

Synthesizable Reconfigurable Array Targeting Distributed Arithmetic for System-on-Chip Applications

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

¹ *School of Engineering and Electronic, The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh EH9 3JL, UK. S.Khawam@ee.ed.ac.uk*

² *Institute for System Level Integration, Livingston, EH54 7EG, UK*

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

Abstract

Domain-specific reconfigurable arrays are embedded arrays optimized for one domain of applications providing performance improvements over generic embedded Field Programmable Gate Arrays (FPGAs). In this paper, an embedded reconfigurable array that targets Distributed Arithmetic (DA) implementations is presented. DA includes calculations that are commonly found in multimedia applications, such as filtering and Discrete Cosine Transform (DCT). Two benchmark DCT circuits are implemented on the array, on conventional FPGAs and on hardwired cores. The performance measured shows considerable improvements in area, power consumption and timing when comparing the presented array with FPGAs. Experimental results are provided which demonstrate the suitability of our architecture in low-power System-on-Chip platforms targeting portable mobile devices.

1. Introduction

Current reconfigurable logic and Field Programmable Gate Array (FPGAs) technologies are providing powerful, cost-effective and flexible solutions for a broad range of applications. The emerging trend in configurable logic are embedded Reconfigurable Arrays (RA) for use in System-on-Chips (SoCs) to run complex computations that could benefit from parallel execution. Embedded versions of commercial FPGAs are provided by several companies [5] [19]. Embedding such an array in a SoC adds extra flexibility to the ASIC to allow post-fabrications modifications. This can be used to reduce the development time, fix errors and add new functionality to the design avoiding a respin, and hence reduce costs. Such flexibility is very useful for implementing complex computations part of changing standards such as MPEG-4.

However, the flexibility added by the embedded FPGA introduces an overhead in power and area consumption that makes it unsuitable for critical applications such as

battery powered devices. Another problem presented by embedded FPGAs is the difficult integration of the array with the rest of the SoC architecture and design-flow.

The addition of a RA to a SoC opens the opportunity of optimizing the array to the application instead of just dropping a generic array. This is underlined by the observation that in most applications the RA is used for one specific calculation that is known to the designer prior to the development of the SoC. Example applications include portable-devices where the RA can be optimized for the domain of applications in such a way to reduce power and area consumption. This concept of domain-specific arrays has been demonstrated by the authors in [1] by presenting a RA targeting the Motion Estimation algorithms found in mobile video devices. The RA showed considerable improvements in performance, area and power consumption when compared to commercial FPGAs, which is achieved by using coarse-grain elements and lower flexibility than commercial FPGAs.

This paper presents a domain-specific RA targeting algorithms that can be implemented using Distributed Arithmetic (DA). DA is a technique to efficiently implement dot-vector multiplications containing constant coefficients. Such algorithms include computations like Discrete Cosine Transform (DCT) and filtering, which are frequently used in audio and video applications. Current implementations of these algorithms in mobile devices suffer from the low throughput of Digital Signal Processors (DSPs) or from the inflexibility of hardwired cores or from the high power consumption of FPGAs.

The presented array permits a high throughput and low-power solution for complex calculations. At the same time it provides a margin of flexibility to allow changes to be done after fabrication, hence, it represents a compromise between FPGAs and hardwired implementations in terms of flexibility, power consumption and area. The RA is provided as synthesizable soft-core to allow the customization of all the aspects of the array at design-time. This also permits an easy integration of the arrays into the SoC architecture

and software flow, as the functionality of the whole system can be easily verified at every stage of the design-flow as if the array was a standard core. This solves the design and integration problems associated with generic embedded FPGA cores.

The rest of the paper is organized as follows. Section 2 describes previous related work. Section 3 overviews the reconfigurable SoC and software flow and in section 4 the DA and DCT algorithms are presented. The reconfigurable array for DA is proposed in section 5, and in section 6 the performance of the array is assessed in terms of area, power consumption and timing.

