FleXilicon: a Reconfigurable Architecture for Multimedia and Wireless Communications

Jong-Suk Lee and Dong Sam Ha
VTVT (Virginia Tech VLSI for Telecommunications) Lab
Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, VA 24061
E-mail: {watsup, ha}@vt.edu

Abstract— **This paper proposes a new reconfigurable architecture for multi-media and wireless communications. The proposed architecture addresses three critical design issues with the loop level parallelism, wide memory bandwidth, reconfigurable controller, and the support of flexible word-length. Several major functions of multimedia and wireless communication applications were implemented in SystemC to estimate the performance of the proposed architecture. Simulation results indicate that the proposed architecture performs far better than conventional processors.**

I. INTRODUCTION

Multimedia and wireless communication applications demand high computing power, flexibility, and scalability. The ASIC solution meets the high computing power requirement, but is inflexible and not scalable. On the other hand, general purpose microprocessors or DSP are flexible, but often fail to provide sufficient computing power. Since early 1990s, reconfigurable architectures have been proposed as a compromise between the two extreme solutions, and been applied for multimedia and wireless communications [1].

Some critical loop operations in multimedia and wireless communications usually consume a good portion of the total execution cycles. Therefore, the key issue in implementing multimedia or wireless algorithms onto a reconfigurable architecture is to map critical loops into processing elements optimally to meet the computing need. Two major techniques for efficient execution of loops for reconfigurable architectures are pipelining and loop level parallelism (LLP). The pipelining technique, which is widely employed, achieves high throughput and it does not require reconfiguration at the clock cycle level. The LLP technique was investigated initially for parallel computing machines such as supercomputers and multi-processor systems, and it simply executes loop operations concurrently with multiple processing elements [2].

A reconfigurable architecture has evolved from fine-grained to coarse-grained architecture [1]. This paper concerns only coarse grained architectures due to some major advantages such as efficient area, high performance and low power [3]. Existing coarse-grained architectures can be categorized into two groups, datapath-oriented and instruction-oriented architectures, based on the types of executions performed by underlying processing elements. A processing element for a datapath-oriented architecture executes only one type of operation once it is configured, and a required dataflow is constructed by routing mesh structured processing elements. To implement the LLP on a datapath-oriented architecture, the body of the loop is replicated on mesh, and multiple iterations are executed concurrently in a pipeline manner. However, it does not lead to high resource utilization when (1) I/Os from/to processing elements are limited and (2) a data flow does not fit into a given mesh topology. In addition, the degree of parallelism in LLP is limited. REMARC [3], MATRIX [4], MorphoSys [5], and PACT XPP [6] belong to this group.

In contrast, a processing element of an instruction-oriented architecture performs sequence of operations, which are defined by instructions, micro-code and/or control signals. Instructions are stored in a configuration memory and fetched by a controller to control the associated processing element. As a processing element can execute the entire body of a loop, employment of the LLP is simply to assign multiple processing elements running concurrently. So, an instruction-oriented architecture leads to high resource utilization and are more suitable than the former type of architectures for multimedia and wireless communications. Existing reconfigurable architectures belong to this group include RAW [7], PADDI [8], Chameleon [9], and AVISPA [10]. Although instruction-oriented architectures are suitable for multimedia and wireless communication applications, there are several shortcomings for the existing reconfigurable machines. To mitigate these problems, we propose a new instruction-oriented reconfigurable architecture called FleXilicon.

II. PRELIMINARIES

In this section, we present three major design issues for instruction-oriented reconfigurable architectures, and the proposed architecture intends to address the issues.

A. Memory bandwidth

Unlike a datapath-oriented architecture which requires memory access at a constant rate, an instruction-oriented architecture requires high memory accesses at certain time

instants. During those peak cycles, processing elements with a limited memory bandwidth should wait until the necessary data is available from the memory. Therefore, a wide memory bandwidth is a critical design issue to achieve a high degree parallelism for the LLP, which is often the bottleneck for high performance for instruction oriented architectures. However, existing instruction oriented architectures such as PADDI [8], Chameleon [9], AVISPA [10] are inherently not suitable for a wide bandwidth, since the number of processing elements exceeds available memory I/Os. Our architecture employs a new memory system, which guarantees necessary operand access from local memory for processing elements.

B. *Controller design*

Memory based controllers determine the type of operation to be performed for processing elements in instruction-oriented architectures. The sequencer of a controller generates global instructions, which, in turn, select VLIW-like configurable instructions from the memory to execute multiple processing elements. Existing memory based controllers have several shortcomings as described in the following. First, the entry of an instruction memory is too small in some of existing architectures (such as PADDI [8] and Chameleon [9] which have only eight entries for the memory). Therefore, when a single iteration requires more than eight instructions, the instruction memory should be reconfigured to cause serious performance degradation. Second, controllers are localized and hence cannot be shared among PEs, even if all PEs have the same functionality. Finally, a memory based controller is not suitable for an instruction pipeline control since each pipeline stage requires access of different memory locations. Hence, it necessitates a large size memory for super-pipeline stages. Unlike other instruction oriented architectures, RAW [7] uses a microprocessor as a processing element. Hence, the instructions are fetched and decoded to execute operations as a conventional microprocessor does, which results in area overhead for instruction cache, instruction fetch logic and decoder logic.

