ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

Design of Hybrid and Fully Optimized DSP Architecture for Higher flexibility and Energy Efficiency

J.Angelin Simi, S.Neelaveni

PG Student, Department of Electronics and Communication, Mookambigai College of Engineering, Pudukottai, India

Assistant Professor, Department of Electronics and Communication, Mookambigai College of Engineering, Pudukottai, India

ABSTRACT: Hardware acceleration has been proved an extremely promising implementation strategy for the digital signal processing (DSP) domain. Rather than adopting a monolithic application-specific integrated circuit design approach, in this brief, we present a novel accelerator architecture comprising flexible computational units that support the execution of a large set of operation templates found in DSP kernels. We differentiate from previous works on flexible accelerators by enabling computations to be aggressively performed with carry-save (CS) formatted data. Advanced arithmetic design concepts, i.e., recording techniques, are utilized enabling CS optimizations to be performed in a larger scope than in previous approaches. Extensive experimental evaluations show that the proposed accelerator architecture delivers average gains of up to 61.91% in area-delay product and 54.43% in energy consumption compared with the state-of-art flexible data paths. In paper their task is fully concentrated on 16 bit operation but we focus on 32 bit operations. The polynomial algorithm is used to reduce the time delay which automatically leads the speed increasing ratio. The Multi dimensional signal processing scheme improves the control unit to control the entire mux unit and register bank. A domain-specific architecture generation algorithm is used to improve the acceleration of DSP.

KEYWORDS: Digital Signal Processing (DSP), Hardware Acceleration, Integrated Circuit Design, Carry-Save Formatted Data, Polynomial Algorithm.

I. INTRODUCTION

Modern embedded systems target high-end application domains requiring efficient implementations of computationally intensive digital signal processing (DSP) functions. The incorporation of heterogeneity through specialized hardware accelerators improves performance and reduces energy consumption. Although application-specific integrated circuits (ASICs) form the ideal acceleration solution in terms of performance and power, their inflexibility leads to increased silicon complexity, as multiple instantiated ASICs are needed to accelerate various kernels.

Many researchers have proposed the use of domain-specific coarse-grained reconfigurable accelerators in order to increase ASICs' flexibility without significantly compromising their performance. High-performance flexible datapaths have been proposed to efficiently map primitive or chained operations found in the initial data-flow graph (DFG) of a kernel. The templates of complex chained operations are either extracted directly from the kernel's DFG or specified in a predefined behavioral template library. Design decisions on the accelerator's data path highly impact its efficiency. Existing works on coarse-grained reconfigurable data paths mainly exploit architecture-level optimizations, e.g., increased instruction-level parallelism (ILP).

The domain-specific architecture generation algorithms vary the type and number of computation units achieving a customized design structure. In flexible architectures were proposed exploiting ILP and operation chaining. Recently, Ansaloni et al. adopted aggressive operation chaining to enable the computation of entire sub expressions using multiple ALUs with heterogeneous arithmetic features.

The aforementioned reconfigurable architectures exclude arithmetic optimizations during the architectural synthesis and consider them only at the internal circuit structure of primitive components, e.g., adders, during the logic synthesis.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

However, research activities have shown that the arithmetic optimizations at higher abstraction levels than the structural circuit one significantly impact on the datapath performance. In timing-driven optimizations based on carry-save (CS) arithmetic were performed at the post-Register Transfer Level (RTL) design stage. In common sub expression elimination in CS computations is used to optimize linear DSP circuits. Verma et al. developed transformation techniques on the application's DFG to maximize the use of CS arithmetic prior the actual data path synthesis.

The aforementioned CS optimization approaches target inflexible data path, i.e., ASIC, implementations. Recently, Xydis et al. proposed a flexible architecture combining the ILP and pipelining techniques with the CS-aware operation chaining. However, the entire aforementioned solutions feature an inherent limitation, i.e., CS optimization is bounded to merging only additions/subtractions. A CS to binary conversion is inserted before each operation that differs from addition/subtraction, e.g., multiplication, thus, allocating multiple CS to binary conversions that heavily degrades performance due to time-consuming carry propagations.

II. RELATED WORK

The major part of the project development sector considers and fully survey all the required needs for developing the project. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

Many researchers have proposed in the past systems, the use of domain-specific coarse-grained reconfigurable accelerators in order to increase ASICs' flexibility without significantly compromising their performance. High-performance flexible data paths have been proposed to efficiently map primitive or chained operations found in the initial data-flow graph (DFG) of a kernel. The templates of complex chained operations are either extracted directly from the kernel's DFG or specified in a predefined behavioral template library. Design decisions on the accelerator's data path highly impact its efficiency. Existing works on coarse-grained reconfigurable data paths mainly exploit architecture-level optimizations, e.g., increased instruction-level parallelism (ILP).