2. Related work

The array presented in this paper is novel in combining different types of coarse grain elements and an interconnect-mesh. This section reviews previous work on related reconfigurable architectures.

The configurable architecture presented in [4] is based on small programmable processors that can be interconnected together for the execution of complex algorithms. The interconnects used are reconfigurable to provide extra flexibility, while the choice of the processors makes the architecture domain specific. When compared to this architecture, our architecture provides a higher throughput and lower power consumption due to the added parallel processing.

Reconfigurable arrays are usually based on programmable Look-Up-Tables (LUTs) that can be configured to implement any desired logic function. This provides a high level of flexibility in implementing random combinatorial and synchronous circuits. Another option is to use coarse-grain elements such as programmable ALUs to allow the implementations of calculations and control processes.

The low-power FPGA presented in [13] is fine-grain and LUT-based and can be integrated with programmable processors. Alternatively, the CHES [18] architecture is 4-bit ALU-based and targets multimedia applications. However, the array is homogenous and all the ALU provide the same functionality. The array presented in [17] is 4-bit and LUT-based; it is made DSP specific by reducing the flexibility of the LUTs in implementing random logic, which makes this homogenous array efficient in mapping datapath logic. In [16] the concept of synthesizable RAs is presented, however it is only used to develop cores for random combinatorial logic and no synchronous computations.

In contrast to other reconfigurable arrays and reconfigurable processor, the array presented in this paper is heterogeneous and uses a number of different coarse-grain cells each optimized to one computation. The cells are up to 16-bit wide and support computations at a

higher-level than other coarse-grain architectures. The array supports both combinatorial and pipelined implementations.

Programmable architecture designed only for DCT such as the one presented in [6] offer flexibility in choosing some of the DCT parameters; however, such flexibility is limited when compared to the wide range of possible DCT implementations. By using FPGA-style interconnects and elements we aim to achieve more flexibility at a lower-level.

3. Reconfigurable system-on-chip

As introduced in [1], our system contains processors and Digital Signal Processors (DSPs) alongside a number of embedded RAs. Each RA can be specific to a computation, such as Motion Estimation or DCT. The system also allows having a combined array that targets multiple computations. The RA is easily incorporated in the SoC design-flow as a synthesizable soft-core.

As will be described later in section 5, a RA consists of programmable clusters interconnected using configurable switches. The clusters define the functionality of the array. The RA can be heterogeneous and could contain different types of interconnectable clusters, with each cluster specific for an operation.

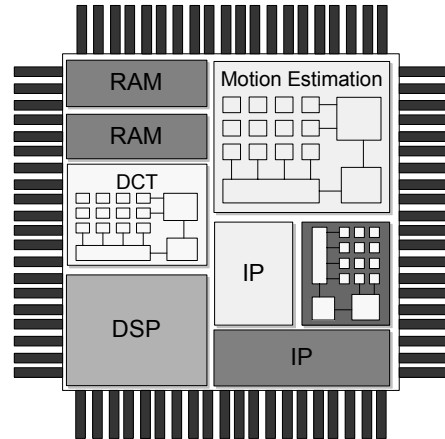


Figure 1: Basic scheme of a SoC with reconfigurable arrays, each targeting a specific domain of computations. The parameters of the reconfigurable arrays can all be set at design time.

The clusters forming the array can be chosen at design-time according to the requirements and constraints. The aim of the project is to develop a library of clusters that can be incorporated in RAs. The domain and the flexibility of the RA can be set according to the choice of clusters and type of interconnects used. The RA is provided as a soft-core which can be simulated, synthesized and routed as a normal core. This allows

incorporating the RA into the design flow of the full SoC, making the design and verification of the system easier.

The array is configured by the processor or the DSPs in the system. This can be done dynamically to allow adjusting the operation of the RA at run-time according to the needs. The data to and from the array are fed and read back by the processor through the on-chip bus.

