of the hardware cost is inevitable. In particular, as in conventional fine grained reconfigurable devices, such as FPGAs, the interconnect part accounts for about 90% of chip area [9]. The interconnect architecture has a major trade-off between flexibility and hardware cost. In order to achieve high flexibility, such as a crossbar switch or a switching matrix should be adopted with a large amount of hardware cost. Therefore, as shown in Fig. 3(a), the interconnect part of the proposed reconfigurable cell only connects adjacent cells so as to reduce hardware cost of reconfigurable device. Besides, this interconnection also acts as a data feedthrough between adjacent cells. Accordingly, two cells, which are located at a certain distance, can even exchange data. As a general purpose reconfigurable device, the above mentioned limitation on connection capability may incur fatal loss in cell utilization. To avoid this problem, we employ heterogeneous array structure, which can eliminate long distance paths when executing video applications (see Sect. 3.7).

As shown in Fig. 3(b), in our architecture the interconnect part of reconfigurable cell is composed of eight 8-bit word lines and eight switches. Eight switches of word lines can change the connection of each two word lines. Four

Table 1 Configuration data for interconnect part.

#bit	Unit	Configuration
8	Status of eight switches	connect or disconnect
4	Direction of four switches	A0→B2 or A0←B2, A1→B3 or A1←B3, A2→B0 or A2←B0, A3→B1 or A3←B1

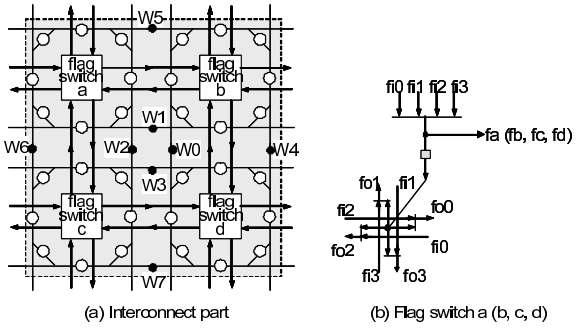


Fig. 4 Interconnect part of memory cell.

Table 2 Configuration data for interconnect part.

#bit	Unit	Configuration
32	Status of thirty-two switches	connect or disconnect
16	Direction of sixteen switches	A0→B2 or A0←B2, A1→B3 or A1←B3, A2→B0 or A2←B0, A3→B1 or A3←B1

switches of them, which connect the word lines of A0-B2, A1-B3, A2-B0, and A3-B1, can configure the direction of signals. As shown in Table 1, the interconnection of word lines are defined by 12-bit configuration data.

On the other hand, since memory cell has a large register file, memory cell has 4 times as large area as the other cells as is explained below. As shown in Fig. 4, the interconnect part of memory cell is composed of four identical components, while each component has the same structure as the interconnect part of other types of cells. Eight inputs to memory cell are W0–W7 in Fig. 4. The interconnect part of memory cell has 4 times as large number of configuration bits as the one of other cells. As shown in Table 2, the interconnection of word lines of memory cell are defined by 48-bit configuration data.

3.3 Basic Cell

Figure 5 shows the organization of the function part of basic cell, which is composed of two input registers, an ALU, a shifter, and an output register.

Word inputs A and B can be selected from A0–A3 and B0–B3, respectively. On the other hand, word output line can be connected to A0–A3 and B0–B3.

Flag inputs e_i and f_i can be selected from f_{i0}–f_{i3}. On the other hand, flag output f_o can be selected from the carry output or the sign from the ALU or the carry output from the shifter. The flag output can also be selected from the flag

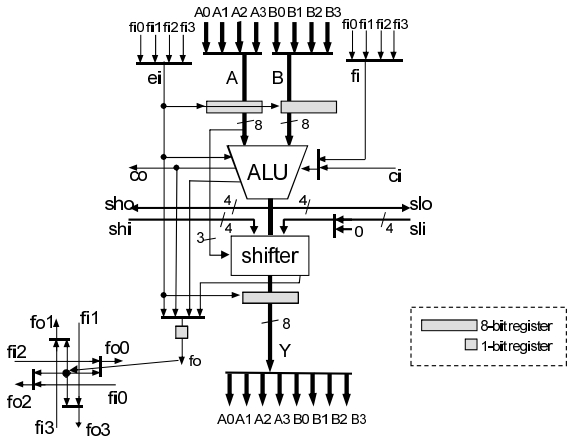


Fig. 5 Organization of function part of basic cell.

inputs f_{i0}–f_{i3} so as to enable a flag signal to be transferred to a distant cell. Furthermore, 1-bit pipeline register for a flag output can be used to insert 1 cycle delay.

Two input registers and an output register can be defined as pipeline registers, registers with write-enable signal e_i, constants, or unused registers.

The ALU can execute logic operations, addition and subtraction. When the ALU performs logic operations, the functions can be defined as AND, OR, and MUX with the selector of f_i. On the other hand, when the ALU performs addition or subtraction, its function can be defined by either of the configuration data, the flag inputs e_i and f_i.

The shifter can be defined as a fixed shift or a variable shift. The shift width of a fixed shifter can be defined as −4–+3 by the configuration data. On the other hand, the shift width of a variable shifter can be defined by the lower 3-bit of input word line A.

A basic cell can cooperate with the neighboring basic cells so as to perform a multibyte operation. The left side basic cell outputs the most significant byte, and the right side basic cell outputs the least significant byte. Carry signals c_o and c_i are the carry output to neighboring cell and the carry input from neighboring cell, respectively. Carry input to the ALU is selected from f_i or c_i. Besides, dedicated inputs (sh_i and sl_i) and outputs (sh_o and sl_o) from/to neighboring cells are equipped for multibyte shifting by concatenation of basic cells.

Configuration data for basic cell is summarized in Table 3.

