

EMBEDDED RECONFIGURABLE ARRAY TARGETING MOTION ESTIMATION APPLICATIONS

Sami Khawam¹, Tughrul Arslan^{1,2}, Fred Westall³

¹ Department of Electronics and Electrical Eng.
University of Edinburgh, King's Buildings,
Mayfield Road, Edinburgh, EH9 3JL, UK

² Institute for System Level Integration
The Alba Centre, Alba Campus
Livingston, EH54 7EG, UK

³ EPSON Scotland Design Centre
Integration House, Alba Campus
Livingston, EH54 7EG, UK

ABSTRACT

Motion estimation is a complex computation found in video compression algorithms, such as standards like MPEG-4 and H.263. This paper proposes an embedded reconfigurable array for low-power and high-throughput implementations of motion estimators. This array is part of a heterogeneous SoC platform with multiple processors and computational elements. The embedded reconfigurable array targets multimedia and low-power system-on-chip applications, such as future mobile devices. We provide results which demonstrate a 75% reduction in power consumption in addition to improvements in timing and area over standard FPGA architectures.

Keywords: reconfigurable, programmable, array, motion estimation, FPGA, domain specific, embedded

1. INTRODUCTION

Future mobile devices will process both video and speech data. This pushes the needs for high-speed low-power architectures for implementing complex compression algorithms like MPEG-4 [1] and H.263 [2], as the current implementation of these algorithms on DSP processors do not provide enough power reduction and require a high operating frequency.

Interest in reconfigurable hardware has increased in the past years as these provide low-cost and flexible architectures for custom hardware, while offering a high throughput using parallel computations. The provided low-level fine-grained flexibility, however, increases both consumed power and area of the chip.

In this paper a novel reconfigurable array targeting complex MPEG-4 operations is presented. Using a domain-specific programmable array for motion-estimation would provide a compromise between performance, speed, power-consumption and flexibility. Such domain-specific programmable logic as in [3] can provide a very high-performance and flexible implementations. For motion estimation, such architectures could be flexible enough to permit changes in algorithms and at the same time provide an efficient and low-power implementation. The programmable array in this paper is embedded in a system-on-chip device and works along digital signal processors (DSPs) and controllers.

This paper is organized as follows: In section 2, the target algorithm and its previous implementations are overviewed.

Section 3 describes the system-on-chip architecture. The array architecture is outlined in section 4 and its performance assessed in 5.

2. MOTION ESTIMATION

The array designed targets the computation of Motion Estimation, which is performed by comparing a block of pixels in the current frame to blocks in the previous frame. This allows selecting a block from the previous frame that resembles the most to the block in the current frame, and hence compressing the video data by using the motion information between the two blocks. The comparison of two blocks is brought about by calculating the sum-of-absolute-differences (SAD) between corresponding pixels in the blocks. This calculation requires subtractions, absolute difference calculations and accumulations.

A number of motion estimation algorithms exists based on the SAD calculation [4]. These algorithms differ by the order, number and size of blocks compared as well as by the bit-width of the pixels.

Some of the existing architectures for motion estimation provide flexibility in the supported algorithms. The hardwired elements proposed in [5] can be configured at run-time to support 3 different bit-widths to save power; however, only one basic algorithm is supported. Similarly, [6],[7] present architectures supporting only one algorithm but having flexibility in the size of blocks and search area. The hardware in [8],[9] offer reconfigurable elements that can switch between two algorithms differing by the number and the order of blocks searched.

The flexible reconfigurable motion estimation array proposed here provides an architecture that supports a larger number of different SAD-based motion estimation algorithms. This flexibility can be used at design-time as well as run-time to adapt the system to real time constraints.

3. SYSTEM OVERVIEW

The target multimedia communication system for the reconfigurable array described in this paper consists of a Digital Signal Processor (DSP) and a number of reconfigurable arrays, each targeting a specific computation. The arrays and the processors communicate through a system-on-chip bus such as the AMBA bus [10]. A controller is used to integrate all the elements together.

To perform the motion estimation computation, the array considered in this paper receives the current-block and the search-area data from the DSP. The array processes the data and the best motion-vector is found and reported back to the processor. The array contains internal memory for storing data coming from the DSP and intermediate data found during the calculation.