4. Distributed arithmetic

4.1. Distributed arithmetic implementations

Distributed Arithmetic (DA) implementations [3] target sum-of-product calculations with multiplications by fixed-coefficients. If x_k are the input samples, A_k the constant coefficients, and K is the number of elements, then the computation:

$$y = \sum_{k=1}^K A_k \cdot x_k \quad (1)$$

can be re-written using the bit-notation of x_k as in equation (2), where x_{nm} is bit m of input x_n , and B is the number of bits. The complexity of the circuit is reduced by replacing multiplications by look-up-tables (LUTs) followed by shift-accumulations. The LUTs contain the precomputed values of the AND-operations and additions shown in equation (2). The address to the LUT is formed by combining bits $x_{1i}, x_{2i}, \dots, x_{Ki}$ from the input variables.

$$y = \begin{aligned} & - [x_{10} \cdot A_1 + x_{20} \cdot A_2 + \dots + x_{K0} \cdot A_K] \cdot 2^0 \\ & + [x_{11} \cdot A_1 + x_{21} \cdot A_2 + \dots + x_{K1} \cdot A_K] \cdot 2^{-1} \\ & \quad \vdots \\ & + [x_{1B} \cdot A_1 + x_{2B} \cdot A_2 + \dots + x_{KB} \cdot A_K] \cdot 2^{-B} \end{aligned} \quad (2)$$

Bit-serial computations can be used to reduce the area of DA implementations: The computation of (2) can be implemented with K shift-registers that convert the K inputs from parallel to serial. The K output bits ($x_{1n}, x_{2n}, \dots, x_{Kn}$) are combined to form the address to one LUT. The output of the LUT is fed to a shift-accumulator, where the result y is found after B cycles. Digit-serial implementations can increase the throughput at the cost of increased area, as more bits are processed at a time. Bit-serial and digit-serial implementations are supported by the array presented below. Furthermore, several techniques can be used to reduce the complexity of the circuit and the size of the memory, as described in [7]. Such methods are also implementable using the array.

4.2. Discrete cosine transform

DA provides an efficient implementation for multiple sum-of-product calculations having the same input. This is the case for the Discrete Cosine Transform (DCT) [8]: A 1-D N -point DCT is computed using:

$$X_u = c(u) \cdot \sum_{i=0}^{N-1} x(i) \cdot \cos\left(\frac{(2i+1) \cdot u\pi}{2N}\right) \quad (3)$$

which can be considered as N parallel Finite Impulse Response (FIR) filters. A bit-serial DA implementation would consist of N shift-registers for parallel-to-serial conversion, N LUT memories and N shift-accumulators. All the N memories receive the same address. The 8-point 1-D DCT is shown in figure 2.

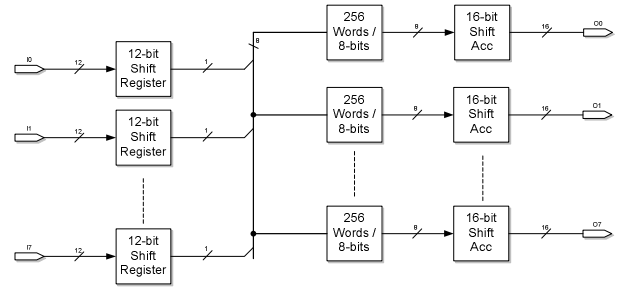


Figure 2: Simple DCT implementation using distributed arithmetic without memory reduction.

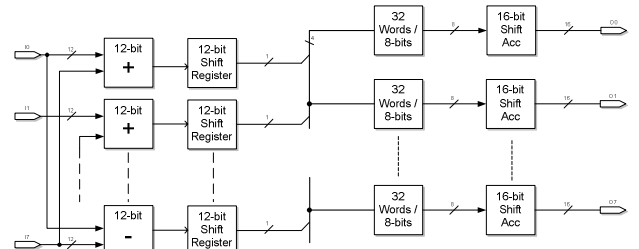


Figure 3: Implementation of DCT using odd-even decomposition for memory reduction.