3.4 Multiplication Cell

Figure 6 shows the organization of the function part of multiplication cell. Being composed of two input registers and a multiplier, multiplication cell can execute 8 × 8-bit multiplication.

In the same manner as basic cell, the word inputs A and B can be selected from A0–A3 and B0–B3, respectively. The output lines of the multiplier can be connected to A0–A3 and B0–B3.

Table 3 Configuration data for basic cell.

#bit	Unit	Configuration
6	Registers	pipeline, write enable, constants, unused
4	ALU	addition with carry f_i or c_i, subtraction, logic operation (AND, OR, or MUX), shifter type (extended or cooperative)
3	Shifter	shift width of −4–+3 bits
4	Flag output	output from f_{o0}–f_{o3}
4	Flag input	input to f_i and e_i
4	Word line input	input to port A and B
4	Word line output	selection of output line or Hi-Z

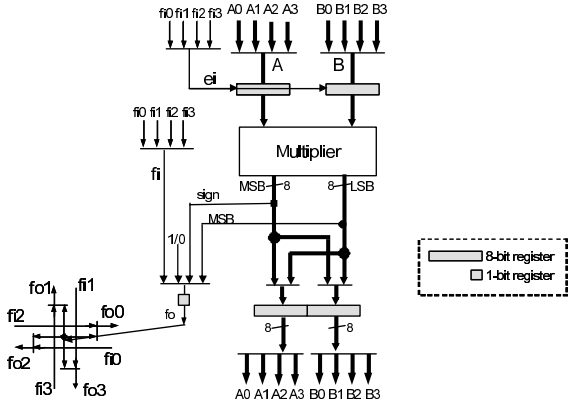


Fig. 6 Organization of function part of multiplication cell.

Table 4 Configuration data for multiplication cell.

#bit	Unit	Configuration
5	Registers	pipeline, write enable, constant, unused
2	Multiplier	Sign of word line input
4	Flag output	output from f_{o0}–f_{o3}
4	Flag input	input to f_i and e_i
4	Word line input	input to port A and B
6	Word line output	selection of output line or Hi-Z

Two input registers can be defined as pipeline registers, registers with write-enable signal e_i, constants, or unused registers. In order to implement the signed multiplier, the signs of input A and B can be defined by the configuration data. On the other hand, an output register can be defined as a pipeline register, or an unused register.

Flag inputs e_i and f_i can be selected from f_{i0}–f_{i3}. On the other hand, flag output f_o can be selected from the sign of the multiplication result or the MSB of the least significant byte of the multiplication result. Same as basic cell, flag output can also be selected from the flag input f_{i0}–f_{i3} in order to transfer a flag signal to a distant cell. Furthermore, 1-bit pipeline register for a flag output can be used to insert 1 cycle delay.

Configuration data for multiplication cell is summarized in Table 4.

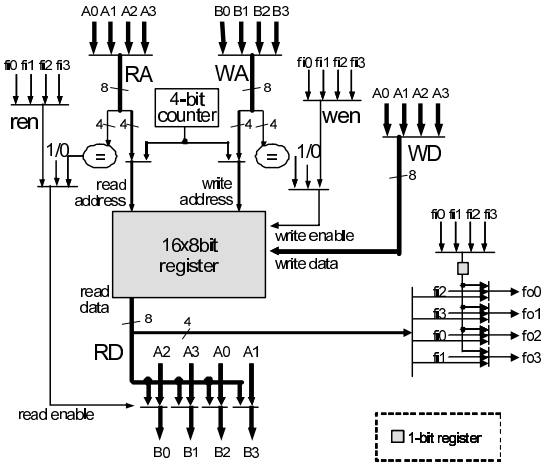


Fig. 7 Organization of function part of register cell.

3.5 Register Cell

Figure 7 shows the organization of the function part of register cell. Register cell is composed of a 16 × 8-bit register file and a 4-bit counter.

For data writing operation to the register file, the write address WA and the write data WD and can be selected from B0–B3 and A0–A3, respectively. On the other hand, for data reading operation from the register file, the read address RA can be selected from A0–A3, and the read data RD is connected to either of B0–B3. The least significant 4-bit of WA and RA are input to the register file as the write address and the read address, respectively. A register cell can also be used as delay unit. By calculating 1–16 cycles using the 4-bit counter, this register cell outputs the input data from WD after 1–16 cycles.

As the distinctive feature of register cell, various register access configurations can be defined in order to achieve an efficient cooperation among multiple register cells.

Write enable signal can be defined as always '1,' always '0,' or either of flag inputs f_{i0}–f_{i3}. As another configuration, when the most significant 4-bit of WA is equal to the certain setting value, the write enable signal is asserted so as to provide periodical register accesses. On the other hand, the read enable signal can be defined as always '1' or either of flag inputs f_{i0}–f_{i3}. Same as reading operation, by using the most significant 4-bit of RA periodical register accesses can be facilitated. When the read enable signal is '0,' the read data RD outputs the data of the opposite line, e.g. B0 can output the data from A2.

Flag handling is almost the same as basic cell and multiplication cell, except that each of the most significant 4-bit of RD can be split and output via flag lines in order to implement fine-grained operations such as a control circuit.

Configuration data for register cell is summarized in Table 5.

Table 5 Configuration data for register cell.

