

A Domain Specific Reconfigurable Viterbi Fabric for System-On-Chip Applications

Cheng Zhan¹, Tughrul Arslan^{1,2}, Sami Khawam¹, Iain Lindsay¹

¹ School of Electronics and Engineering

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

c.zhan@ed.ac.uk, Sami.Khawam@ee.ed.ac.uk, Tughrul.Arslan@ee.ed.ac.uk, Iain.Lindsay@ed.ac.uk

Abstract - A novel embedded dynamically reconfigurable fabric for implementing the Viterbi algorithm in a System-on-Chip device is presented in this paper. The proposed reconfigurable fabric can support Viterbi implementations for different standards, such as GSM, IS-95, CDMA and Wireless LAN. Our results illustrate that the proposed architecture has superior power consumption and throughput characteristics and it is demonstrated a 80% reduction in power consumption over generic field programmable gate array (FPGA) and 40 times improvement in throughput over digital signal processor (DSP), respectively. Thus, the reconfigurable system-on-chip platform based on this kind of domain specific reconfigurable fabrics is an efficient solution for the high-performance portable communication systems.

I. INTRODUCTION

Future mobile communication systems will not only offer the same old services, e.g. GSM and IS-95 with improved quality, but also new services, and hence they are required to be more flexible, as well as providing a low power consumption and high throughput. With the progress of semiconductor technologies and the increasing complexity of the silicon devices, the concept of "Reconfigurable-System-on-Chip" (RSoC) has become a reality. Compared with other design methods, the RSoC platform with domain specific reconfigurable fabrics [1] [2] provides an efficient solution for the future portable communication systems. This is because such domain specific reconfigurable fabrics can be tailored to accommodate the requirements and operational constraints of the specific systems.

The Viterbi algorithm [3] is commonly used for decoding the convolutional code, a widely used channel coding techniques in today's digital communication systems. The configuration of the convolutional codes is disparate in each communication standard which imparts different requirements on the Viterbi decoder. The global system for mobile communication (GSM) standard uses a Viterbi decoder with the constraint length of 5, and a rate of 1/2. The wireless LAN (WLAN) standard specifies a constraint length of 7, and a rate of 1/2. The Viterbi decoder for third generation (3G) standard has a specification of constraint length of 9 and a rate of 1/2 or 1/3. A reconfigurable Viterbi decoder with low power consumption and high throughput is a key challenge for future portable devices. In this paper, our proposed domain specific reconfigurable fabric exhibits a superior compromise of flexibility, power-consumption and throughput when compared to ASIC, FPGAs and DSPs.

The paper is organized as follows: In section 2 describes the reconfigurable system-on-chip platform. The domain specific reconfigurable fabric for Viterbi decoder is addressed in section 3 and its performance is assessed in section 4.

II. SYSTEM OVERVIEW

The proposed reconfigurable SoC platform contains a general purpose micro-processor, DSP, a number of domain specific reconfigurable modules, as well as some glue logics and ASIC parts, shown in the Figure 1.

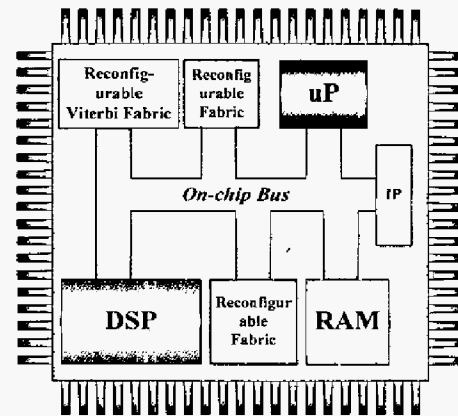


Figure 1 Reconfigurable System-on-Chip

The different parts communicate through the system-on-chip bus, such as the AMBA bus. Since a DSP incurs the significant overhead of fetching, decoding and executing a stream of instructions which means most of the clock cycles and power are wasted in fetching the instruction and setting the appropriate control signals. So some signal processing algorithms can be handled in the domain specific fabric to achieve high throughput rate and low power consumption. The data to be processed is sent to the domain specific reconfigurable fabric by DSP and reads back to the DSP via on-chip bus. The programming of the reconfigurable fabric is done by the micro-processor. By downloading the different reconfigurable bits to the configurable memory, the processor can configurable the different Viterbi implementations on the reconfigurable fabric during the run-time.

III. RECONFIGURABLE FABRIC FOR VITERBI DECODER

A. Viterbi Algorithm

The Viterbi algorithm is known to be an efficient method to perform the maximum likelihood sequence detection of the convolutional codes. It can be viewed as a technique to find the shortest path metric in the trellis diagram. The computation of Viterbi algorithm is an iterative operation. The most computationally intensive steps in each iteration are as follows:

- Branch metric (BM) computation
- Add-compare-select operation to update the path metric (PM) of each trellis state and generate decision bits
- Decision bits store and output decoding sequence

The detailed processing units of this proposed domain specific reconfigurable fabric will be talked about below.

