# Providing flexibility in a convolutional encoder

Kamuf, Matthias; Anderson, John B; Öwall, Viktor

*Published in:*
Proceedings of the 2003 IEEE International Symposium on Circuits and Systems (Cat. No.03CH37430)

*DOI:*
10.1109/ISCAS.2003.1205959

2003

# PROVIDING FLEXIBILITY IN A CONVOLUTIONAL ENCODER

*Matthias Kamuf*[1], *John B. Anderson*[2], *and Viktor Öwall*[1]

[1]Department of Electroscience and [2]Department of Information Technology
Lund University, SE-221 00 Lund, Sweden
{mkf, vikt}@es.lth.se, anderson@it.lth.se

## ABSTRACT

In future radio systems, flexible coding and decoding architectures will be required. In case of the latter, implementing architectural flexibility with regard to low power issues is a challenging task. The flexible encoding platform in this paper is a first step toward this envisioned decoder. It generates a wide class of codes, starting with convolutional codes. As an extension to this, turbo codes will be included by adding an interleaver. At this prototyping stage, the system is implemented on an FPGA. The decision to choose the observer canonical form is defended by a thorough investigation of its critical path properties. Proper configuration allows code rates of $b/c, b = 1 \ldots 15, c = 2 \ldots 16, b < c$. Power can be saved by shutting down unused system modules.

## 1. INTRODUCTION

Wireless personal area networks (WPANs) which provide short-range ad hoc connectivity among different communication devices will be both multi-standard and multi-rate. A gadget in this envisioned environment must be flexible enough to be able to establish communication for a wide variety of applications, reaching from low bit-rate sensor networks to high bit-rate WLAN communication as depicted in Figure 1. Since these devices will be mostly battery-operated, efficient power and energy management is essential for realization. However, there are several orders of magnitude in power efficiency between a flexible software solution and a fully hardware mapped one. Clearly, there will always be a trade-off between flexibility, processing performance and power consumption.

Channel coding and decoding algorithms are key components in a communication system and will vary accordingly due to the mentioned application variety. Although these algorithms have different properties, there are still common blocks that can be identified. Hence, one has to find an efficient way to reuse hardware for different types of tasks in these algorithms. This project aims at a decoder that should be able to handle a predefined set of algorithms by sole configuration of its hardware with suitable parameters. These parameters include, for example, precision, speed, and type of code. As a result, the grade of achieved flexibility could be measured based on the limits in performance and power consumption, indicating a range of feasible solutions.

This paper presents an FPGA-based flexible convolutional encoder which supports power saving by shutting down unused parts through specific enable logic. Care has been taken that the design can be easily upgraded for future purposes; for example, by adding an interleaver block, this design turns into a turbo encoder.

In Section 2 several possible architectures are introduced with a careful look at critical path properties. Section 3 presents the system specification and implementation aspects will be addressed in Section 4. Synthesis and performance results are then outlined in Section 5. The paper ends with a conclusion and a look at future research tasks.

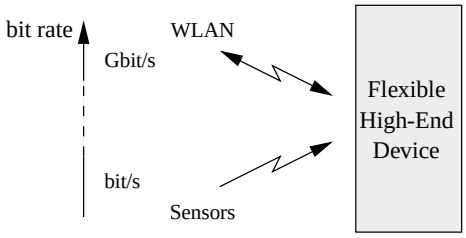


Figure 1: Communication scenario in a WPAN

## 2. ARCHITECTURAL ISSUES

From coding theory [1] it is known that the output of a convolutional encoder at time $k$ is defined by

$$v_k = \sum_{i=0}^{m} f_i \cdot w_{k-i}, \tag{1}$$

where

$$w_k = u_k + \sum_{i=1}^{m} q_i \cdot w_{k-i}. \tag{2}$$

The coefficients $f_i, q_i \in \{0; 1\}$ determine the feedback and feed-forward polynomials, respectively. By definition, $q_0$ is set to 1.