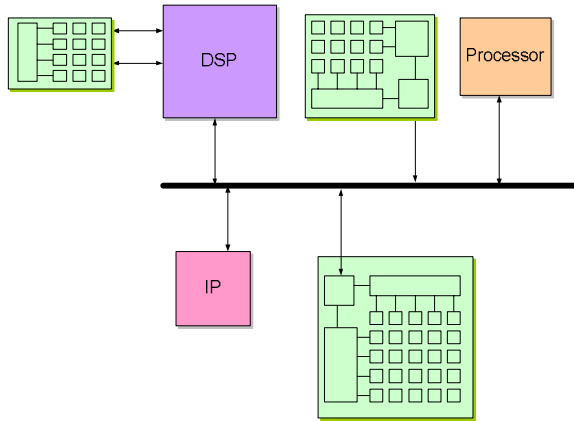


Figure 1: Embedded reconfigurable arrays part of a System-on-chip

The reconfiguration of the array is performed dynamically at run-time by the controller or by the DSP, which controls the algorithm programmed into the array. Different motion-estimation algorithms have different speed and power consumption performance depending on the number of pixels compared and numbers of bits compared per pixel [9]. Thus, the motion-estimation algorithm configured in the array can be adapted during run-time according to the constraints on the system, such as the required power consumption and picture quality.

In the future, a library of reconfigurable elements could be created where each element supports a specific operation. Creating a domain-specific reconfigurable array would be performed by arranging the required elements in an array and using reconfigurable interconnects. A number of domain-specific arrays can be used alongside a DSP processor with each array specific to one MPEG-4 calculation. Alternatively a single large array with all the required elements could be used instead of many arrays.

4. RECONFIGURABLE ARRAY

4.1 Computational elements

In order to enable the array to support different motion estimation architectures, elements with the following functions are provided:

- Multiplexers: 2-to-1 multiplexers with optional register at the output. Using interconnects the multiplexers can be cascaded to create larger input sizes.
- Adders: Modules supporting combinatorial 2-input additions and subtractions. An optional combinatorial absolute-

difference calculators, useful for SAD based motion estimation, is also found at the output of the module.

- Accumulators: Sequential accumulators which can also be configured as simple combinatorial adder/subtractors. The accumulator contains an internal register.
- Comparators: Modules enabling the comparison of two numbers producing greater-than and equal signal. Registers and logic are also available for finding and storing the minimum/maximum value useful for the minimum SAD selection.

The adders and multiplexers are 8-bits wide and can be cascaded to produce higher bit count, in case the pixels bit-width changes. The accumulators and comparators are 16-bit wide and can also be cascaded. The use of 8-bit elements reduces the number of configurable interconnects used when compared to 1-bit elements, thus, improving power consumption and area. In usual image data 8-bit values are used for representing one color of a pixel.

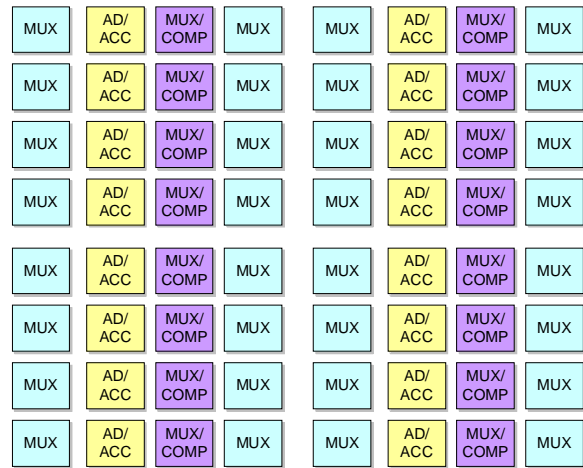


Figure 2: Possible array arrangement of clusters

The array elements provided enable to compute the SAD using the absolute-difference calculators, accumulators and comparators to select the minimum SAD. The array could also be configured to enable other computations using the same modules. The usage of an array of elements permits having parallel computations and implementations of tree based-structures such as adders/subtractors.

Data outputs from elements contain optional flip-flops that make it possible to register the output of the modules. This is useful for situations where a pipelined arrangement is required or for implementing systolic motion-estimation arrays [11],[12],[13]. Furthermore, unconfigured elements that are unused are disabled in order to reduce power consumption.

The elements are arranged in clusters of four, with each of the four elements having its pins on one of the four sides of a cluster. The elements of the cluster are not necessarily all the same. The clusters are arranged in an array as shown in Figure 2. Other array arrangements providing speed and area improvements are possible, however for the purpose of manually

generated array configuration a uniform cluster arrangement was chosen.

4.2 Control elements

The elements described above provide the computations for motion estimation. Furthermore, generic reconfigurable elements are required to provide control logic as well as general logic such as the selection of the minimum SAD value. These elements are not described here nor included in the performance evaluation, as this paper focuses on performance critical sections only.

4.3 Interconnects

Two levels of interconnects are provided in the array: Interconnects inside the clusters and interconnects between the different clusters. This provides a compromise between speed and area: Inside the cluster, short reconfigurable connection lines are provided to make it easier to implement high-speed common connections such as the connection between the output of the absolute-difference calculator and the input of the accumulator. Other supported connections are those needed to concatenate 8-bit modules into 16-bit modules. Such lines are provided inside every cluster.