The array proposed in this paper is flexible enough to support a number of possible DCT implementations using DA, such as the one presented in [9] where COordinate Rotation DIGital Computer (CORDIC) computations are used to reduce the memory size, and in [10] where 3-bit digit serial are used to improve the throughput of the array. The odd-even decomposition technique also described in [10] and shown in figure 3 can be used to reduce the memory size by using adders and subtractors at the input.

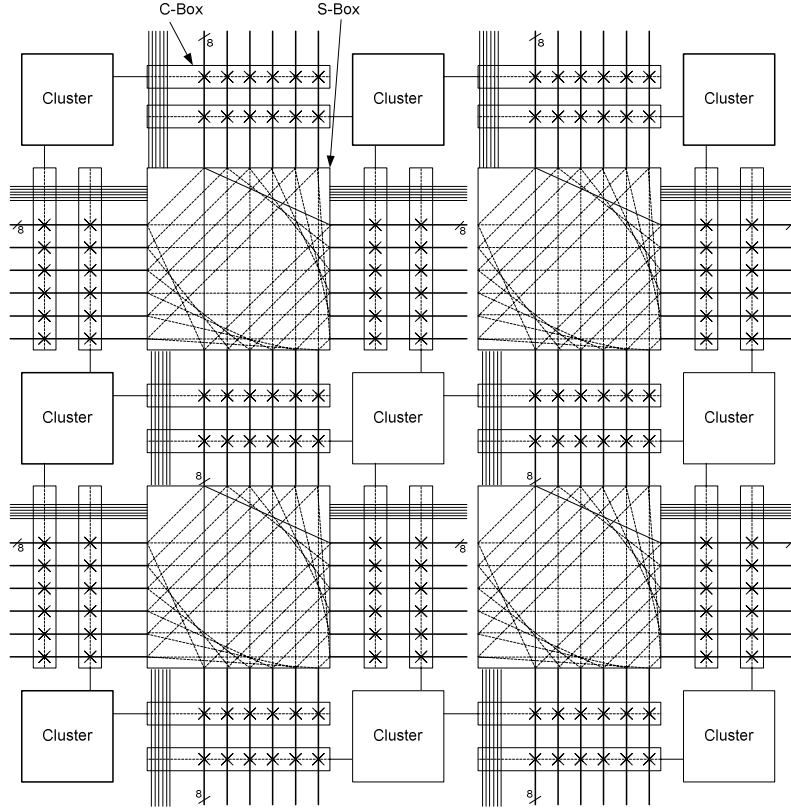


Figure 4: Basic interconnects mesh scheme used in the synthesizable array to connect clusters together. In the initial array six 1-bit tracks and six 8-bit tracks are used. The Connection-Boxes have maximum flexibility, and the Switch-Boxes have an $F_s=3$.

5. Reconfigurable distributed arithmetic array

5.1. Elements for distributed arithmetic

As described in the previous section, a simple DA implementation requires shift-registers, memory elements and shift-accumulators. Additionally, to accommodate for a wider range of algorithms such as odd-even DCT or reduced-memory DA, adders and subtractors are needed. Hence, two types of clusters have been identified and used in the proposed RA: Memory clusters for LUTs and add-shift clusters for making add/sub/shift and accumulation. These are described below.

5.1.1. Memory cluster

A dual-port 512-bit RAM, organized as 64 words 8-bits per word, is used as the basic memory element. Four such memory elements are grouped together to form a 2K-bit memory cluster. The grouping is done using logic to enable configuring the cluster as a memory with all the possible geometries listed in table 1. The logic used is similar to the one presented in [11]. It should be noted that each memory element can be turned off and on separately; hence, allowing the lower sized memories of table 1. This also reduces power consumption in unused memory.

Table 1: Possible geometries of the memory cluster.

Word Size	Bits per word			
	8-bits	16-bits	24-bits	32-bits
64	x	x	x	x
128	x	x		
192	x			
256	x			

Having elements with these memory sizes enables the realization of basic DA implementations, as well as those with compressed memory described in section 2. Clusters of memory can be further combined together using interconnects between the memories to make wider memories. Dual-port memories were chosen for easier configuration: Data is written during configuration on one port and read during operation on the other port.