A. A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems

Two of the most important design issues for next generation handheld devices are wireless networking and the processing of multimedia content. Both applications rely heavily on computationally intensive digital signal processing algorithms. Programmable architectures that keep pace with the increasing performance requirements become more and more power hungry. This is problematic for a battery powered mobile device, since it has only a limited amount of energy available. Conversely, dedicated architectures are too inflexible to keep pace with changing standards and feature sets. A mobile device requires high-performance, flexibility and (energy-)efficiency. These contradicting requirements need to be balanced in the system architecture of a mobile device. In this paper a heterogeneous architecture of domain specific processing tiles is proposed. The focal point is the coarse-grained reconfigurable architecture of the Montium processing tile, which is designed to execute digital signal processing algorithms energy-efficiently.

B. ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix

The coarse-grained reconfigurable architectures have advantages over the traditional FPGAs in terms of delay, area and configuration time. To execute entire applications, most of them combine an instruction set processor(ISP) and a reconfigurable matrix. However, not much attention is paid to the integration of these two parts, which results in high communication overhead and programming difficulty. To address this problem, we propose a novel architecture with tightly coupled very long instruction word (VLIW) processor and coarse-grained reconfigurable matrix. The advantages include simplified programming model, shared resource costs, and reduced communication overhead. To exploit this architecture, our previously developed compiler framework is adapted to the new architecture. The results show that the new architecture has good performance and is very compiler-friendly.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

C. A high-performance data path for synthesizing DSP kernels

A high-performance data path to implement digital signal processing (DSP) kernels is introduced in this paper. The data path is realized by a flexible computational component (FCC), which is a pure combinational circuit and it can implement any 2 times 2 template (cluster) of primitive resources. Thus, the data path's performance benefits from the intracomponent chaining of operations. Due to the flexible structure of the FCC, the data path is implemented by a small number of such components. This allows for direct connections among FCCs and for exploiting intercomponent chaining, which further improves performance. Due to the universality and flexibility of the FCC, simple and efficient algorithms perform scheduling and binding of the data flow graph (DFG). DSP benchmarks synthesized with the FCC data path method show significant performance improvements when compared with template-based data path designs. Detailed results on execution time, FCC utilization, and area are presented.

D. Automatic design of reconfigurable domainspecific flexible cores

Reconfigurable hardware is ideal for use in systems-on-a-chip (SoC), as it provides both hardware-level performance and post-fabrication flexibility. However, any one architecture is rarely equally optimized for all applications. SoCs targeting a specific set of applications can greatly benefit from incorporating customized reconfigurable logic instead of generic field-programmable gate-array (FPGA) logic. Unfortunately, manually designing a domain-specific architecture for every SoC would require significant design time. Instead, this paper discusses our initial efforts towards creating a reconfigurable hardware generator capable of automatically creating flexible, yet domain-specific, designs. Our tests indicate that our generated architectures are more than 5times smaller than equivalent FPGA implementations and nearly as area-efficient as standard cell designs. We also use a novel technique employing synthetic circuit generation to demonstrate the flexibility of our architecture generation techniques.

E. Designing coarse-grain reconfigurable architectures by inlining flexibility into custom arithmetic data-paths

This system introduces a design technique for coarse-grained reconfigurable architectures targeting digital signal processing (DSP) applications. The design procedure is analyzed in detail and an area-time-power efficient reconfigurable kernel architecture is presented. The proposed technique inlines flexibility into custom carry-save (CS) arithmetic data paths exploiting a stable and canonical interconnection scheme. The canonical interconnection is revealed by a transformation, called uniformity transformation, imposed on the basic architectures of CS-multipliers and CS-chain-adders/subtractors. Experimental results including quantitative and qualitative comparisons with existing reconfigurable arithmetic cores and exploration results of the proposed reconfigurable architecture are provided.

III. PROPOSED SCHEME

In this system, we propose a high-performance architectural scheme for the synthesis of flexible hardware DSP accelerators by combining optimization techniques from both the architecture and arithmetic levels of abstraction. We introduce a flexible data path architecture that exploits CS optimized templates of chained operations. The proposed architecture comprises flexible computational units (FCUs), which enable the execution of a large set of operation templates found in DSP kernels. The proposed accelerator architecture delivers average gains of up to 61.91% in area-delay product and 54.43% in energy consumption compared to state-of-art flexible data paths, sustaining efficiency toward scaled technologies.