A common way to describe a convolutional code visually is depicted in Figure 2 and is called controller canonical form. Looking at this architecture from an implementation's perspective it becomes obvious that the critical path, which depends on the number of combinational circuits in series, is a function of the respective coding polynomial used. This length can be described by

$$L_{crit} = \max\{m - h(\mathbf{f}); f_0 \cdot m + g(\mathbf{q})\} \tag{3}$$

where

$$g(\mathbf{x}) = \max\{i \mid x_i \neq 0\} + 1 - \delta_{im} \tag{4}$$

and

$$h(\mathbf{x}) = \min\{i \mid x_i \neq 0\} - 1 + \delta_{i0}. \tag{5}$$

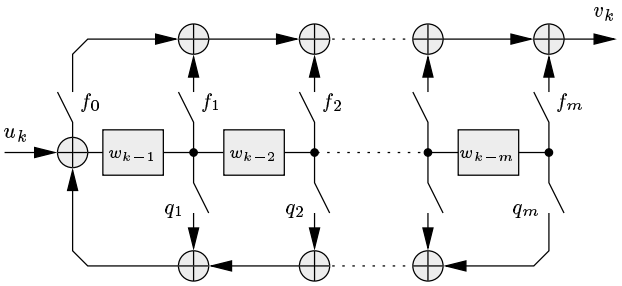


Figure 2: Controller canonical form of a convolutional encoder

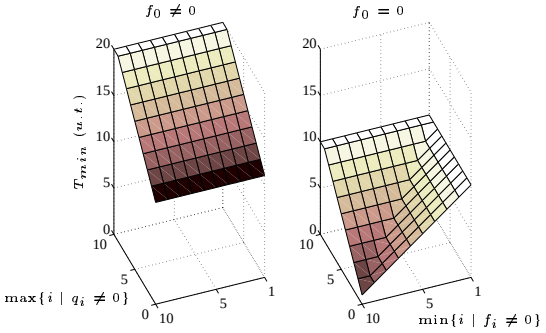


Figure 3: Clock period versus coding polynomial, architecture according to Figure 2, $m = 10$

The Kronecker symbol $\delta_{ij}$ is defined as

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & \text{otherwise} \end{cases}$$

and is used as a correction factor for the case when there are two coefficients connected to the same cell which must not cause additional delay. If a unit time (u.t.) is the propagation delay of a basic XOR-cell the minimum clock period is simply $T_{min} = L_{crit}\ u.t.$.

Figure 3 visualizes $T_{min}$ as a function of the respective coding polynomial, that is, $T_{min}(\mathbf{f}, \mathbf{q})$. According to (4) and (5), the x- and y-axis show the indices of the feedback and feedforward polynomials resulting from the maximum and minimum functions, respectively. The minimum clock period $T_{min}$ is then drawn on the z-axis. By looking at these graphs it is obvious that this architecture is very sensitive to setting coefficient $f_0$ to 1. In this case, the complete feedforward path contributes to the critical path. Hence, the longest path occurs when $f_0$ and either $q_{m-1}$ or $q_m$ are set to 1, resulting in a minimum clock period of $2m$ unit times.

Reordering the cells in Figure 2 according to the law of associativity to form the architecture depicted in Figure 4 [2] modifies the length of the critical path to

$$L_{crit} = \max\{g(\mathbf{f}); f_0 + g(\mathbf{q})\}. \tag{6}$$

Both feedforward and feedback path compete now with each other instead of contributing concurrently to the critical path. In fact, this dramatically decreases the impact of $f_0$ compared to the original controller form. However, in this case coefficients with higher indices will have a larger weight. The minimum clock period in this case is reduced to $(m+1)$ unit times when both $f_0$ and either $q_{m-1}$ or $q_m$ are 1, as shown in Figure 5.