5.1.2. Add and shift cluster

The add-shift modules provided can be configured as:

- Parallel, digit-serial or bit-serial adders/subtractors.
- Shift registers that can be used for parallel-to-serial conversion. Right and left shifts are supported.
- Accumulators with optional shift-accumulation.

Each module is 4 bits wide; four modules are grouped into a cluster and configurable switches are provided between them to support cascading to get wider bit ranges

(up to 16-bits). Wider operations are possible by cascading multiple clusters using the interconnects mesh described below.

5.2. Elements and interconnects mesh

The columns are arranged uniformly as in figure 5 to make it simpler to make manual routing and placement when configuring the array. Other arrangements are also possible, but this would require automatic routing software.

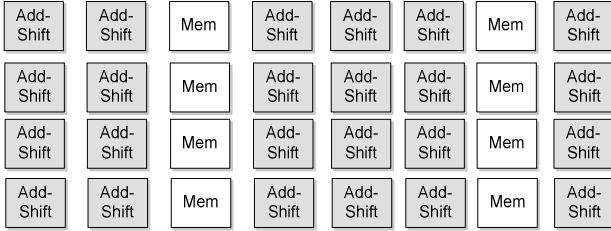


Figure 5: Arrangement of the clusters in the array. More add-shift clusters are used according to the needs.

In the initial measurements presented in this paper, the number of add-shift clusters used is three times more than that of memory clusters. This allows the mapping of a wide range of applications. The arrangement of the clusters in the array is done at design-time and according to the required application and flexibility.

The elements of the array are interconnected with a symmetrical mesh of configurable switches, as described in [1] (see figure 4). Six 8-bit tracks and six 1-bit tracks are provided for both data and control lines. Connection-boxes (C-Boxes) connect the pins of a cluster to the tracks and switch-boxes (S-Boxes) connect together the intersections of tracks [12]. The C-Boxes currently used have a flexibility of $F_c=6$ (as defined in [12]), and the S-Boxes have a flexibility of $F_s=3$. These values were chosen here for simplicity and can be configured by the designer according to the requirements.

5.3. Tri-state buffers as reconfigurable Switches

To make the array synthesizable and adaptable to any process, the configurable switches were implemented as tri-state buffers. Using tri-state buffers instead of pass-transistors significantly increases the area, power consumption and delays [14], however, tri-state buffers make the architecture fit easily with the design-flow used for the rest of the SoC and more customizable at design-time.

6. Measured performance

Two benchmark DCT calculations were implemented on the array to measure its performance. The same DCTs were also implemented using different architectures and compared; The performance results are detailed below along with a comparison of the two DCT implementations and measurement of the added overhead to the implementation.

6.1. Comparison to other systems

The simple 8-point 1-D DCT calculation without memory compression and the DCT with odd-even decomposition described in section 2 and shown in Figures 2 and 3 were implemented on the RA. The same DCTs were also implemented using standard hardwired ASIC and using a commercial Xilinx Virtex-E FPGA [15]. The performance in terms of area, power consumption and maximum frequency is shown in table 2 and table 3. All of these systems use a $0.18\mu\text{m}$ CMOS technology and run at 1.8V and 10MHz. The values are measured for one row only of the array; the result for a full 1D DCT or a 2D DCT would be similar.