A. Carry-Save Arithmetic: Motivational Observations And Limitations

CS representation has been widely used to design fast arithmetic circuits due to its inherent advantage of eliminating the large carry-propagation chains. CS arithmetic optimizations rearrange the application's DFG and reveal multiple input additive operations (i.e., chained additions in the initial DFG), which can be mapped onto CS compressors. The goal is to maximize the range that a CS computation is performed within the DFG. However, whenever a multiplication node is interleaved in the DFG, either a CS to binary conversion is invoked or the DFG is transformed using the distributive property. Thus, the aforementioned CS optimization approaches have limited impact on DFGs dominated by multiplications, e.g., filtering DSP applications. In this brief, we tackle the aforementioned limitation by exploiting the CS to modified Booth (MB) recoding each time a multiplication needs to be performed within a CS-optimized data path. Thus, the computations throughout the multiplications are processed using CS arithmetic and the operations in the targeted data path are carried out without using any intermediate carry-propagate adder for CS to binary conversion, thus improving performance.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

B. Proposed Flexible Accelerator

The proposed flexible accelerator architecture is shown in Fig. 1. Each FCU operates directly on CS operands and produces data in the same form1 for direct reuse of intermediate results. Each FCU operates on 16-bit operands. Such a bit-length is adequate for the most DSP data paths, but the architectural concept of the FCU can be straightforwardly adapted for smaller or larger bit-lengths. The number of FCUs is determined at design time based on the ILP and area constraints imposed by the designer. The CS to Bin module is a ripple-carry adder and converts the CS form to the two's complement one. The register bank consists of scratch registers and is used for storing intermediate results and sharing operands among the FCUs. Different DSP kernels (i.e., different register allocation and data communication patterns per kernel) can be mapped onto the proposed architecture using post-RTL data path interconnection sharing techniques. The control unit drives the overall architecture (i.e., communication between the data port and the register bank, configuration words of the FCUs and selection signals for the multiplexers) in each clock cycle.

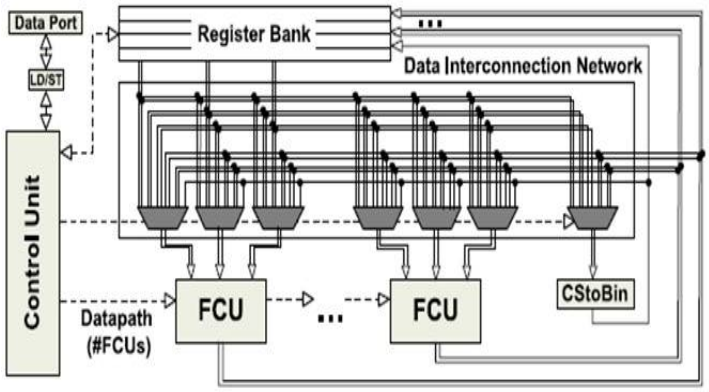


Fig.1 Abstract form of the Flexible Data Path

C. Structure of the Proposed Flexible Computational Unit

The structure of the FCU (Fig. 2) has been designed to enable high-performance flexible operation chaining based on a library of operation templates. Each FCU can be configured to any of the T1–T5 operation templates shown in Fig. 3. The proposed FCU enables intra-template operation chaining by fusing the additions The FCU is able to operate on either CS or two's complement formatted operands, since a CS operand comprises two 2's complement binary numbers.

The following relation holds for all CS data: $X* = \{XC, XS\} = XC + XS$. The operand A is a two's complement number. The alternative execution paths in each FCU are specified after properly setting the control signals of the multiplexers MUX1 and MUX2 (Fig. 2).

The multiplexer MUX0 outputs $Y*$ when CL0 = 0 (i.e., $X* + Y*$ is carried out) or $Y*$ when $X* - Y*$ is required and CL0 = 1. The two's complement 4:2 CS adder produces the $N* = X* + Y*$ when the input carry equals 0 or the $N* = X* - Y*$ when the input carry equals 1. The MUX1 determines if $N*$ (1) or $K*$ (2) is multiplied with A. TheMUX2 specifies if $K*$ (1) or $N*$ (2) is added with the multiplication product. The multiplexer MUX3 accepts the output of MUX2 and its 1's complement and outputs the former one when an addition with the multiplication product is required (i.e., CL3 = 0) or the later one when a subtraction is carried out (i.e., CL3 = 1). The 1-bit ace for the subtraction is added in the CS adder tree.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

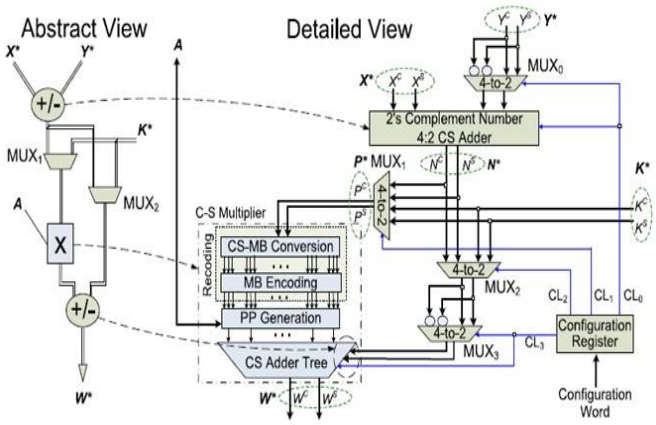


Fig.2 FLU

The multiplier comprises a CS-to-MB module, which adopts a recently proposed technique to recode the 17-bit P∗ in its respective MB digits with minimal carry propagation. The multiplier's product consists of 17 bits. The multiplier includes a compensation method for reducing the error imposed at the product's accuracy by the truncation technique. However, since all the FCU inputs consist of 16 bits and provided that there are no overflows, the 16 most significant bits of the 17-bit W∗ (i.e., the output of the Carry-Save Adder (CSA) tree, and thus, of the FCU) are inserted in the appropriate FCU when requested.