B. Branch Metric Unit

The BM computation [4] is to calculate the distance between the received signals and the ideally transmitted signals at each trellis stage. In specific application, the BM calculation diagram is varied with convolutional code rate R (R bits are transmitted for each bit in the message, $R-1$ of them being parity bits). Since the BM for code rate $1/2$ will be used in calculating the BM for code rate $1/3$, we do not need to design two different hardware circuits for each code rate. We can reuse the code rate $1/2$ BM calculation circuit to decrease the power and area overhead.

C. Add-Compare-Select Unit

The path metric update for each new trellis state according to the following equation:

$$PM(i)_{t+1} = \min_{\text{all possible } j} (PM(j)_t + BM(j,i))$$

Where the $PM(i)_{t+1}$ and $PM(j)_t$ are corresponding to the path metric of state i at time stage $t+1$ and state j at time stage t , respectively. The $BM(j,i)$ is associated with the branch metric from state j to state i which is generated by the BM unit. So we call, this kind of data-dependent feedback loop, add, compare and select (ACS) process. In order to increase the modularity and decrease the complexity of the reconfigurable design, we combine two ACS operations together, named *butterfly operation*. In our reconfigurable architecture each *butterfly operation* will be implemented in one butterfly processing unit, shown in figure 2. From the trellis structure of different constraint length k , we find that one path metric input varies with k and another is fixed. In this way, by using the 7-input multiplexer, we can reconfigure the path metric inputs for each butterfly unit. Meanwhile, according to the intrinsic characteristic of the convolutional coding generator polynomials, the branch metrics $BM_{(i)}$ and $BM_{(j)}$ are always as the input signal together, i refer to the codeword of the each branch metric. For instance, when the code rate is $1/2$, the branch metrics are divided into four input groups, BM_{00} with BM_{11} , BM_{01} with BM_{10} , BM_{10} with BM_{01} and BM_{11} with BM_{00} . One 8-to-2 look up table (LUT) can be used to select the branch metrics input for each butterfly processing unit.

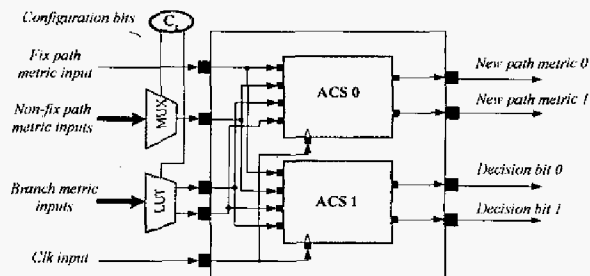


Figure 2 Butterfly processing unit

D. Survivor Management Unit

Each processing cycle, in addition to update two PM s for next iteration, every butterfly processing unit offers two decision bits d_t^i and d_t^{i+1} , where d_t^i represents the decision bit of trellis state i at time cycle t , and the decision vector, $d_t^{2^k-1} d_t^{2^k-2} \dots d_t^0$, which is a set of decision bits, one per trellis state, at time cycle t , will be stored in the same decision memory location. The final objective of the Viterbi decoder is to exploit the decision vector to reconstruct the trellis path. The previous trellis path state S_{t-1}^k given by the current path state S_t^k according to the following update [5]:

$$S_{t-1}^k = d_t (S_t^k \gg 1)$$

which corresponds to a right shift of the current state introducing the value of surviving bit d_t in the vacant position. Since the length of S_t^k is equal to $k-1$, the size of the right shift register need to vary with constraint length k , we use seven 2-to-1 multiplexers to implement this varying sized right shift register, shown in figure 3.

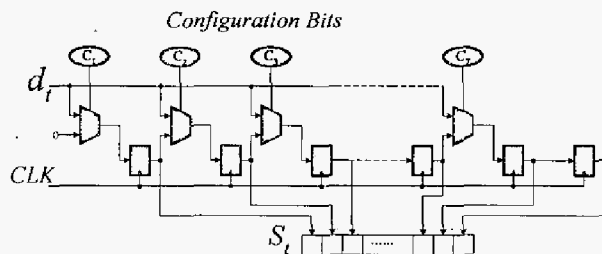


Figure 3 Reconfigurable Right Shift Register

Meanwhile, the whole decision memory is partitioned into a write region, and a read region which is divided into the trace-back part and decision part. Four independent same size memory blocks are exploited to implement the decision memory. This architecture allows four processes, Write, Trace-back, Decision and Idle, to parallel access a different memory block at the same time. The structure and operation of the memory blocks are shown in figure 4.