Table 2. Performance of the simple DCT implementation

	.18 μm ASIC	<i>Our array</i>	Xilinx's Virtex-E
Area (μm^2)	17 483	212 135	234 510
Power consumption. (mW)	0.52	1.922	3.2
Max Frequency (MHz)	210	77	50

Table 3. Performance of the odd-even DCT implementation

	.18 μm ASIC	<i>Our array</i>	Xilinx's Virtex-E
Area (μm^2)	10 518	235 234	267 725
Power consumption. (mW)	0.48	1.50	2.9
Max Frequency (MHz)	250	77	66

The DCT is implemented using 12-bits input coefficients and 8-bits output coefficients from the LUT, which results in a 16-bit output values. The first DCT without memory compression has been manually mapped such that:

- A 12-bits shift register is mapped to three add-and-shift elements part of one cluster.
- A 2-Kbit memory is mapped to four memory elements found in one cluster.
- A 16-bit shift accumulator is mapped to four add-shift modules part of one cluster.

In the second DCT implementation with odd-even decomposition the mapping was similar to the previous one but with the following differences:

- The 8-bit adder/subtractor at the input is mapped to two add-and-shift elements part of one cluster.

- The 32x8 bit memory is mapped to one 256x8 bits memory element found in one cluster.

It should be noted that the routing and mapping of our array was done manually, and hence, the result is less optimized than that of the Virtex-E which is done automatically. Using intelligent software for placement and routing could improve the results from our array in the future.

6.1.1. Area

The area measured in the case of our array and the Virtex-E excludes the area used by configuration memory, and only includes the logic and reconfigurable interconnects used in order to compare the logic usage. The proposed coarse-grain array requires less configuration bits than a standard FPGA, however, the area per bit is larger when comparing a synthesizable flip-flop to a SRAM cell. The Virtex-E area estimation assumes that a CLB slice [15] and its belonging C-boxes and S-boxes have an area of 3303 μm^2 ; this was measured by implementing an equivalent slice on the uses 0.18 μm technology and estimating the area overhead of the interconnects.

It can be seen that our implementation is 11% to 14% smaller than the Virtex-E one. In both DCT cases, the area consumed by the ASIC implementation stays significantly smaller than the reconfigurable ones.

It should be noted that in the case of the ASIC and the Virtex-E, no programmable memory was used for the LUT. Instead, the LUT was formed using fixed combinatorial logic, which greatly reduces the area when compared to the RAM in our implementation. The RAM also allows more flexibility and easier reprogramming to accommodate for different coefficients.

6.1.2. Power consumption

The power consumption values measured for our array and for the hardwired implementation are obtained with post-routing simulation with typical switching activity and accurate parasitic and load information, using *Cadence Silicon Ensemble* and *Synopsys PrimePower*. In the case of the Virtex-E FPGA, the power consumption is obtained from typical estimations provided by Xilinx. The values provided include the power consumed by the configuration circuit and configuration memory. Our proposed array consumes between 38% and 48% less power than the Virtex-E. This is mainly caused by the fact that less interconnects and larger clusters are used in our array. Nevertheless, the consumption of the reconfigurable implementations remains significantly higher than hardwired ASIC. This is caused by the configurable switches overhead and the use of RAM over hardwired-LUTs.

6.1.3. Timing

Our array has a maximum frequency 16% to 54% higher than the Virtex-E, however this is still 63% to 70% less than the maximum frequency achievable in ASIC. This reduced frequency represents the delays introduced in the reconfigurable switches and the higher-loads and longer routings.

6.2. Comparison of the two DCT algorithms

The odd-even decomposition in the DCT requires less memory due to the smaller LUT; however, an extra adder/subtractor is required per row. This is reflected in the area used by the second implementation, which is 10% higher than the first one.

Power consumption is reduced by 22% in the second implementation due to the fact the adder/subtractor consumes less power than the large memory.

The maximum frequency is the same in both implementations, due to the fact that the largest delay is between the output of the shift-registers and the output of the shift-accumulator, and not at the input. It is also possible to implement the adder/subtractor as bit-serial elements after the shift-register, but this may introduce extra delay.

Similar results are found when comparing the ASIC and the Virtex-E implementations of both DCTs.

6.3. Measurement of overhead

The measured overhead in power and area introduced by the reconfiguration system is similar to the ones found in usual bit-level reconfigurable architectures, such as the one reported in [13]. This is described below.