IV. ARCHITECTURAL DESIGN

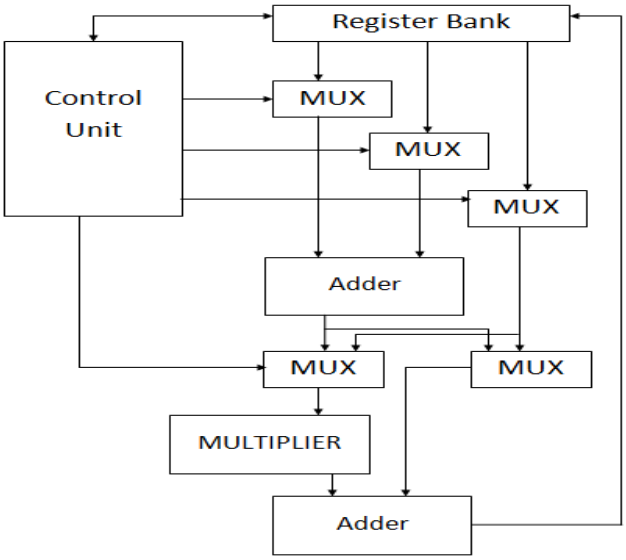


Fig.3 Block Diagram

The major part of the project development sector considers and fully survey all the required needs for developing the project. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations. Generally algorithms shows a result for exploring a single thing that is either be a performance, or speed, or accuracy, and so on. An architecture description is a formal description and representation of a system, organized in a

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them.
Their task is fully concentrated on 16 bit operation but we focus on 32 bit operations. The polynomial algorithm is used to reduce the time delay which automatically leads the speed increasing ratio. The Multi dimensional signal processing scheme improves the control unit to control the entire MUX unit and register bank. A domain-specific architecture generation algorithm is used to improve the acceleration of DSP. Two sets of MUX principles are used in this system for performing different ALU operations.
The first MUX is selected based on the select line which will get the data input data A and B from the register bank, which is stored in some particular memory regions. The Control Unit which controls the register bank. Based on the select line the MUX generates any one of the ALU output, that output is feeded back to the second MUX. The select line is control by using Control Unit and as well as the K value is given to MUX. All the input values are 32-bit. The resultant W output from the MUX is stored in register bank address name it as RB-Write / Register Bank-Write. In this system, we propose a high-performance architectural scheme for the synthesis of flexible hardware DSP accelerators by combining optimization techniques from both the architecture and arithmetic levels of abstraction. We introduce a flexible datapath architecture that exploits CS optimized templates of chained operations.
The proposed architecture comprises flexible computational units (FCUs), which enable the execution of a large set of operation templates found in DSP kernels. The proposed accelerator architecture delivers average gains of up to 61.91% in area-delay product and 54.43% in energy consumption compared to state-of-art flexible data paths, sustaining efficiency toward scaled technologies.
Each FCU operates directly on CS operands and produces data in the same form1 for direct reuse of intermediate results. Each FCU operates on 16-bit operands. Such a bit-length is adequate for the most DSP data paths, but the architectural concept of the FCU can be straightforwardly adapted for smaller or larger bit-lengths. The number of FCUs is determined at design time based on the ILP and area constraints imposed by the designer. The CS to Bin module is a ripple-carry adder and converts the CS form to the two's complement one. The register bank consists of scratch registers and is used for storing intermediate results and sharing operands among the FCUs. Different DSP kernels (i.e., different register allocation and data communication patterns per kernel) can be mapped onto the proposed architecture using post-RTL data path interconnection sharing techniques.
The control unit drives the overall architecture (i.e., communication between the data port and the register bank, configuration words of the FCUs and selection signals for the multiplexers) in each clock cycle. In order to efficiently map DSP kernels onto the proposed FCU-based accelerator, the semiautomatic synthesis methodology presented has been adapted. At first, a CS-aware transformation is performed onto the original DFG, merging nodes of multiple chained additions/subtractions to 4:2 compressors. A pattern generation on the transformed DFG clusters the CS nodes with the multiplication operations to form FCU template operations.
The designer selects the FCU operations covering the DFG for minimized latency. Given that the number of FCUs is fixed, a resource-constrained scheduling is considered with the available FCUs and CS to Bin modules determining the resource constraint set. The clustered DFG is scheduled, so that each FCU operation is assigned to a specific control step. A list-based scheduler has been adopted considering the mobility2 of FCU operations. The FCU operations are scheduled according to descending mobility. The scheduled FCU operations are bound onto FCU instances and proper configuration bits are generated. After completing register allocation, a FSM is generated in order to implement the control unit of the overall architecture.