#bit	Unit	Configuration
3	Write enable	selection of write enable
3	Read enable	selection of read enable
4	Constants	constants for enable address
3	Flag input	selection of flag input
6	Flag output	selection of flag output (f_{o0}–f_{o3})
6	Word line input	input to port WA, WD, and RA
3	Word line output	selection of output line or Hi-Z

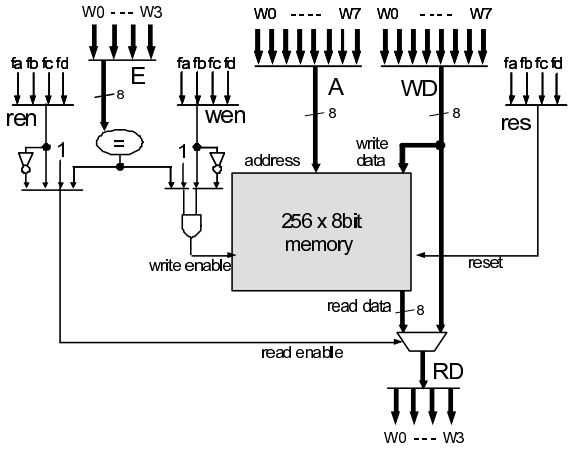


Fig. 8 Organization of function part of memory cell.

Table 6 Configuration data for memory cell.

#bit	Unit	Configuration
4	Write enable	selection of write enable
4	Read enable	selection of read enable
8	Constants	constants for enable address
8	Flag input	selection of flag input (f_a, f_b, f_c, f_d)
8	Flag output	selection of flag output (f_{o0}–f_{o3}) in flag switches a–d (Fig. 4)
8	Word line input	input to port A, WD, and E
3	Word line output	Selection of output line or Hi-Z

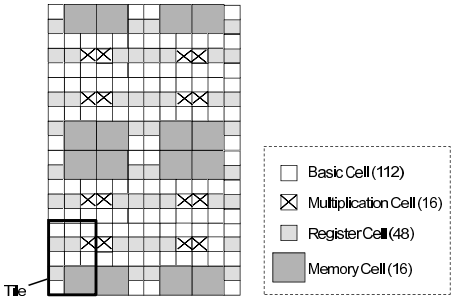


Fig. 9 Heterogeneous structure of 4 types of cells.

3.6 Memory Cell

Figure 8 shows the organization of the function part of memory cell, which has a 256 × 8-bit memory and occupies 4 times as large area as any of other cells.

The input signal and the output signal of memory cell can be selected from W0–W7. The flag input signals f_a–f_d correspond with the flag output from four flag switches, as illustrated in Fig. 4.

For data writing operation to the memory, the write address A and the write data WD can be selected from W0–W7. On the other hand, for data reading from the memory, the read address A and the read data destination RD can be selected from W0–W7 and W0–W3, respectively.

The write enable signal can be selected from the flag input f_a–f_d. When the address extension signal E, which is selected from W0–W3, is equal to the 8-bit certain setting value, the write enable signal is also asserted so as to facilitate access interleaving between multiple memory cells. On the other hand, the read enable signal can be defined as always '1' or either of flag inputs f_a–f_d. When the address extension signal E is equal to the 8-bit certain setting value, the read enable signal can be asserted as well. When the read enable signal is '0,' the read data RD bypasses the write data WD.

Each of four flag switches can independently handle the flag output in the same manner as other types of cells as shown in Fig. 4.

Configuration data for memory cell is summarized in Table 6.

3.7 Heterogeneous Array Structure

As described in Sect. 2.3, the proposed reconfigurable array adopts a heterogeneous structure by combination of four types of cells. First, in order to implement multi-standard video decoder on a reconfigurable array, the ratio of four types of cells is determined based on the required number of cells for the implementation of IDCT of MPEG-2 decoder, since various video decoding algorithms utilize IDCT in almost the same form and its technology mapping requires all types of cells. Then, a heterogeneous array structure of cells is determined to be suitable for mapping of the multi-standard video decoder by considering the following.

- Concatenation use of adjacent cells must be facilitated in order to implement multibyte operations.
- Neighboring memory cells are designed to contain an intermediate decode data of macroblock during run-time reconfiguration.
- Neighboring basic cells and multiplication cells can implement a 16 × 16 multiplier efficiently.
- Register cells, which can implement a delay circuit, must be almost uniformly distributed in the array.
- In order to achieve a scalability of reconfigurable cell array size, the reconfigurable cell array is based on a regular pattern of cell array.

In accordance with the above conditions, the heterogeneous structure of four types of cells is determined as shown in Fig. 9. The heterogeneous structure is based on a basic pattern, called "Tile," and 16 tiles constitute the whole array.

As shown in Fig. 9, tiles are placed with flipped hor-

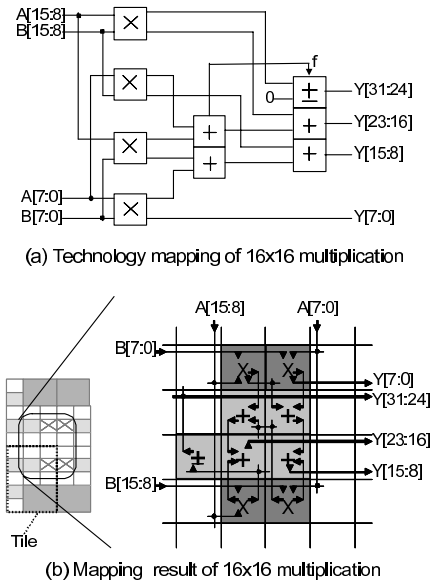


Fig. 10 Mapping of 16 × 16-bit multiplication.

izontally/vertically so that neighboring four memory cells can store the intermediate decode data of one macroblock during reconfiguration. In addition, as shown in Fig. 10, a 16 × 16-bit multiplier can be implemented with five basic cells and four multiplication cells, and mapped on our reconfigurable array with only adjacent cells. Moreover, a 32-bit adder-subtractor can be implemented with adjacent four basic cells. Therefore, 8-point 1D-IDCT operation can be efficiently implemented on the reconfigurable array with four 16-bit multipliers and five 32-bit adder-subtractor.