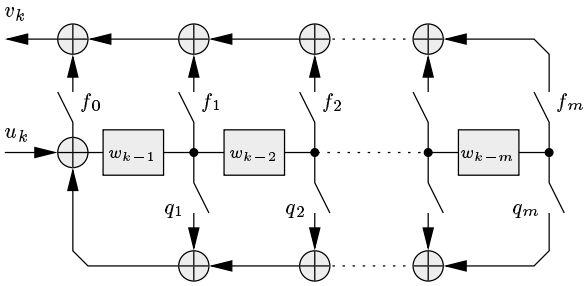


Figure 4: Reordered controller canonical form

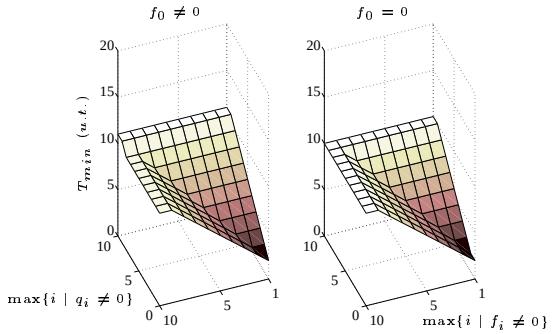


Figure 5: Clock period versus coding polynomial, architecture according to Figure 4, $m = 10$

Cutset retiming [2], as depicted in Figure 6, can still improve this architecture. An $n$-retimed version diminishes the minimum clock period to $\lceil \frac{m+1}{n+1} \rceil$, $n = 1 \ldots m-1$ along with increasing the number of registers by $n$. However, retiming makes sense only up to the point where $T_{min}$ becomes 2 unit times which is the lower bound of this circuit and is determined by the two cells that are connected by $f_0$.

Another way of describing the same system behavior can be found by transforming the signal-flow graph in Figure 2 according to transposition. The directions of all edges are reversed, and input and output nodes are exchanged, while the respective edge delay remains the same [2]. Thus, the output can be written as

$$v_k = f_0 u_k + \sum_{i=1}^{m} (f_i u_{k-i} - q_i v_{k-i}). \tag{7}$$

Since addition and subtraction are equivalent operations in $\mathbb{F}_2$, the realization of (7) is represented in Figure 7 and is called observer canonical form. In this case, the delay elements do not form a shift register anymore since they are separated by modulo-2 adders.

Contrary to the preceding canonical forms, the worst case critical path of the observer canonical form does not depend on the coding polynomial, since this path, given a recursive function, always consists of two XOR-cells in series. However, the drawback is that a single XOR-cell in this architecture will have a larger propagation delay since the operation has to be performed on two logic levels. If the propagation delay of this 3-input cell is two times the delay of the mentioned basic XOR-cell, the minimum clock period will always be 4 unit times.

Concluding these considerations, it is clear that the observer canonical form guarantees a critical path that is independent of

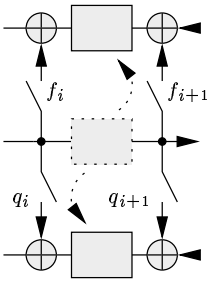


Figure 6: Cutset retiming

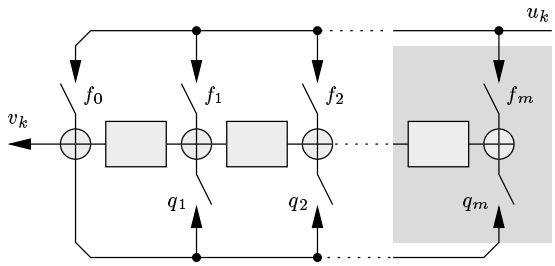


Figure 7: Observer canonical form of a convolutional encoder

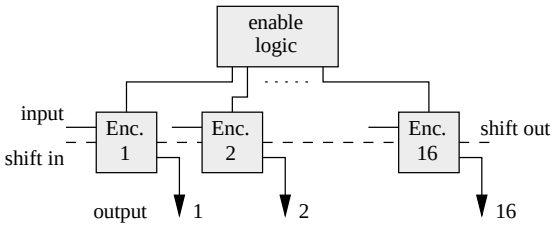


Figure 8: Flexible encoder architecture

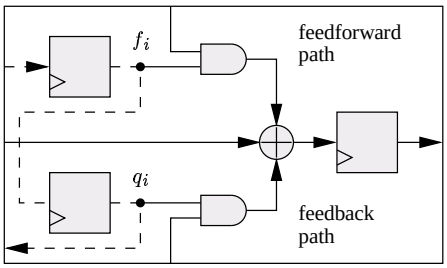


Figure 9: Basic building block

the respective coding function. Applying extensive retiming to the circuit in Figure 4 could provide the same property; however, it proportionally increases the number of registers based on the used cutset stages. Consequently, the observer canonical architecture uses a minimal number of registers, and for encoder memory of $m > 2$ it is the only form that fully exploits flexibility in terms of speed since the minimum clock period applies for the whole range of possible polynomials. Hence, the basic encoder is realized according to Figure 7.