6.3.1. Power overhead

When modules and clusters are unconfigured and if they have no activity at their inputs, they exhibit only static power consumption and no dynamic one. In the case of unconfigured C-boxes, some switching power is dissipated when the output of the cluster connected to the C-box is switching.

The total static power consumption of the array was measured to be only 0.03% of the total power consumption. Hence we can consider the static power consumption of unconfigured fabric to be negligible when compared to the total power consumption.

Figure 6 shows the total power consumption of one add-shift cluster and its associated C-Box and S-Box. The values shown are the average of both the shift-register and shift-accumulator used in one row of DCT.

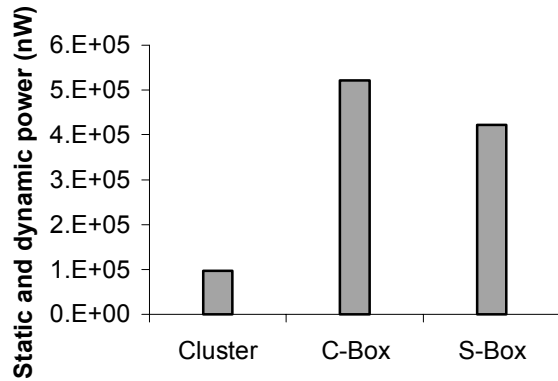


Figure 6: Distribution of the average power consumption between an add-shift cluster and its associated C-box and S-Box. The C- and S-boxes consume around 91% of the total power.

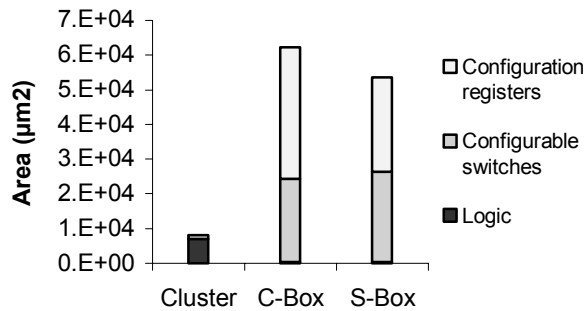


Figure 7: Area of add-shift cluster and its associated C- and S-boxes, and percentage of area occupied by logic, configuration memory and by reconfigurable switches. The boxes occupy a considerable 91% of the whole area.

From the graph it can also be concluded that the power consumed by the cluster is only 9% of the total power, while the C-Box consumes 50% and the S-Box 41%. This is expected due to the high number of switches and buffers introduced in the signals and due to the long routing. This could be improved by reducing the flexibility of the boxes taking into consideration that the flexibility is not decreased greatly [12]. Hence, the next step in future power reductions would be in optimizing the interconnects.

6.3.2. Area overhead

Similarly, figure 7 shows the area overhead used to make the hardware reconfigurable. The add-shift cluster occupies only 6% of the total area while the C- and S-boxes occupy 50% and 44% respectively. As it can be seen from the graph these area values include the area occupied by the configuration registers, which represents a large percentage of the area of the boxes. The total area can be reduced considerably if the flexibility of the C- and S-boxes is lowered: This would reduce the size of the configuration memory as well as area switches.

Using a data coding style to compress the bit-stream in the configuration registers, e.g. usage of a decoder in the C-Boxes to allow connecting a pin to one track only would reduce substantially the number of configuration registers required, while maintaining the same number of configurable switches. This would reduce the area at the expense of removing the option of connecting a pin to multiple tracks.

7. Conclusion

An embedded domain-specific Reconfigurable Array (RA) was introduced. The array is based on two configurable clusters: Add-shift and memory. The elements are arranged in an array with a mesh of reconfigurable interconnects. The reconfigurability of the array permits mapping a number of distributed arithmetic implementations such as DCT and filtering calculations used in video coding.