The proposed heterogeneous array structure, which may be observed with less flexibility as a reconfigurable device, can offer good performance when executing media processing applications with small area occupancy. Moreover, the proposed heterogeneous array structure is microscopically heterogeneous and macroscopically homogeneous, and thus the proposed reconfigurable architecture has scalability in the size of reconfigurable cell array. Naturally in case the cell array size is scaled up, the amount of configuration data increases, for which reconfiguration overhead is discussed in Sect. 3.8.

3.8 Operation of Dynamic Reconfiguration

In order to perform runtime reconfiguration of reconfigurable array, the proposed reconfigurable array executes the transfer of configuration data, the initialization of 8-bit registers of basic cells and multiplication cells, and the initialization of register of register cells, and the initialization of the 256 × 8-bit memory of memory cells is performed only on start-up. For the purpose of reducing reconfiguration overhead, it is very important to reduce the number of cycles needed for the transfer of configuration data and the initialization of registers. Therefore, the proposed reconfigurable array executes the transfer of configuration data, the initialization of 8-bit registers in basic cells and multiplication cells, and the initialization of register file in register cells during runtime reconfiguration, while the initialization of the 256 × 8-bit memory of memory cells is performed only on start-up. As shown in Fig. 1, the configuration data of each cell and the initial value of each register for runtime configuration of reconfigurable array are stored in the configuration memory and the data memory, respectively.

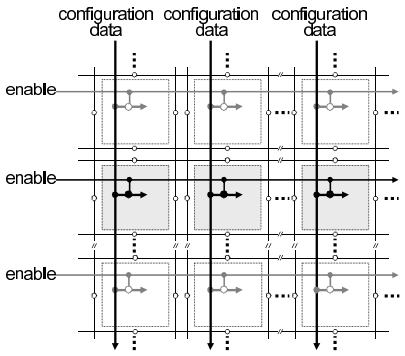


Fig. 11 Delivery of configuration data.

As shown in Fig. 11, the proposed reconfigurable array has dedicated configuration lines for the transfer of configuration data. Specifically, the configuration data can be transferred at the throughput of 492 bits/cycle, which corresponds to the configuration data size for one row of the cell array. Thus the transfer of configuration data to whole reconfigurable array is completed within 20 cycles, which corresponds to the number of cells in one column. If the size of reconfigurable array is scaled up to $m \times n$ tiles, which is equal to the array size of $5m \times 3n$ basic cells, the transfer of configuration data consumes $5m$ cycles at the throughput of $3n \times 41$ bits/cycle.

As shown in Fig. 12(a), the proposed reconfigurable array initializes input/output registers using vertical word lines, which are A1-B3 and B1-A3 in Fig. 3(b). Two 8-bit input registers are initialized with the same value. An initial value is shifted vertically to the next basic cell. Initialization of registers in basic cells and multiplication cells is completed in 24 cycles. In case the size of reconfigurable array is scaled up to $m \times n$ tiles, the initialization of 8-bit registers of basic cells and multiplication cells takes $3m \times 2$ cycles.

As shown in Fig. 12(b), the initialization of 16 × 8-bit register file in register cells is performed using horizontal word lines, which are A2-B0 and B2-A0 in Fig. 3(b). An initial value is fed to the register file in 16 cycles with the write address generated using the 4-bit counter in the register cell. Due to the constraints of the interconnect resources, it is unreasonable to initialize all register cells simultaneously. In our reconfigurable array, as shown in Fig. 12(b), initialization of register cells is performed in a pair of cells. In first 16 cycles, the initial value of the right register cell is stored in the left register cell. In next 16 cycles, the initial value of the right register cell is transferred from the left register cell in parallel with the initialization of the left reg-

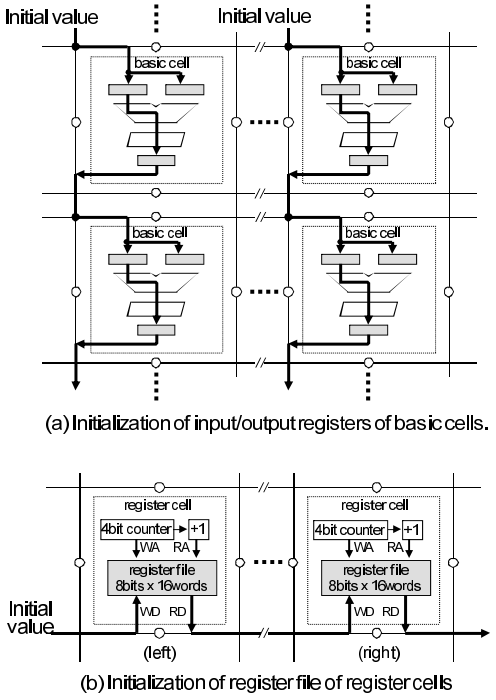


Fig. 12 Initialization of registers of basic cells, multiplication cells, and register cells.

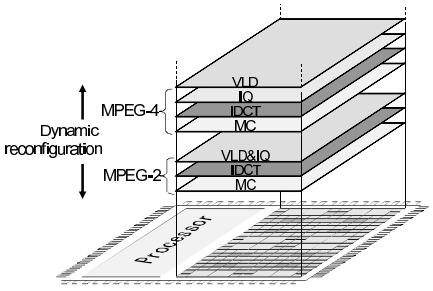


Fig. 13 Dynamic reconfiguration for media processing.

ister cell. In this way, all register cells can be initialized in 32 cycles.

Consequently, in our reconfigurable architecture, the delivery of configuration data and the initialization of registers can be simultaneously performed in 32 cycles.