## 3. SYSTEM SPECIFICATION

In order to cover a wide range of convolutional codes there should be a certain number of polynomials generated simultaneously. The memory of a single encoder will be 10, and we have chosen up to 16 data streams that can be emitted in parallel or, by proper configuration, can be joined to form code rates of $1/c, c = 2 \ldots 16$. Besides, using more than one input stream broadens the range to $b/c, b = 1 \ldots 15, b < c$. Another question is whether to include recursive convolutional codes or not. In [3] it is shown that these codes give better performance at rate $1/2$ and low $E_b/N_0$ compared to non-recursive ones, so it makes sense to realize these functions as well. The use of recursive codes can also be motivated by the fact that they are a basic part of turbo codes [4], that will be realized on this platform in the future by adding an interleaver. Since they use less memory, usually $m = 2 \ldots 4$, it becomes obvious that the whole model has to be fully parameterizable.

## 4. IMPLEMENTATION

The choice of a proper implementation platform for this initial project state is motivated by different reasons. If pursuing low power issues it is advisable to develop dedicated hardware. This is the approach with highest efficiency per computation because power saving techniques can be applied on all levels of the design [5]. On the other hand, the design cycle for an FPGA is usually much shorter than for an ASIC, where chip fabrication time has to be taken into account. Thus, an FPGA is better suited for prototyping because functional behavior can be verified much faster. Since this aspect is more important in this initial stage, the system is realized on a XILINX Spartan-II FPGA [6]. However, modules are still developed with regard to reusability in a future ASIC, suitable for low power solutions. Figure 8 depicts the flexible encoder architecture that basically consists of 16 parallel encoders described by Figure 7.

The dashed line connecting the encoders acts as a single bit line used for configuration purposes. Flexibility is incorporated by having total control over the coding polynomials, which requires switches representing these coefficients. A switch uses a register and an AND-gate. Since there are 16 encoders, each having $2m + 1 = 21$ coefficients, there will be 336 switches in total. Thinking about how to configure these switches, it becomes clear that a serial approach has to be pursued since it is virtually impossible to efficiently route such a number of registers in this FPGA to the required gates. By using a shift register chain to set up these coefficients the routing effort is put on a local level. On this level one can gain a lot by using a so called *Configurable Logic Block* (CLB) [6]. A CLB in a Spartan-II device consists of two identical slices, with each slice having two configurable registers, two look-up tables realizing combinational functions, and dedicated carry and control logic.

The gray box in Figure 7 enclosing a register, two switches and an XOR-gate is the basic building block of the design and is shown in detail in Figure 9. The coefficients of a polynomial are represented by the output of the registers on the left side. A multiplication in $\mathbb{F}_2$ is simply performed by an AND-gate incorporating the respective switch. Again, the dashed line in this block represents the smallest part of the shift register chain. In configuration mode all the coefficient bits ripple through this chain. Since 10 serially connected blocks form a basic encoder, 21 shifts are needed

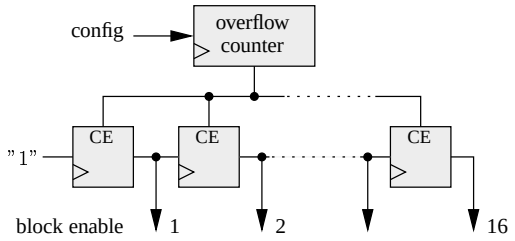


Figure 10: Block enable logic

to completely set up one coding polynomial. Then, the bits are shifted to the next block and so on until the desired number of polynomials is established. Notice that on every hierarchy level there is always just one shift input and one shift output, simplifying the whole routing process and increasing modularity.