V. EXPERIMENTAL RESULTS

A. Circuit-Level Exploration of the Proposed FCU With Respect to Technology Scaling

A circuit-level comparative study was conducted among the proposed FCU, the flexible computational component (FCC) and the reconfigurable arithmetic unit (RAU) in scaled technology nodes. The CS representation requires twice the number of bits of the respective two's complement form, thus, increasing wiring and affecting performance in scaled technologies. This experimentation targets to show that the scaling impact on the performance does not eliminate the benefits of using CS arithmetic. The three units considered were described in RTL using Verilog. The CSA tree of the proposed FCU and the adders and multipliers of the FCC were imported from the Synopsys Design Ware library. We used Synopsys Design Compiler to synthesize the examined units and the TSMC 130, 90, and 65 nm standard cell libraries. We synthesized each unit with the highest optimization degree at its critical clock period and 20

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

higher ones with a step interval of 0.10 ns. Fig. 5 reports the area complexity of the evaluated units at 130, 90, and 65 nm of synthesis technology.

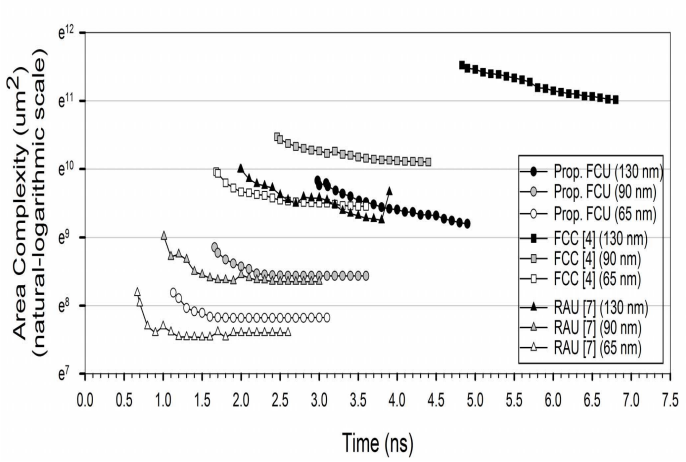


Fig.4 Area Time Diagram of FCU

At 130 nm, the proposed FCU, the FCC, and the RAU operate without timing violations starting at 2.98, 4.83, and 1.99 ns, respectively. At 90 nm, the proposed FCU, the FCC, and the RAU are timing functional starting at 1.66, 2.46, and 1.01 ns, respectively. In addition, at 65 nm, the proposed FCU, the FCC, and the RAU start operating without timing violations at 1.13, 1.68, and 0.67 ns, respectively. As the synthesis technology is scaled down, Fig. 5 shows that the proposed FCU outperforms the FCC in terms of critical delay and area complexity, but presents larger values for these metrics than the RAU in all the technology nodes. However, RAU's flexible pipeline stage (FPS) features limited ability in carrying out heavy arithmetic operations as shown from the mega operations per second/watt (MOPS/W) evaluation in Fig. 5.

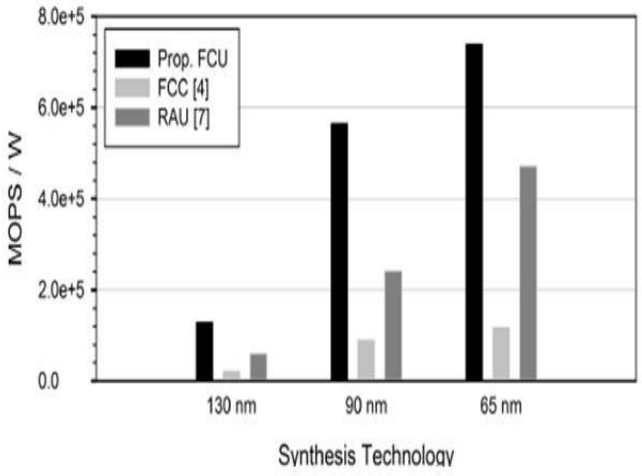


Fig.5 MOPS/W Values of FCU at the lowest achievable clock periods with respect to the Synthesis Technology

Fig. 5 shows the MOPS/W values for the proposed FCU, the FCC, and the RAU at their critical clock periods with respect to the synthesis technology. For each unit, we consider the templates with the largest number of active operations without overlapping the one another and calculate the average ((#Ops)/(#Cycles)) factor. The #Ops derives from the two's complement arithmetic operations (additive or multiplicative). For CS-aware units, i.e., FCU and RAU, the CS to Bin module runs in parallel with the FCU or RAU, thus counting as one additive operation. In particular, for