As shown in Fig. 13, while the configuration is changed in the sequence of IQ, VLD, IDCT, and MC, in decode operation of one macroblock, the reconfigurable array changes its configuration four times per macroblock. Considering the MPEG-2 decoder of 720×480 pixels 30 fps bitstream, the decode operation of one macroblock has to be performed in 24.7 μs, then the decode process of one macroblock has to be performed in 2,470 cycles at the clock frequency of 100 MHz. Our reconfigurable architecture can change its configuration in 32 cycles, and therefore suppresses the reconfiguration overhead to 5.2% of 2,470 cycles. It can be confirmed that our reconfigurable architecture can perform dynamic reconfiguration with low overhead. Due to the lack of space, detailed discussion concerning application mapping will be given elsewhere [10].

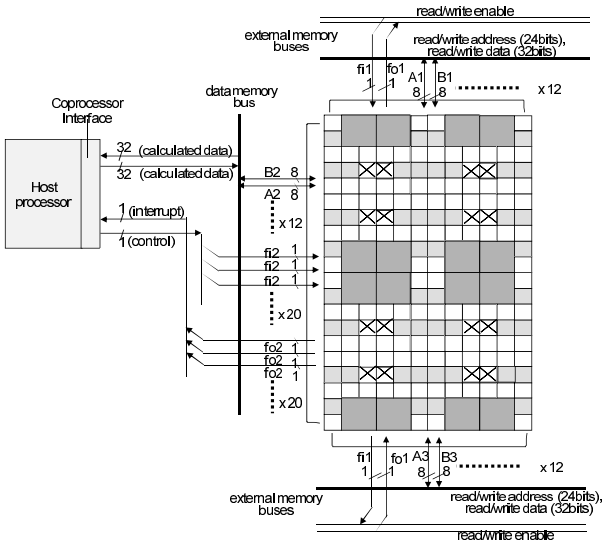


Fig. 14 Interface of reconfigurable device (host processor and external memory).

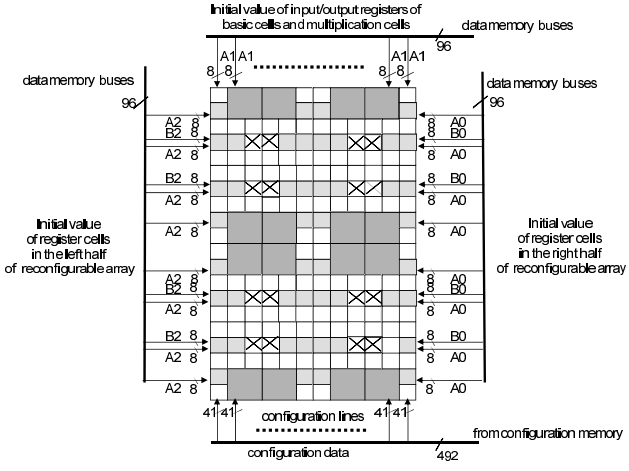


Fig. 15 Interface of reconfigurable device (data memory and configuration memory).

3.9 Interface of Reconfigurable Device

Figure 14 illustrates the interface between reconfigurable array and host processor. A control signal can be connected to a flag input line of a cell on the left side of the reconfigurable array. Similarly, an interrupt signal, generated in a cell on the left side, is reported through a flag out line. On the other hand, data transfer between the host processor and the array is accomplished by means of data memory bus, each 8 bits of which is connected to an A2/B2 word line of any basic cell on the left side of the array. As for the data transfer from/for the external memory, data bus, address bus, and control (enable) lines are connected to any of top and bottom cells. Moreover, memory cells (top or bottom) and register cells (second row) can be used as I/O buffer.

The interface between reconfigurable array and configuration/data memory is shown in Fig. 15. As shown in Fig. 11, the configuration data from configuration memory is transferred through vertical configuration lines. On the other hand, as shown in Fig. 12, initial values of input/output registers of basic cells and multiplication cells are delivered by means of vertical word lines (A1s) on the upper side of the array. Also, initial values of register files in register cells are transferred via horizontal word lines (A0s, B0s, A2s, and B2s) on the right/left side of the array.

4. Implementation Results

4.1 Prototype Chip Implementation

In order to implement the prototype reconfigurable device, at first, an estimation of gate count and delay has been performed by Synopsys Design Compiler in 90 nm CMOS technology. Estimation results of gate count and delay are shown in Table 7.

Then we decided the structure of interconnect switch on the basis of Table 7. In general, an interconnect architecture has large impact to the flexibility and hardware costs of reconfigurable device. As shown in Fig. 3, in the proposed reconfigurable architecture, neighboring cells can be interconnected through a bidirectional switch.

An interconnect switch can be implemented by a pass transistor or tri-state buffer [9]. Pass transistor based interconnect switch is usually smaller and faster than the tri-state buffer based switch. However, in the case of constituting longer path, which has dozens of switches, the tri-state buffer based interconnect is faster than pass transistor based interconnect. Because the area and performance of reconfigurable device depend heavily on the interconnect architecture, it is very important to decide the ratio of the number of pass transistors and tri-state buffers. In other words, the advantages of both switch types can be gained by placing a tri-state buffer after every N pass transistor, where N depends on the interconnect architecture, and must be determined based on the simulation results.

In the array structure of Fig. 9, the longest path between the most distant cells incurs about thirty switches. Fig. 16 shows the waveform of change in electric potential of the wire after thirty interconnect switches. The broken line indicates that the delay of thirty pass transistors is about 7.0 ns. On the other hand, the continuous line indicates that the delay of thirty mixing switches is about 3.0 ns. The mixing switches have a tri-state buffer after every ten pass transistors. The delay of thirty mixing switches is equal to the delay of memory cell. Therefore, as shown in Fig. 17, tri-state buffers are placed in the interconnect, which is across the border of tiles.

The proposed reconfigurable device is designed using 90 nm CMOS technology. After the design of the prototype reconfigurable device, the wire resistance and the wire capacitance are evaluated by Synopsys Star-RCXT. Then an accurate estimation of critical path delay of each cell was performed by Synopsys PathMill, whose simulation result is presented in Table 8.