Clearly, the configuration process can be utilized at the same time to support power saving since only the programmed blocks should be running. By keeping track of the number of shifts, every block can be enabled separately as shown in Figure 10. An overflow event should occur when a single block is programmed, that is, after 21 clock cycles. The overflow counter then emits a clock enable pulse and the shift register is set to its next state, enabling one block after another. Contrary to an ASIC, it is highly recommended to avoid clock gating in an FPGA [7] since it can introduce glitches, and increase clock delay and clock skew due to unprofitable routing. However, since a CLB already has a separate clock enable input it can be used to explicitly prevent its flip-flops from changing states and thus consuming switching power.

## 5. RESULTS

After thoroughly looking at architectural issues, there is still the question of how these approaches are mapped into the FPGA. This is important since it basically determines the performance of the system. The whole design is based upon the block in Figure 9 and improvement on this level has therefore a positive effect on the system level. Consequently, a bottom-up design style was applied. According to the reports from the synthesis and routing tools, this logic is mapped into one CLB. As mentioned, the three-input XOR-operation will be executed in two steps. First, the result from the two coefficient paths is evaluated in a look-up table with four inputs, namely the two actual inputs from the feedback and the feedforward path and the two respective switch settings that are stored in registers. The 1-bit output of this table is then merged with the input from the previous stage in a basic XOR-cell which is inherent in a CLB. Finally, this result is saved in a register. The hardware utilization of this building block is therefore three registers, one look-up table and a basic XOR-cell. Clearly, this approach efficiently uses the given resources. The maximum estimated clock frequency for this building block is 157 MHz. However, introducing input and output pads that add delay to the design decreases the speed to 110 MHz.

Routing reports of the complete system show that 422 slices out of 1200 are used which is equivalent to a CLB usage of 35%. Furthermore, 517 flip-flops and 235 4-input look-up tables are utilized in those CLBs which corresponds to 21% and 9% usage, respectively. These numbers match well with the preceding considerations where it was shown that a basic building block uses three out of four registers in a CLB and one out of four look-up tables.

The speed requirements are supported by applying suitable synthesis constraints, for example, using an attribute called LOC which advises the routing tool to place logic in defined areas. This can increase the design density which at the same time positively affects the wiring delay. The correctness of the system was verified on all stages of the design process, from functional simulation of the component descriptions to simulation after routing and final testing on the actual FPGA-board. The flexible encoder can be safely clocked with up to 50 MHz. However, post-synthesis reports and simulations even verified functional correctness at clock frequencies up to 95 MHz, which could not be tested on the board due to limitations of the clock source.

## 6. CONCLUSIONS

This paper presented a flexible convolutional encoder that fully exploits the range of coding polynomials for given memory. A thorough investigation of critical path properties for different architectures is the basis for this observation. Power saving issues are already supported with regard to implementation in a future flexible decoder architecture. Since the system performance relies mainly on the basic building block shown in Figure 9, optimization effort on this level has a positive effect for the whole design. It is shown that the presented approach efficiently uses the resources provided by the FPGA. When developing future dedicated hardware, all modules can be reused and an implementation of this basic block in a custom cell will be considered. In order to broaden the application range of the platform, future work will address a flexible interleaver that has to be added to the design to generate turbo codes.

## 7. ACKNOWLEDGMENTS

This project is supported by the Swedish Socware program, the EU Pacwoman project, and the Competence Center for Circuit Design (CCCD) at Lund University.

## 8. REFERENCES

[1] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. New York: IEEE Press, 1999.

[2] K. K. Parhi, *VLSI Digital Signal Processing Systems*. New York: Wiley, 1999.

[3] J. B. Anderson, "Best short rate 1/2 tailbiting codes for the bit-error rate criterion," *IEEE Transactions on Communications*, vol. 48, pp. 597–610, April 2000.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE International Conference on Communications*, vol. 2, pp. 1064–1070, May 1993.

[5] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proceedings of the IEEE*, vol. 83, pp. 498–523, April 1995.

[6] XILINX, http://www.xilinx.com, *Spartan-II FPGA Family: Functional Description*, 2001.

[7] XILINX, http://www.xilinx.com, *Synthesis and Simulation Design Guide*, 2000.