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

the proposed FCU, we consider the templates T1 (or T2) with six operations (e.g., five operations from T1 and one operation from CS to Bin) in one cycle and T3 with five operations in one cycle.
Thus, ((#Ops)/(#Cycles))FCU = 11/2. For the FCC, we consider only the full load case of four operations in one cycle resulting in ((#Ops)/(#Cycles))FCC = 4. In addition, for the RAU, we consider the template of five additive operations that are carried out in one cycle, and the template of one multiplication that needs four cycles. Thus, ((#Ops)/(#Cycles))RAU = 21/8. The clock frequency ClkFreq is the highest one that each unit achieves (i.e., the critical clock period). The power values for computing the MOPS/W ones have been extracted by simulating each unit with Modelsim for 216 random inputs and using the Synopsys Primetime-PX with the average calculation mode triggered. As shown, the proposed FCU delivers average MOPS/W gains across technology nodes of 6× and 2× over FCC and RAU, respectively. Thus, as the synthesis technology is scaled down, the benefits of the proposed FCU remain.

B. Mapping of DSP Kernels onto the Proposed FCU-Based Architecture

We examined the efficiency of the proposed solution by mapping and accelerating DSP kernels onto the FCU data path, that is: 1) a sixth-order ELLIPTIC filter; 2) a 16-taps Finite Impulse Response filter (FIR16); 3) a nonlinear IIR VOLTERRA filter; 4) an 1-D unrolled Discrete Cosine Transform (DCT) kernel (UDCT); 5) the 2-D JPEG DCT (JPEGDCT); and 6) the 2-D Inverse MPEG DCT (MPEG_IDCT). The kernels were scheduled and mapped onto the architectures based on the proposed FCU, the FCC, and the RAU. The proposed architecture comprises four FCUs and one CS to Bin module, the FCC-based one contains two FCCs (= 8 ALUs + 8 Multipliers), and the RAU-based architecture consists of four FPSs and one CS to Bin module. For each kernel mapped, the maximum memory bandwidth has been allocated between the local storage and the processing elements. All the data paths were synthesized at 65 nm. The clock periods were specified at 1.60, 2.20, and 1.20 ns for the proposed FCU-, the FCC-, and the RAU-based architectures, respectively, considering the critical delays of Section V-A plus a slack of 0.50 ns for absorbing any delays inserted by the multiplexers and control units. Energy consumption values were calculated through PrimeTime-PX after simulating each data path with Modelsim.
Table I reports the execution latency, area complexity, and energy values of the DSP kernels mapped onto the examined architectures. The execution latency is the total number of cycles multiplied by the clock period for the synthesis of each architecture.

TABLE I

EXECUTION LATENCY, AREA COMPLEXITY, AND ENERGY FOR DSP KERNELS MAPPED ONTO THE PROPOSED FCU, THE FCC, AND THE RAU AT 65 nm

Kernel	Prop. FCU			FCC [4]			RAU [7]			A×L Gain over FCC (%)	E Gain over FCC (%)	A×L Gain over RAU (%)	E Gain over RAU (%)
	L[a]	A[b]	E[c]	L	A	E	L	A	E				
ELLIPTIC	9.6	21636	198.72	13.2	41226	348.48	14.4	22042	234.72	61.83	42.98	34.56	15.34
FIR16	9.6	21150	182.40	17.6	17387	297.44	19.2	25975	391.68	33.65	38.68	59.29	53.43
VOLTERRA	8.0	19385	144.00	15.4	38299	264.88	25.2	21069	337.68	73.71	45.64	70.79	57.36
JPEGDCT	209.6	42803	5240.00	301.4	59897	7625.42	520.8	47588	14009.52	50.30	31.28	63.80	62.60
MPEG_IDCT	216.0	41525	3218.40	286.0	59382	6292.00	580.8	46136	11906.40	47.19	48.85	66.53	72.97
UDCT	212.8	17387	3553.76	281.6	40100	4111.36	625.2	25169	10128.24	67.23	13.56	76.49	64.91
Average	-	-	-	-	-	-	-	-	-	55.65	36.83	61.91	54.43

[a] Execution latency in ns. [b] Area complexity in um^2. [c] Energy consumption in pJ.

The average latency gains of the proposed FCU-based architecture over the ones built on the FCC and the RAU are 33.36% and 56.69%, respectively. Regarding the area complexity, the proposed FCU-based flexible data path delivers average gains of 31.75% and 13.23% over the FCC- and RAU-based solutions, respectively. As shown, different kernels demand different area resources due to the differing needs regarding the number of scratch registers and multiplexers, as well as the complexity of the control unit (i.e., the more cycles a mapped kernel needs for execution,

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016

the more complex the control unit). Table I also reports the metrics of the area-delay product and the energy providing a clear view of the beneficial characteristics of the proposed approach and allowing us to safely conclude that the proposed architecture forms a very efficient solution for DSP acceleration.