In this paper, two DCT calculations were implemented on the RA. These were also implemented in standard hardwired ASIC and in standard commercial FPGA. It was demonstrated that the array provides approximately a 40% reduction in power consumption, a decrease of 12% in area and an improvement of about 35% in timing. These figures show that the proposed array architectures provide a good compromise between hardwired ASICs and generic FPGAs in terms of flexibility, area, timing and power consumption when targeting a domain-specific application. This allows adding enough flexibility to ASICs to permit post-fabrication changes without having a large area or power overhead. However, it was measured that up to 91% of the total power and area is consumed by the programmable interconnects between the elements of the array. This is due to the restriction on the use of standard-cell libraries for building synthesizable switches. Hence, future research has to be concentrated on optimized synthesizable interconnects mesh.

8. References

- [1] Khawam S., Arslan T., Westall F., "Embedded reconfigurable array targeting motion estimation applications", *Proceedings of the 2003 IEEE International Symposium on Circuits and Systems (ISCAS'03)*, May 2003, Vol. 2, Page(s): 760-763
- [2] 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*, 12-15 Sept. 2001, Page(s): 169-174
- [3] White, S.A., "Applications of distributed arithmetic to digital signal processing: a tutorial review", *ASSP Magazine*, IEEE, Volume: 6 Issue: 3, Jul 1989, Page(s): 4-19

- [4] Abnous A., Rabaey J.M., "Ultra-low-power domain-specific multimedia processors", *IEEE VLSI Signal Processing*, 1996
- [5] Bryant I., Tanurhan Y., "The Actel Embeddable FPGA Core", *Actel Corporation*, 2001
- [6] Burlleson, 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 DVC*, Vol., 2001, Pages: 4- 12
- [7] Sungwook Yu; Swartzlander, E.E., Jr., "DCT implementation with distributed arithmetic", *IEEE Transactions on Computers*, Vol. 50 Is. 9, Sept. 2001
- [8] Ahmed N., Natarjan T., Rao K.R., "Discrete Cosine Transform", *IEEE Trans. On Computers*, Vol. C-23, No. 1, pp.90-93, December 1984
- [9] Yi Yang; Chunyan Wang; Omair Ahmad, M.; Swamy, M.N.S., "An on-line CORDIC based 2-D IDCT implementation using distributed arithmetic", *Sixth International Symposium on Signal Processing and its Applications*,. 2001, Vol. 1
- [10] Kyeounsoo Kim; Jong-Seog Koh, "An area efficient DCT architecture for MPEG-2 video encoder" ,*Consumer Electronics, IEEE Transactions on* , vol. 45 Issue: 1 , Feb. 1999
- [11] Wilton, S.J.E.; "Embedded memory in FPGAs: recent research results", *Communications, Computers and Signal Processing, 1999 IEEE Pacific Rim Conference on* , 1999 , Page(s): 292 -296
- [12] 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
- [13] George V., Zhang H., Rabaye J, "The design of low energy FPGA, Proceedings", *1999 International Symposium on Low Power Electronics and Design*, pp. 188-193. 1999
- [14] Betz V., Rose J., and Marquardt A., "Architecture and CAD for Deep-Submicron FPGAs", Kluwer Academic Publishers, 1999. ISBN 0-7923-8460-1
- [15] Xilinx, *The Programmable Logic Data Book*, Xilinx Inc., 2001
- [16] Kafafi N., Bozman K., Wilton S.J.E, "Architectures and Algorithms for Synthesizable Embedded Programmable Logic Cores", *ACM International Symposium on FPGA*, Monterey, CA, Feb 2003.
- [17] Leijten-Nowak, K.; Katoch, A, "Architecture and implementation of an embedded reconfigurable logic core in CMOS 0.13 μ m", *ASIC/SOC Conference, 2002. 15th Annual IEEE International* , pp. 3 -7
- [18] Marshall A., Stansfield T., Kostarnov I., Vuillemin J., Hutchings B., "A Reconfigurable Arithmetic Array for Multimedia Applications", *ACM/SIGDA International Symposium on FPGAs*, Monterey, Feb 1999
- [19] eASIC, "eASIC 0.13um Core", www.easic.com