The longest delay for memory cell is 2.7 ns, which stands for the maximum clock frequency of the proposed reconfigurable device is about 370 MHz. Delay of 32-bit adder, which can be implemented on the neighboring four basic cells, is 2.63 ns, which is almost the same delay as the memory cell. Therefore, in order to prevent the decrease of the maximum clock frequency, the maximum bit width of multibyte operation should be up to four bytes.

Table 9 summarizes the implementation results of the proposed reconfigurable device. The proposed reconfigurable device can be implemented with about 297,000 gates in 90 nm CMOS technology. The size of external memory is 51.6 kbits for configuration memory, and 36.9 kbits for data memory.

A prototype chip of the reconfigurable device has been implemented. Due to the area constraint of the prototype chip, 3×4 tiles and 2×12 basic cells wit single context are implemented, which can execute the realtime processing of MPEG-2 decoder. Figure 18 depicts microphotograph of

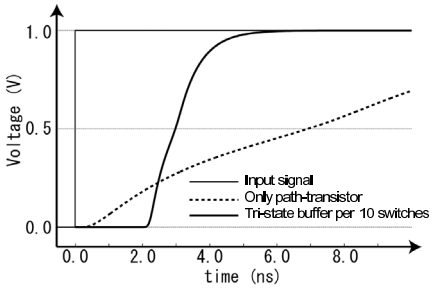


Fig. 16 Change in electric potential after thirty switches.

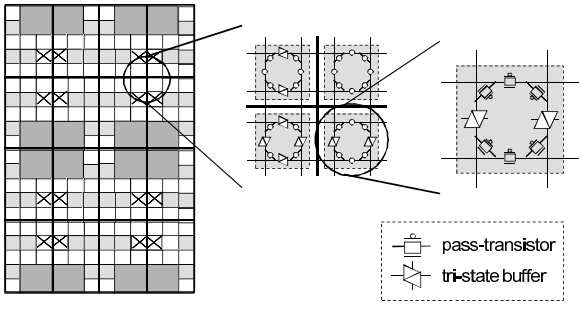


Fig. 17 Mixing structure of interconnect switch.

Table 7 Logic synthesis results of each cells.

	basic cell	mult. cell	register cell	memory cell
# gates	1,015	1,367	2,167	3,568
delay (ns)	2.58	3.50	1.43	3.86

Table 8 Delay of each cells.

	basic cell	mult. cell	register cell	memory cell
delay (ns)	1.47	2.50	1.78	2.70

Table 9 Implementation results.

Technology	90 nm CMOS
Number of gates	296,656
Area of reconfigurable device	1.1 mm × 1.4 mm
Delay on configurable cell	2.70 ns
Max clock frequency	370 MHz
Size of configuration data	8,448 bits/context
Configuration memory (five contexts)	51.6 kbits
Data memory (five contexts)	36.9 kbits

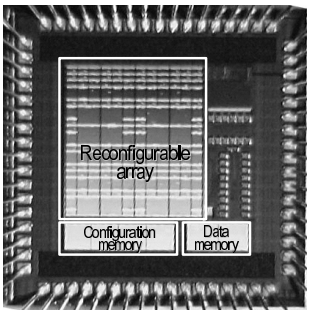


Fig. 18 Chip micrograph of prototype reconfigurable device.

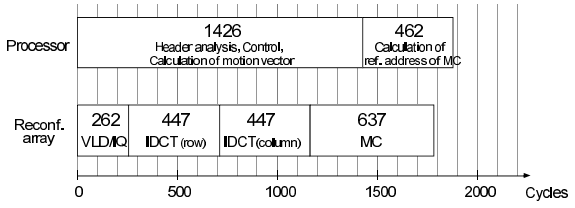


Fig. 19 Number of cycles required for one macroblock processing on MPEG-2 decoder.

the fabricated chip, which is composed mainly of the reconfigurable cell array, the configuration memory, and the data memory.

4.2 Process Scheduling

The number of clock cycles required for one macroblock processing of MPEG-2 decode process is shown in Fig. 19. Here, the overhead of runtime reconfiguration is included, and the targeted host processor is ARM9TDMI [11].

Header analysis on the host processor and VLD on the reconfigurable device needs to access the bitstream, and it is not preferable to be invoked simultaneously. Therefore, as shown in Fig. 20, while the reconfigurable device performs IDCT and MC for a macroblock, the processor performs the header analysis for the next macroblock. On the contrary, while the reconfigurable device performs VLD/IQ, the processor executes the process which does not need the bitstream.

As shown in Fig. 20, the numbers of cycles for MPEG-2 decoding operations of one macroblock are 1,888 and 1,793 on the host processor and the reconfigurable array, respectively. An interrupt from the reconfigurable array to the host processor notifies end-of-block (EOB), and enables the

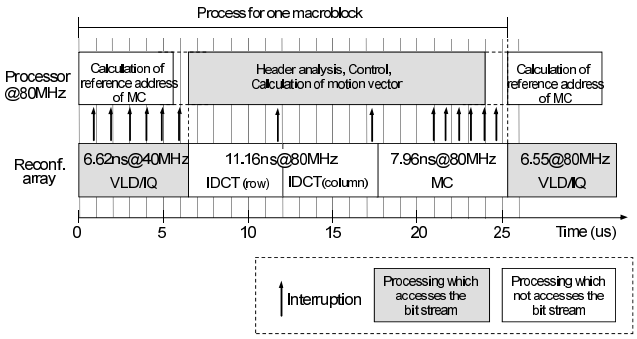


Fig. 20 Process scheduling on MPEG-2 decoder.

host processor to start reconfiguration of the array. In the scheduling of Fig. 20, the overhead of an interruption is assumed to be 15 cycles. Bitstream and reference frame data are directly transferred from the external memory to memory cells or register cells in the array. Decoded picture data is output from register cells in the array to the external memory. Reference addresses of MCs are transferred from the host processor to the array via data memory bus.