In addition, in order to reduce the employ of the on-chip bus and save the memory access time, we design the memory unit, which is inside the reconfigurable fabric rather

than the on-chip memory to store and read these decision vectors.

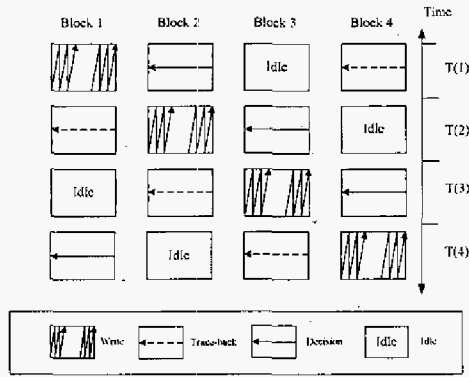


Figure 4 The structure and operation of the memory blocks

The 32x4bits dual-port RAM is used as the basic element to buildup the big RAM through the connect boxes, which is similar to the one found in [6]. By using the input reconfigurable bits to configure the connect box, the neighbor basic RAM elements can be combined together or not. For example, we can use four basic RAM elements with three connection boxes to implement the 64x8bits RAM, which is shown in figure 5. Using this RAM array, we can compose the different RAM size for different constraint lengths.

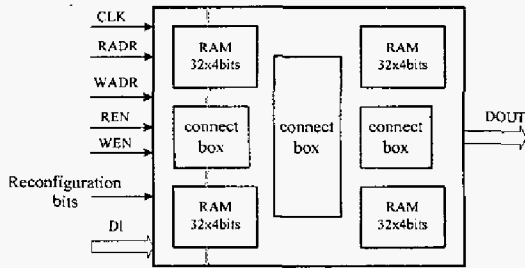


Figure 5 64x8 configurable RAM

E. Power Controller

In our architecture, most of computations are triggered by the rising edge of the clock. When the clock input of a certain unit is reduced to zero, this unit is effectively shut down. When the constraint length is less than 9, not all butterfly processors are being utilized. We exploit tri-buffers to turn off the clock input of the unused units. When the tri-buffer is disabled, the frequency of the clock input is zero, thus neither the inputs, nor the outputs can toggle, and the non-used butterfly units can not consume power at all. For instance, only a quarter of butterfly units are active when the constrain length is 7, if shut down the clock of the unused components, nearly 75% power consumption will be saved.

F. Configuration Memory Unit

This memory stores the configuration bits which are used to program the switch of multiplexers, LUTs and the processing units. Like FPGA, if want to implement different version of the Viterbi decoder on our fabric, we need to load the configuration bits to the configuration memory. FPGAs

[7] [8] use SRAM cells to keep the configuration bits and a serial bits stream of configuration information to update the SRAM cells. Because of our synthesizable fabric, we use latch composing shift register chain to implement the configuration memory. Most FPGAs do need to rewrite the whole configuration memory, even when only a small part needs to be changed. In order to improve the efficiency of the domain specific reconfiguration, we adopt the partial reconfiguration method [9]. In this method, the configuration memory is divided into frames and each frame is uniquely addressable. Using separate address information, allows for selective updating a part of the configuration memory. The adoption of this partial reconfiguration can avoid writing the whole reconfigurable data to the configurable memory unit when only parts of the whole architecture requires reprogramming and the clock cycle and power consumption of the reconfiguration can be saved.

IV. IMPLEMENTATION RESULTS

This domain specific reconfigurable Viterbi decoder fabric is targeted on a UMC 0.18um CMOS technology library. For comparative results, we also implement the non-reconfigurable architecture on standard UMC0.18um ASIC technology and on Xilinx Virtex-E FPGA (XCV300e-8-PQ240). All of these devices use the 0.18um CMOS technology and run at 1.8V and 20MHz. Table 1(a) and (b) show the power consumption and area of an ASIC and our reconfigurable architecture, respectively. The power consumptions for logic and memory have been separated in order to give an in-depth analysis of this reconfigurable core. We can see that the memory power consumptions have a dominant effect on the whole design. The RAM array in this application is very efficient to less the power consumption when the constraint length is less than 9. After comparing the ASIC and our reconfigurable architecture, we can see that the power and the area of our architecture are around 5.4 times and 1.5 times bigger than that of ASIC.