C. *Sub-word parallelism*

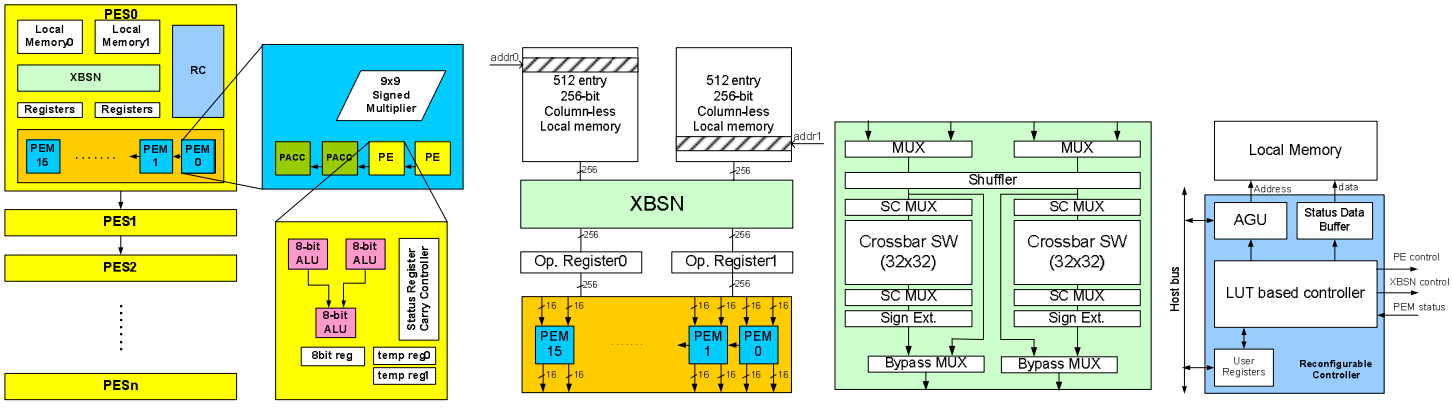
Sub-word parallelism (SWP) is a method to increase the parallelism by partitioning the datapath into sub-word, so that multiple sub-word data can be processed concurrently [11]. Various algorithms in multimedia and wireless communication applications require data with different precisions. For example, audio algorithms usually require a high resolution ranging from 16 bits to 24 bits, while video algorithms from 8 bits to 16 bits. Wireless communication algorithms have a wide range of the precision from 4 bits to 32 bits. Therefore, the SWP is effective for parallel processing of data with various precisions in multimedia and wireless communication applications. However, only a few of reconfigurable architectures adopt the SWP in a limited fashion. PADDI [8] supports 32-bit additions by concatenating 16-bit EXU blocks. Chameleon [9] supports two 16-bit additions and single 32-bit additions. Note that mesh structured architectures are not suitable for the SWP due to complex interconnections among processing elements.

III. PROPOSED RECONFIGURABLE ARCHITECTURE

In this section, we present the overall architecture of the proposed FleXilicon, which intends to address the above problems.

A. *Overall architecture*

Fig 1 (a) shows the overall architecture of FleXilicon. FleXilicon has an array of processing elements slices (PESs), which may be concatenated for scalability. Different outer loops or independent multi-threads can be assigned to different PESs. One PES consists of local memories, XBSN (crossbar switch network), 16 PEMs (processing element and multipliers) and a RC (reconfigurable controller). A PES is the basic block for execution of multiple iterations of a loop in parallel. The number of iterations of a loop which can be executed concurrently on a PES depends on the type of the operations. Local memories store the input and output data streams to read from or to write onto the host processor and PEMs. The XBSN supports various types of memory



(a) Overall architecture (b) PES (c) XBSN (d) Controller

Figure 1. FleXilicon Architecture

accesses and flexible word length operations. The reconfigurable controller is responsible for generating control signals of the local memories, the XBSN and 16 PEMs. One PEM consists of two PEs (processing elements), two PACCs (partial accumulators), and one 9-bit multiplier, and a PEM can perform single 8x8 MAC operations or two 8-bit ALU operations. A PE consists of three 8-bit ALUs, three 8-bit data registers, and a status registers, and is responsible for actual processing of data. A PE supports various operations including ALU operations and configurable application specific operations such as ACS (Add Compare Select), SAD (Sum of Absolute Difference), weighted sum and clipping operations. Other application specific operations can be added through reconfiguration of the datapaths of PEs, which can reduce execution cycles for loop processing.