In order to perform the realtime processing of 720×480 pixels 30 fps MPEG-2 decoding, a total of 40,500 macroblocks must be processed in each second. Thus the host processor must operate at the clock frequency of 77 MHz or more. On the other hand, implementation results claim that the critical path of VLD/IQ, IDCT, and MC are 22.7 ns, 9.9 ns, and 11.5 ns, respectively. Thus, the maximum clock frequency of VLD/IQ, IDCT, and MC on reconfigurable device are 44.0 MHz, 101 MHz, and 87.0 MHz, respectively.

Considering the above constraints, in order to attain the realtime processing of MPEG-2 decoding, the clock frequencies of 40 MHz for VLD/IQ and 80 MHz for IDCT and MC are required on the reconfigurable device, while the host processor operates at the clock frequency of 80 MHz.

In the same manner as in MPEG-2 decoder, realtime processing of H.263 decoder and MPEG-4 decoder for 352 × 288 pixels 30 fps 384 kbps bitstream can be implemented on the proposed reconfigurable device. Table 10 shows the number of required cells and the number of clock cycles for the context of each algorithms, details of which will be reported in another publication [10]. As shown in Table 10, MPEG-2 decoder, H.263 decoder, and MPEG-4 decoder can be implemented on the proposed reconfigurable device. While MPEG-2 decoder and H.263 decoder are attained by four contexts of configuration data, MPEG-4 decoder takes five contexts.

4.3 Comparison with Conventional Architecture

Henceforth, the performance/area efficiency is evaluated in comparison with other architectures. The area of our reconfigurable device, which includes the configuration memory and the data buffer for four contexts, is 2.1 mm^2.

Table 11 summarizes the comparison results of performance/area efficiency. In case all basic cells and all multi-

Table 10 Cell utilization and number of cycles.

		# cells				# cycles
		basic	mult.	reg.	mem.	
IDCT (row, column)		52	12	9	8	800
MPEG-2	VLD/IQ	87	3	12	10	140
MPEG-2	MC	87	0	13	4	515
H.263	VLD/IQ	63	1	14	8	25
MPEG-4	VLD	83	0	13	9	42
MPEG-4	IQ	36	4	0	9	394
MPEG-4	MC	65	0	9	4	496

Table 11 Comparison of performance/area efficiency.

	area (mm^2)	performance (GOPS)	efficiency ($GOPS/mm^2$)	comment
Proposed	1.6	47.3	29.6	8-bit, 90 nm
		22.2	13.9	16-bit
		10.0	6.3	32-bit
FE-GA [3]	4.5	17.0	3.9	16-bit, 90 nm
D-Fabrix [4]	8.6	14.4	1.8	32-bit, 130 nm

plication cells can operate at the maximum clock frequency of 370 MHz, the performance of the reconfigurable device in the granularity of an 8-bit operations is 47.3 GOPS, namely 22.5 $GOPS/mm^2$. On the other hand, the performances of our reconfigurable device in terms of granularity of 16-bit and 32-bit are 22.2 $GOPS/mm^2$ and 10.0 $GOPS/mm^2$. Due to the difference between our architecture and other architecture in the memory size or the interface of reconfigurable device, it is difficult to make a straightforward comparative evaluation. However, it is apparent that our reconfigurable device has an advantage in the performance/area efficiency when executing media applications.

5. Related Works

One of the advantages of coarse-grained reconfigurable devices is a small area/time overhead for its reconfiguration. In particular, in recent years, ALU-based heterogeneous reconfigurable devices are devised actively.

FE-GA (Flexible Engine/Generic ALU Array) [3] has 16-bit granularity and heterogeneous array structure, which consists of ALU cells, MLT (Multiplier) cells, and LS (Load/Store) cells. The interconnection of cells can connect only four adjacent cells. The main architectural difference from our reconfigurable device is the use of a large crossbar network, which enables flexible access to LS cells and local memory.

On the other hand, MuCCRA (Multi-Core Configurable Reconfigurable Array) [12] is a framework, which generates configurable DRPAs (Dynamically reconfigurable processor arrays) for various target applications. MuCCRA architecture adopts a heterogeneous array structure of PEs (Processing Elements), which consists of an ALU, an SMU (Shift & Mask Unit), and a Register File, Hard Macros, such as multiplier or local memory, and SEs (Switching Elements). MuCCRA architecture equips the equal number of ALUs and Register files, and the limited number of special hard macros at the edge of PE array, which may suffer from low cell utilization rate when executing media processing applications.

An embedded domain-specific reconfigurable architecture is introduced in [13], [14]. However, these reconfigurable architectures offer only limited types of computation, such as ME (Motion Estimation) or DCT (Discrete Cosine Transform), and there still remains considerable disadvantages in terms of area occupation against ASICs.

6. Conclusions

An area-efficient reconfigurable device, which has sufficient performance and flexibility to implement realtime multi-standard decoder, has been proposed. In order to reduce the configuration data and its hardware costs, the application domain of our reconfigurable device is oriented on media processing, and enables the reconfigurable architecture to be suitable to implement media processing applications. Implementation results show that multi-standard decoder can be attained on the proposed reconfigurable device with 1.1 mm × 1.2 mm in 90 nm CMOS technology. The proposed reconfigurable device achieves 3.5 times as high performance as conventional reconfigurable architectures.

Acknowledgements

The authors are grateful to the members of the system architecture group, platform development center, Matsushita Electric Industrial Co., Ltd, for their helpful comments. The 90 nm CMOS process technology in this study has been provided through the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo, with the collaboration of STARC, Fujitsu Limited, Matsushita Electric Industrial Company Limited, NEC Electronics Corporation, Renesas Technology Corporation, and Toshiba Corporation.