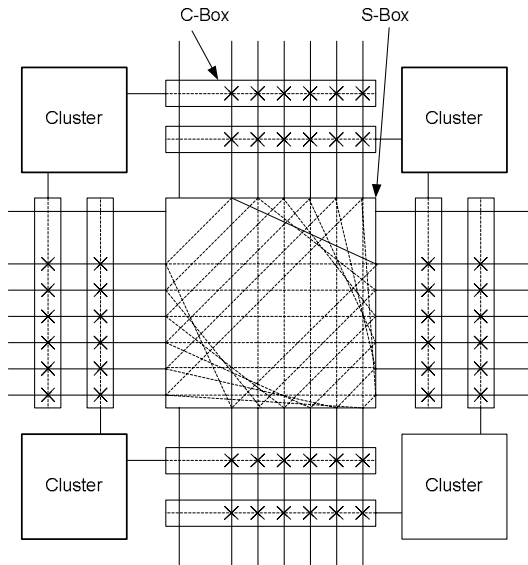


Figure 3: Clusters, connection and switch boxes

On the array level, FPGA-like symmetrical mesh array interconnects are used to provide the connections between the clusters (see Figure 3). Two types of tracks are provided: Six 8-bit wide tracks for data and six 1-bit tracks for control lines. These interconnects are provided by connection-boxes (C-Boxes) that connect the pins of a cluster to the tracks and switch-boxes (S-Boxes) that connect together the intersections of tracks.

In the initial design, the C-Boxes connected to a pin enable to route the signal on the pin to any of the 6 tracks, meaning a flexibility of $F_c=6$ according to the definition in [14]. This is the

maximum F_c value possible, and was chosen for its simplicity. The S-Boxes used have the standard flexibility of $F_s=3$, with F_s representing the number of choices offered to each wire entering the box. Switch boxes with higher flexibility could provide small routing improvements [14].

For reasons of synthesizability, the configurable switches were implemented using tri-state buffers as shown in Figure 4. Unidirectional switches are also available for unidirectional pins of the clusters. The use of tri-state buffers affects the area, timing and power consumption of the array [15], when compared to using pass-transistors. Hence, the performance measured in section 5 can be greatly improved.

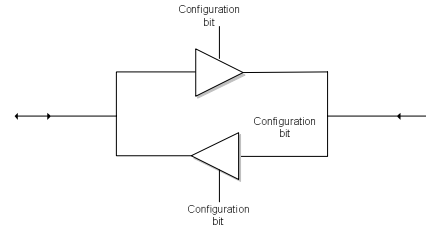


Figure 4: Bi-directional configurable switch using tri-state buffers

5. PERFORMANCE COMPARAISON

The benchmark circuit used for determining the performance of the system is the motion-estimation array described in [13]. In this implementation, 16 processing-elements (PEs) are used simultaneously to find the SAD values of 16 candidate motion-blocks. The block size is 16x16 and the search area is 32x32 pixels wide. The current motion-block data is propagated through the PEs, while two pixels from the search-area are broadcasted to the PEs.

Each PE is composed of a multiplexer, register for propagation, absolute-difference calculator, accumulator and a comparator for selecting the minimum SAD found on that PE. In our reconfigurable array, a PE has been initially mapped to 3 clusters. The simple mapping was done manually and the following clusters were used:

- A cluster of four multiplexers and registers for implementing one multiplexer and one register.
- A cluster of two absolute-difference calculators and two accumulators for implementing one of each.
- A cluster of two comparators and registers and two multiplexers to implement one minimum-value finder.

Clearly, the mapping of elements is not the most efficient in terms of area usage since it was performed manually. An intelligent automatic mapping process, similar to the ones found in current FPGA implementation software would have produced better results in terms of area and timing.

For comparative results, the [13] architecture has been implemented in standard ASIC, on Xilinx Virtex-E [17] FPGA and on an ARM7 [16] low-power DSP. All of these systems use .18 μ m CMOS technology and run at 1.8V and 30MHz. In the case of the ARM processor, it is estimated that the processor

computes simultaneously the calculations made by two PEs, and that the power consumption is 0.39 mW/MHz [16]. The area estimation of the Xilinx FPGA is based on the estimation of $710\mu\text{m}^2$ per slice and its surrounding routings. Table 1 shows the performance results.