B. *Processing Elemente Slice (PES)*

Like other instruction-oriented architectures, the proposed architecture has a 1-D array of PESs. A PES is the basic processing unit for the LLP execution, and its structure is shown in Figure 1 (b). To meet the peak demand on memory access, the PES has two 16 kB (256x512) column-less memories. The local memory has a 256-bit wide input/output port without a column decoder, and thus it accesses an entire row data on a single clock cycle. Sense amplifiers are used as a register file, which loads 64 8-bit data concurrently from the memory. To support various types of memory accesses necessary for multimedia and wireless communication applications and to minimize the communication overhead among PEs, each PES includes an XBSN as shown in Fig 1(c). An XBSN includes two 32x32 8-bit crossbar switches, so any 8-bit word of 32 operands (fetched from a memory) can be available to each operand register. In other words, 64 8-bit operands are fetched from the two memories on a clock cycle and available to the two operand registers, in turn, the 16 PEMs. So that the two PEs of a PEM are guaranteed to receive two operands, which addresses the peak memory access problem faced for other existing architectures. An XBSN is a key block of the architecture.

C. *Reconfigurable controller*

Our controller generates control signals for local memories, the XBSN, and PEMs for each instruction pipeline stage. To mitigate the problem in existing memory based controllers explained earlier, FleXilicon employs a reconfigurable controller which is a fine-grained LUT (look-up table) block like an FPGA. Unlike previous memory based controllers, the proposed controller implements for a finite state machine (FSM) to generate control signals. Since combinational logic in FSM can be minimized through logic optimization technique, more functionality than the memory based controller can be provided in given area. In addition, a controller can be shared across multiple processing elements through flexible implementation of a FSM, which reduces the overall hardware complexity. Fig 1 (d) shows the controller block, in which an address generation unit (AGU) is a pre-designed block and generates control signals for access of a local memory. The host processor communicates with the controller through user registers to initiate the controller or to retrieve the status of the controller and PESs. The status buffer stores the status of PEMs.

D. *Sub-word parallelism with flexible word-length support*

FleXilicon supports flexible word-length operations – addition/subtraction, shift, and multiplication. For addition/subtraction operations, multiple 8-bit PEs can simply concatenate to construct a higher precision, where each PE configures as a carry selection adder to minimize the critical path delay of a long word-length addition/subtraction. For shifting operations, the XBSN provides various word-length parallel shift operations – 8 bit, 16 bit and 32 bit. In the XBSN, scramble multiplexer performs bit scrambling operation so that the crossbar switch can arrange the bit position according to shift amount, and multiple shifted data can be obtained by de-scrambling the output of crossbar switch. A new method is proposed to provide flexible word-length multiplications. Since any MAC (multiplication and accumulation) operations can be expressed with low-precision MACs, FleXilicon provides various types of MACs using 8x8 atomic MAC units, PEMs. The XBSN divides multiplicands into 8-bit chunks and feeds them to appropriate PEM units with sign extension. While this method requires additional sum cycles for summation of partial products, parallelized executions of multiplications can speed up MAC operations.

IV. PERFORMANCE OF FLEXILICON

To estimate the performance of the proposed architecture, we modeled an SOC system embedding FleXilicon architecture using SystemC. We implemented a Viterbi decoder, a 16x16 SAD operation in MPEG4 video, a GSM pulse shaping filter, an MP3 cosine filter, and a DFT block for GPS (Global Positioning System) correlation on single PES using the SystemC model. We compared the performance with ARM 920T, and TI 320C64xx DSP [12].

Table 1 shows simulation results for a Viterbi decoder and a 16x16 SAD operation. Execution cycles for the Viterbi decoder are for the update of a single stage of a state metric. Execution cycles for the 16x16 SAD operation are for calculation of SAD for a 16x16 macro-block. FleXilicon reduces the number of cycles significantly for a Viterbi decoder, by 1892 times over ARM 920T and by 94 times over TI 320C64xx. The reduction ratio is much less for the SAD operation, but FleXilicon is still less than the other two platforms. Note that the speedup of FleXilicon is even more impressive in terms of execution time.

Table II shows the comparison results for three different filter operations. Compared to TI 320C64xx, FleXilicon reduces the number of cycles by 4.8 times for the GSM, 2.7 times for the MP3, and 11.3 times for the GPS. The reduction ratios are much larger over ARM 920T. In GPS, FleXilicon requires only 0.06 cycles per single MAC

operation, while TI 320C64xx and ARM 920T 0.71 cycles and 7.14 cycles, respectively.