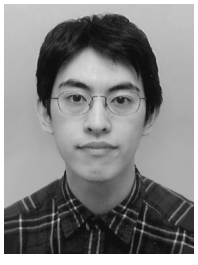
References

[1] R. Hartenstein, "Coarse grain reconfigurable architectures," Proc. Asia and South Pacific Design Automation Conference 2001 (ASP-DAC 2001), pp.564–569, Feb. 2001.
[2] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.26, no.2, pp.203–215, Feb. 2007.
[3] T. Kodama, T. Tsunoda, M. Takada, H. Tanaka, Y. Akita, M. Sato, and M. Ito, "Flexible engine: A dynamic reconfigurable accelerator with high performance and low power consumption," Proc. IEEE Symposium on Low-Power and High Speed Chips (COOL Chips IX), pp.394–408, April 2006.
[4] Elixent Corporation, "http://www.elixent.com/"
[5] N.S. Voros and K. Masselos, System level design of reconfigurable systems-on-chip, Springer, 2005.
[6] M.B. Gokhale and P.S. Graham, Reconfigurable computing, Springer, 2005.
[7] M. Motomura, "A dynamically reconfigurable processor architecture," Proc. Microprocessor Forum, Oct. 2002.
[8] T. Sugawara, K. Ide, and T. Sato, "Dynamically reconfigurable processor implemented with IPFlex's DAPDNA technology," IEICE Trans. Inf. & Syst., vol.E87-D, no.8, pp.1997–2003, Aug. 2004.

[9] G. Lemieux and D. Lewis, Design of Interconnection Networks for Programmable Logic, Kluwer Academic Publishers, 2003.
[10] Y. Mitsuyama, K. Takahashi, R. Imai, M. Hashimoto, T. Onoye, and I. Shirakawa, "Application design flow for media-centric reconfigurable architecture," to be submitted to IEICE Trans. Fundamentals.
[11] ARM Ltd., "ARM9TDMI Technical Reference Manual," ARM DDI 0180A, March 2000.
[12] H. Amano, Y. Hasegawa, S. Tsutsumi, T. Nakamura, T. Nishimura, V. Tanbunheng, A. Parimala, T. Sano, and M. Kato, "MuCCRA chips: Configurable dynamically-reconfigurable processors," Proc. IEEE Asian Solid-State Circuits Conference (A-SSCC2007), pp.384–387, Nov. 2007.
[13] S. Khawam, S. Baloch, A. Pai, I. Ahmed, N. Aydin, T. Arslan, and F. Westall, "Efficient implementations of mobile video computations on domain-specific reconfigurable arrays," Proc. Design Automation and Test in Europe (DATE2004), vol.2, pp.1230–1235, Feb. 2004.
[14] S. Khawam, T. Arslan, and F. Westall, "Synthesizable reconfigurable array targeting distributed arithmetic for system-on-chip applications," Proc. Reconfigurable Architectures Workshop (RAW2004), pp.150–157, April 2004.

Yukio Mitsuyama received B.E. and M.E. degrees in Information Systems Engineering from Osaka University, Japan, in 1998 and 2000, respectively. He is currently an assistant professor in Graduate School of Engineering, Osaka University. His research interests include reconfigurable architecture and its VLSI design. He is a member of IEEE and IPSJ.

Kazuma Takahashi received B.E. and M.E. degrees in Information Systems Engineering from Osaka University, Japan, in 2005 and 2007, respectively. He is currently working at MegaChips Corporation.



Rintaro Imai received B.E. and M.E. degrees in Information Systems Engineering from Osaka University, Japan, in 2004 and 2006, respectively. He is currently working at Renesas Technology Corp.

Masanori Hashimoto received the B.E., M.E. and Ph.D. degrees in Communications and Computer Engineering from Kyoto University, Kyoto, Japan, in 1997, 1999, and 2001, respectively. Since 2001, he was an Instructor in Department of Communications and Computer Engineering, Kyoto University. Since 2004, he has been an Associate Professor in Department of Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University. His research interest includes computer-aided-design for digital integrated circuits, and high-speed circuit design. He is a member of IPSJ.

Takao Onoye received B.E. and M.E. degrees in Electronic Engineering, and Dr.Eng. degree in Information Systems Engineering all from Osaka University, Japan, in 1991, 1993, and 1997, respectively. He is currently a professor in the Department of Information Systems Engineering, Osaka University. His research interests include media-centric low-power architecture and its SoC implementation. He is a member of IEEE, IPSJ, and ITE-J.

Isao Shirakawa received the B.E., M.E., and Ph.D. degrees in Electronic Engineering from Osaka University, Japan, in 1963, 1965, and 1968, respectively. He joined Faculty of Engineering, Osaka University, as a Research Assistant in 1968, and was promoted to an Associate Professor in 1973 and to a Professor in 1987, where he worked as the Dean in 2001–2002. He retired from Osaka University in 2003, and is now a Professor Emeritus of the same university. Since the retirement he has been a Director of Synthesis Corporation (an academia-industry collaboration based start-up). Since the academic year of 2004 he has been the Dean of Graduate School of Applied Informatics, University of Hyogo, Kobe, Japan. Meanwhile, in the academic year of 1974–1975, he was a Visiting Scholar at the Electronic Research Laboratory, University of California, Berkeley. Prof. Shirakawa was a Vice President of IEEE CAS Society in 1995 and 1996, and the Chair of IEEE Kansai Section in 2002 and 2003. He was a Director of Editorial Board of IEICE of Japan in 1996 and 1997, and was a Vice President of IEICE in 2004–2005. He is now the Chair of Industry Liaison Committee of IEEE Region 10. He has been engaged in the education as well as in the research mainly on circuit theory, graph theory, VLSI CAD, VLSI architecture, and RFID system. He is now a Life Fellow of IEEE.