	.18 μm ASIC	Our array	Xilinx's Virtex-E	ARM7
Power cons. (mW)	0.68	1.08	4.37	5.85
Area (μm^2)	8594	21189	38340	N/A
Max Freq. (MHz)	440	111	90	N/A

Table 1: Performance of the implementations of one processing-element from [Yang89]

It can be seen that a considerable power reduction, of 75%, is achieved in comparison to commercial generic reconfigurable logic. It should be noted that these figures do not include the power consumed by configuration memories. In the case of our array, the figures shown represent the power consumption in four clusters and their associated row of C- and S-boxes. In comparison to hardwired ASIC, our array consumes 59% more power. The reduced power compared to the processor implementation is also significant. Moving the motion estimation from software to hardware should also reduce the timing constraints on the processor.

The proposed array with non-optimum mapping is 240% larger in area than the ASIC implementation; nevertheless this is 45% smaller than the FPGA implementation. From the timing point of view, both the proposed array and FPGA implementations have a similar maximum frequency which is about 4 times lower than the maximum frequency in hardwired systems.

6. SUMMARY

A domain-specific embedded reconfigurable array targeting the motion-estimation computation has been presented. The array provides elements such as registers, multiplexers, adders, absolute-difference calculators, accumulators and comparators which permit to map different motion-estimation algorithms to the array. The elements are grouped in clusters which form the array. Different levels of reconfigurable interconnects are provided to improve timing, area and power consumption.

Initial performance figures show that the proposed array provides a compromise between hardwired ASICs, generic FPGAs and DSP processors. When compared to generic FPGA, the power consumption is reduced considerably by around 75%, whereas area is reduced by 45% and timing improved by 23%. With respect to hardwired ASIC, the area of the proposed array is 240% larger and the maximum frequency is reduced by 75%

7. REFERENCES

[1] ISO/IEC 14496-2, Information technology — Coding of audio-visual objects — Part 2: Visual, AMD1, Geneva, 2000

[2] ITU, ITU Recommendation H.263, DRAFT H.263, Video Coding for low bit rate communication, Jan. 1998

[3] Hounsell, B.I.; Arslan, T., *An embedded programmable core for the implementation of high performance digital filters*, ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International, 2001, Page(s): 169 -174

[4] Namuduri K.R., Aiyuan Ji., *Computation and performance trade-offs in motion estimation algorithms*, Information Technology: Coding and Computing, 2001. Proceedings. International Conference on, 2001, Page(s): 263 - 267

[5] Takagi, A.; Muramatsu, S.; Kiya, H., *Motion estimation with power scalability and its VHDL model*, Image Processing, 2000. Proceedings. 2000 International Conference on, Vol.3, 2000, Pages: 118- 121 vol.3

[6] Fanucci, L.; Saletti, R.; Bertini, L.; Moio, P.; Saponara, S., *High-Throughput, Low Complexity, Parametrizable VLSI Architecture for Full Search Block Matching Alg*, Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on, Vol.3, 1999, Pages: 1479- 1482 vol.3

[7] Yuan-Hau Yeh; Chen-Yi Lee, *Scalable VLSI Architectures For Full-Search Block Matching Algorithms*, Image Processing, 1996. Proceedings., International Conference on, Vol.1, 1996, Pages: 1035- 1038 vol.2

[8] Xiao-Dong Zhang; Chi-Ying Tsui, *An Efficient And Reconfigurable Vlsi Architecture For Different Block Matching Motion Estimation Alg*, Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on, Vol.1, 1997, Pages: 603- 606 vol.1

[9] Burleson, W.; Jain, P.; Venkatraman, S., *Dynamically parameterized architectures for power-aware video coding: motion estimation and DCT*, Digital and Computational Video, 2001. Proceedings. Second International Workshop on, Vol., 2001, Pages: 4- 12

[10] ARM, *AMBA Specification*, (Rev. 2.0), 1999

[11] Komarek, T., Pirsch, P, *Array architectures for block matching algorithms*, Circuits and Systems, IEEE Transactions on, Volume: 36 Issue: 10, Oct. 1989, Page(s): 1301 -1308

[12] De Vos, L.; Stegherr, M., *Parameterizable VLSI architectures for the full-search block-matching algorithm*, IEEE Transactions on Circuits and Systems, Vol.36 Issue: 10, Oct. 1989

[13] Yang, K.-M.; Sun, M.-T.; Wu, L., *A family of VLSI designs for the motion compensation block-matching algorithm*, IEEE Transactions on Circuits and Systems, Vol. 36 Issue: 10, Oct. 1989

[14] Rose J., Brown S., *Flexibility of interconnection structures for field-programmable gate arrays*, Solid-State Circuits, IEEE Journal of, Vol.26, Iss.3, 1990, Pages: 277- 282

[15] V. George, H. Zhang J. Rabaye. *The design of low energy FPGA*, Proceedings. 1999 International Symposium on Low Power Electronics and Design, pp. 188-193. 1999.

[16] ARM, ARM7 Datasheet (Rev. 20C), 1994

[17] Xilinx, *The Programmable Logic Data Book*, Xilinx Inc., 2001