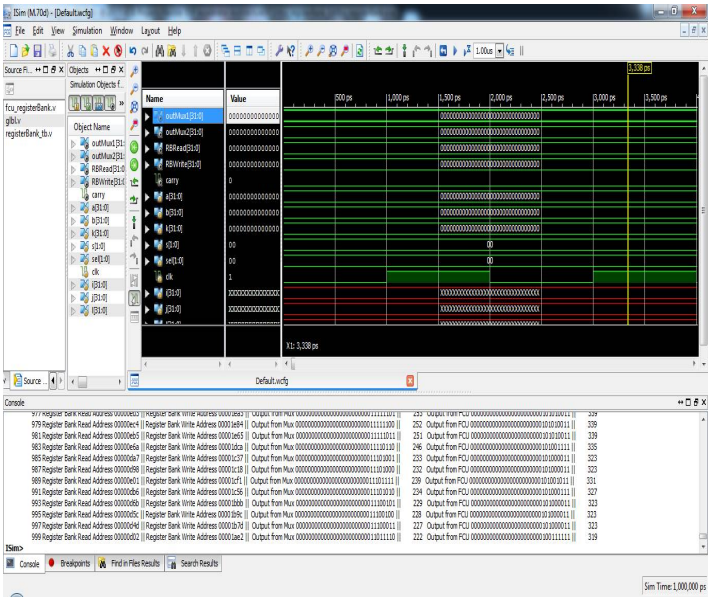
TABLE II
THEORETICALLY ESTIMATED EXECUTION LATENCY AND AREA COMPLEXITY FOR DSP KERNELS

Kernel	Prop. FCU		FCC [4]		RAU [7]	
	L^a	A^b	L	A	L	A
ELLIPTIC	258	14572	534	30056	360	27642
FIR16	258	14860	712	27176	480	26850
VOLTERRA	215	14092	623	28136	630	24666
JPEGDCT	5633	22420	12193	36806	13020	33912
MPEG_IDCT	5805	23572	11570	36806	14520	33768
UDCT	5719	14284	11392	29624	15630	28044

[a] Execution latency in time equivalents (T_g).
[b] Area complexity in area equivalents (A_g).

Table II shows the theoretically estimated values for the execution latency and area complexity of the DSP kernels mapped onto the examined architectures. Regarding both the execution latency and the area complexity and considering all the DSP kernels, the proposed FCU-based architecture outperforms the ones built on the FCC and the RAU. As expected, the timing constraints and the effects of cell sizing implied by the Design Compiler synthesis tool, in some cases result in inconsistencies between the experimental and the theoretical studies, e.g., in Table I, the latency of ELLIPTIC kernel on FCC is more efficient than the one on RAU, but in Table II, the RAU-based ELLIPTIC kernel outperforms the one based on the FCC. In any case, both the experimental and theoretical analysis indicated that the proposed approach forms the most efficient architectural solution.