Table 1 (a) Reconfigurable architecture

Constrain Length K	Power (mW)		Size (μm^2)
	Logic	Memory	
3	1.09	1.33	46571
5	1.55	10.64	103689
7	5.27	42.94	335265
9	20.13	169.67	1192911

Table 1 (b) Non-reconfigurable architecture

Constrain Length K	Power (mW)		Size (μm^2)
	Logic	Memory	
3	0.49	1.33	32539
5	0.94	1.70	68193
7	3.39	5.51	221443
9	12.52	34.17	784919

Table 2 illustrates the results of the non-reconfigurable Viterbi implementation on the Xilinx Virtex-E family, and the power consumption in this table do not include the quiescent power (Quiescent power = 540 mW). We can clearly deduce that the power consumption is 5 times more than that of our architecture. Because of the power figure for the FPGA, the reconfigurable SoC platform based on embedded FPGAs is unsuitable for the low power requirement of future portable devices. From Figure 6, we can clearly see the difference of power consumption among the non-reconfigurable ASIC, our reconfigurable architecture and Xilinx FPGA. 80% power reduction is achieved in our architecture in comparison to Xilinx FPGA. Compared with ASIC, our architecture consumes 5.4 times more power as the tradeoff for the flexibility.

Table 2 Power consumption on Xilinx FPGA

Constrain Length K	Power (mW)	Max Freq. (MHz)	# LUT	# FF
3	22.97	105.2	434 (7%)	193 (3%)
5	61.02	100.1	1045 (17%)	381 (6%)
7	213.17	83.7	4351 (70%)	1174 (19%)

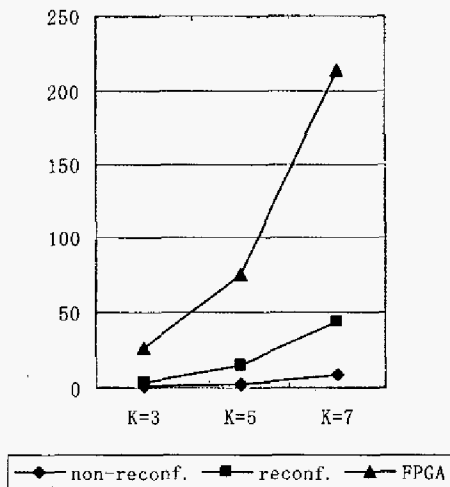


Figure 6 Power comparison among non-reconfigurable, reconfigurable and FPGA architecture

The Viterbi algorithm was also can implemented on a TI TMS320C6416T DSP with a Viterbi coprocessor customized for 3G wireless infrastructures [10]. This processor can provide a maximum of 5Mbps decoding throughput. However, the maximum frequency of our architecture can reach up to 200MHz or a decode rate of 200Mbps, which is 40 times faster than DSP. By analyzing the results, we can see that the generic DSP and the FPGA cannot achieve the high throughput and low power consumption requirements for the future portable devices. Our reconfigurable SoC platform with the domain specific reconfigurable fabric can better satisfy the flexible, low power consumption and high throughput requirement of future communication system.

V. SUMMARY

We have addressed a novel domain specific reconfigurable Viterbi decoder architecture as a reconfigurable core for integration into a reconfigurable SoC platform. The core provides high flexibility as well as good power and timing performance characteristics for future portable devices. By loading the different reconfiguration bits to the configuration memory, the status of switch, interconnection components and processing units can be reconfigured and the different version of the Viterbi decoder can be implemented on this architecture.

After comparasion with Xilinx FPGA and TI DSP, we find that the power consumption is reduced considerably by around 80% over FPGA and the throughput improved 40 times over DSP. These results confirm that the proposed reconfigurable SoC platform based on domain specific reconfigurable fabrics is very suitable for a high-performance communication system.

REFERENCES

- [1] C. Zhan, S. Khawam, T. Arslan, "Domain Specific Reconfigurable Fabric Targeting Viterbi Alogorithm," 3rd International Conference on Field Programmable Technology. December 2004
- [2] 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
- [3] A.J Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Informaton Theory*, vol. IT-13, pp.260-269, April 1967
- [4] J.Proakis, Digital Communications, 3rd ed. McGraw Hill, 2001
- [5] P.J.Black and T.H.Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE Journal of Solid-State Circuit*, vol. 27, no.12, pp.1877-1885, December 1992
- [6] S.J.E., Wilton; "Embedded memory in FPGAs: recent research results", *Communications, Computers and Signal Processing*, 1999 IEEE Pacific Rim Conference on, 1999, Page(s):292-296
- [7] Xilinx, The Programmable Logic Data Book, Xilinx Inc., 1994, 2001
- [8] Altera, The Data Book, Altera Inc., 1998
- [9] S. Goldstein et al. "PipeRench: A Reconfigurable Architecture and Compiler," *IEEE Computer Magazine*, Vol. 33, No. 4, pp. 70 -77, April 2000.
- [10] TI TMS320C6416T DSP data sheet, <http://www.ti.com>