TABLE I. PERFORMANCE COMPARISON I

		Viterbi decoder		16x16 SAD	
	Frequency	*Cycles*	*Execute Time(us)*	*Cycles*	*Execute Time(us)*
FleXilicon	200 MHz	12	0.06	13	0.07
ARM 920T	200 MHz	22706	113.53	5,493	27.47
TI C64xx	600 MHz	3399	5.66	74	0.12

TABLE II. PERFORMANCE COMPARISON II

	GSM		MP3		GPS	
	Cycles	*Cycles/ MAC*	*Cycles*	*Cycles/ MAC*	*Cycles*	*Cycles/ MAC*
FleXilicon	3890	0.15	4619	0.25	2565	0.06
ARM920T	181860	7.21	61417	4.44	292747	7.14
TI64xx	18589	0.74	12497	0.9	28925	0.71

TABLE III. FPGA MAPPING RESULT OF CONTROLLER

		FPGA Resource		
	# logic gates	*Slice*	*F/F*	*LUT*
Viterbi decoder	115.9	9	10	16
16x16 SAD	76.8	8	9	14

Table III shows FPGA implementations of Viterbi and SAD controllers. The controllers were designed in SystemC and were synthesized and mapped into a Xilinx Virtex2 FPGA with user-defined reconfigured instructions sets and manually designed FSM. As shown in the results, both controllers can be implemented with around 100 gates or less and can be mapped into FPGA only using less than 3% of resources in the Virtex2 xc2v40 which is the smallest version of Virtex2.

The above simulation results show the superiority of FleXilicon architecture over conventional architectures and the efficiency of FleXilicon for both multimedia and wireless communication algorithms. It is important to note that the results were obtained for single PES, and the performance will increase in proportion to the number of PESs. Finally, the estimated target operating frequency for FleXilicon is 200MHz in 0.18-µm CMOS process, while TI 320C64xx DSPs were fabricated in 0.13-µm CMOS process. FleXilicon will achieve additional speedup with a more advanced CMOS process.

V. CONCLUSION

In this paper, we propose a new architecture, which mitigates shortcomings of existing architectures in wide memory bandwidth, controller and flexible word-length. A wide bandwidth memory system enables a high degree of loop level parallelism. The proposed reconfigurable controller resolves the inflexibility and inefficiency of existing memory based controllers. Flexible word-length enhances sub-word parallelism. Several representative types of applications were implemented, and simulation results demonstrate the superiority and efficiency of the proposed architecture.

REFERENCES

[1] R. Hartenstein, "A Decade of Reconfigurable Computing: a Visionary Retrospective", Design, Automation and Test in Europe, Conference and Exhibition Proceedings, 2001.

[2] C.D. Polychronopoulous, et al., "Utilizing multidimensional loop parallelism on large-scale parallel processor systems", IEEE Transactions on Computers, Vol. C-38, No. 9, September 1989.

[3] T. Miyamori, K. Olukotun, "A Quantative Analysis of Reconfigurable Comprocessors for Multimedia Applications", IEEE Symposium on FPGA for Custom Computing Machines, April 1998, pp. 2- 11.

[4] E. Mirsky, A. DeHon, "MATRIX: a reconfigurable computing architecture with configurable instruction distribution and deployable resources", Proceedings. IEEE Symposium on FPGA for Custom Computing Machines 1996.

[5] H. Singh, et al., "MorphoSys: An Integrated Reconfigurable System for Data-Poarallel and Computation-Intensive Applications", IEEE Trans. On Computers, Vol. 49, No. 5, 2000.

[6] V. Baumagarten, et. al., "PACT XPP – a self-reconfigurable data processing architecture", Journal of Supercomputing, 2003, pp. 167-184.

[7] M.B. Taylor, et al., "The Raw microprocessor: a computational fabric for software circuits and general purpose programs", IEEE Micro, Vol. 22, Issue 2, March-April 2002, pp. 25-35.

[8] D.C. Chen, J.M. Rabaey, "A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic-Specific High-Speed DSP Data Paths", IEEE Journal of Solid-State Circuits, Vol. 27, No. 12, December 1992, pp. 1895-1904.

[9] B. Salefski, L. Caglar, "Re-Configurable Computing in Wireless", Proceedings of Design Automation Conference, 2001, pp. 178-183.

[10] J. Leijten, J. Huisken, E. Waterlander, A. V. Wel, "AVISPA: A Massively Parallel Reconfigurable Accelerator", Proceedings of International Symposium on System-on-Chip, 2003, pp. 165-168.

[11] J. Fridman, "Sub-word Parallelism in Digital Signal Processing", IEEE Signal Processing Magazine, March 2000.

[12] S. Agarwala, et al., "A 600-MHz VLIW DSP", IEEE Journal of Solid-state Circuits, Vol 37, No. 11, November 2002.