VI. SAMPLE SCREENSHOTS

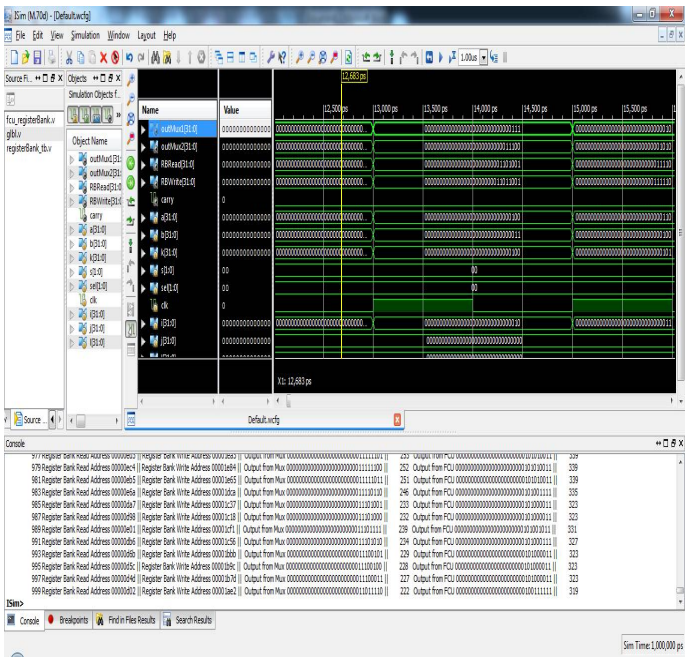


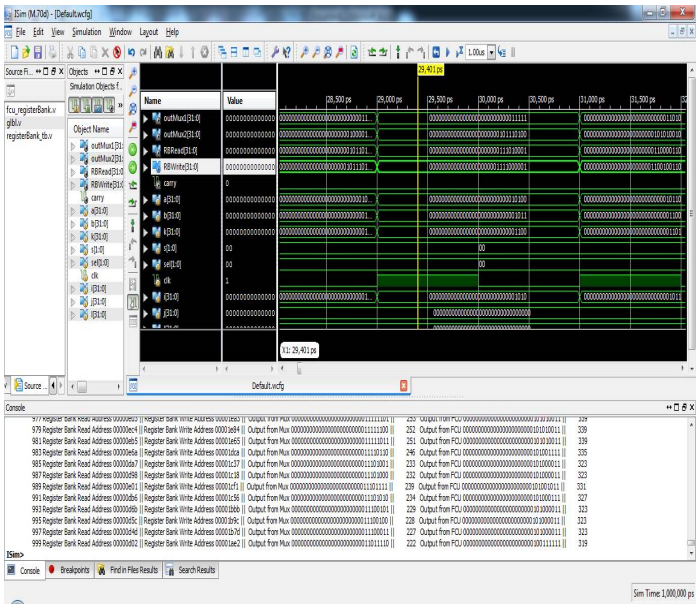
ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016



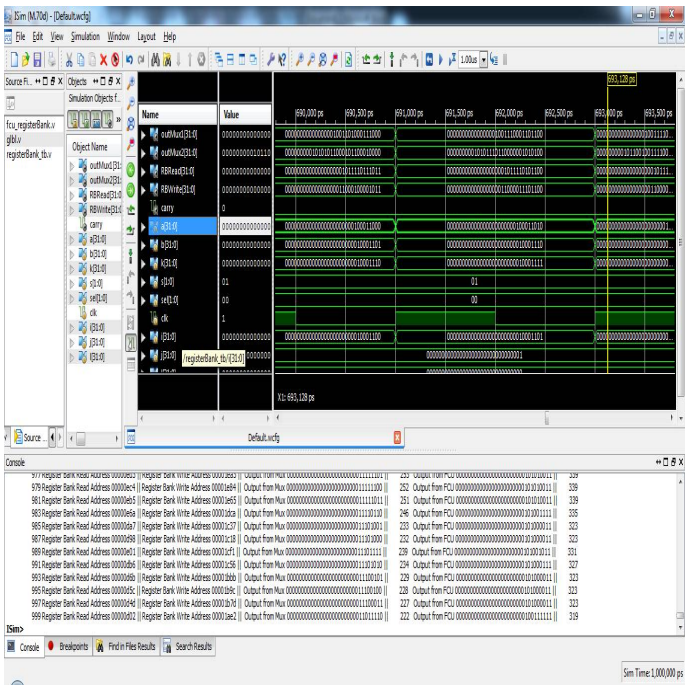


ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 2, February 2016



VII. CONCLUSION

We introduced a flexible accelerator architecture that exploits the incorporation of CS arithmetic optimizations to enable fast chaining of additive and multiplicative operations. The proposed flexible accelerator architecture is able to operate on both conventional two's complement and CS-formatted data operands, thus enabling high degrees of computational density to be achieved. Theoretical and experimental analyses have shown that the proposed solution forms an efficient design tradeoff point delivering optimized latency/area and energy implementations.

REFERENCES

[1] P. Ienne and R. Leupers, Customizable Embedded Processors: Design Technologies and Applications. San Francisco, CA, USA: Morgan Kaufmann, 2007.
[2] P. M. Heysters, G. J. M. Smit, and E. Molenkamp, "A flexible and energy-efficient coarse-grained reconfigurable architecture for mobile systems," J. Supercomput., vol. 26, no. 3, pp. 283–308, 2003.
[3] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in Proc. 13th Int. Conf. Field Program. Logic Appl., vol. 2778. 2003, pp. 61–70.
[4] M. D. Galanis, G. Theodoridis, S. Tragoudas, and C. E. Goutis, "A high-performance data path for synthesizing DSP kernels," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 25, no. 6, pp. 1154–1162, Jun. 2006.
[5] K. Compton and S. Hauck, "Automatic design of reconfigurable domainspecific flexible cores," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 5, pp. 493–503, May 2008.
[6] S. Xydis, G. Economakos, and K. Pekmestzi, "Designing coarse-grain reconfigurable architectures by inlining flexibility into custom arithmetic data-paths," Integr., VLSI J., vol. 42, no. 4, pp. 486–503, Sep. 2009.
[7] S. Xydis, G. Economakos, D. Soudris, and K. Pekmestzi, "High performance and area efficient flexible DSP datapath synthesis," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 3, pp. 429–442, Mar. 2011.
[8] G. Ansaloni, P. Bonzini, and L. Pozzi, "EGRA: A coarse grained reconfigurable architectural template," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 6, pp. 1062–1074, Jun. 2011.