Simplified Hardware Bit Correlator

Christophe Cunat, Emmanuel Boutillon*†

January 24, 2007

Abstract

This letter proposes a suboptimal implementation of a binary correlator suitable for detecting a known fixed pattern in a binary stream. The theoretical performances in terms of the probability of nondetection and the probability of false alarm are evaluated. These performances show that the degradations are negligible. Compared to a proprietary core provided by FPGA vendor, this implementation allows a 15 % look-up table reduction, a 30 % register reduction and up to a 30 % higher clock frequency in a FPGA.

1 Introduction

Binary correlator is a common tool in digital signal processing systems used for synchronisation in a binary stream. The principle is based on the detection of a known fixed pattern in a binary stream which can be corrupted by a noise with a probability of error p. If the degree of match between the N last received bits and the N bits of the training sequence $\text{seq}(k)_{k\in\{0,\ldots,N-1\}}$ is above a given threshold, a new frame is detected. This principle is widely

*Manuscript received January 24, 2007.

†Dr Cunat used to be with the LESTER Laboratory at the Université de Bretagne Sud and is now with TurboConcept. Pr Boutillon is with the LESTER Laboratory.

used in the context of symbol timing synchronisation for OFDM transmission [1, 2, 3].

In this letter, we propose a suboptimal algorithm which simplifies the bit-correlator hardware implementation in a FPGA. The probability of false alarm (P_{fa}) and the probability of nondetection (P_{nd}) for both optimal and the proposed bit correlator are given.

2 Bit correlation

The exact match score $c^e(n)$ for the n^{th} bit between the training sequence $\text{seq}(k)_{k\in\{0,\ldots,N-1\}}$ and the last N received bits of the sequence $x(k)_{k\in\mathbb{N}}$ is defined in eq. (1) as the number of common bits.

$$c^e(n) = \sum_{k=0}^{N-1} \overline{\text{seq}(k) \oplus x(n-k)} \tag{1}$$

Let's denote $b_k(n) = \overline{\text{seq}(k) \oplus x(n-k)}$ in the following discussion. If the bit rate is equal to the clock rate (i.e., a received bit at each clock cycle), the computation of $c^e(n)$ requires typically a shift register of size N to store the sequence $x(k)_{k\in\{n-N+1,\ldots,n\}}$ [2]. Then, the $b_k(n)$ values can be directly computed from eq. (2).

$$b_k(n) = \begin{cases} x(n-k) & \text{when } \text{seq}(k) = 1, \\ \overline{x(n-k)} & \text{when } \text{seq}(k) = 0. \end{cases} \tag{2}$$

Finally, a pipelined addition tree performs the summation of the N partial results. The summation result is then $c^e(n)$.

For the sake of simplicity, the sequence length N is assumed to be a multiple of 4. In order to implement efficiently the bit correlation function

described above onto a FPGA, the internal logic structure of the FPGA is taken into account. This structure consists of a set of 4-input/1-output Look-Up Tables (LUT) [4, 5]. This structure is found within FPGAs provided by the two companies representing about 90 % of the FPGA market [6]. To fully exploit the 4 inputs of a LUT, eq. (1) can be rewritten as in eq. (3)

$$c^e(n) = \sum_{k=0}^{\frac{N}{4}-1} f_k\Big(x(n-4k), x(n-4k-1), \\ x(n-4k-2), x(n-4k-3)\Big), \tag{3}$$

with $f_k(i_0, i_1, i_2, i_3)$ defined in eq. (4).

$$f_k(i_0, i_1, i_2, i_3) = \sum_{l=0}^{3} \overline{\mathrm{seq}(4k+l) \oplus i_l}. \tag{4}$$

It can be noticed that each f_k is the sum of 4 binary values. Thus, they take their value between 0 and 4; 3 bits (f_k^2, f_k^1, f_k^0) are required to code a value f_k: $f_k = 4f_k^2 + 2f_k^1 + f_k^0$. A direct implementation of f_k can be done using 3 LUTs to generate the bit values f_k^2, f_k^1 and f_k^0 directly from (i_0, i_1, i_2, i_3).

Let p_e be the channel bit error probability and p_b the probability that bit b_k equals 0. When the received sequence $x(k)$ is synchronised, the probability that bit b_k equals 0 is $p_b = p_e$. Otherwise, p_b equals 0.5. In fact, when not synchronised, at least P ($P \in [1..N]$) bits of the last N received bits are random data not correlated with the training sequence. If $P < N$, $N - P$ bits are non synchronized bits from the training sequence and the partial match score of these $N - P$ bits is in average around $\frac{N-P}{2}$ (training sequences are generally chosen to have good auto-correlation properties). This partial score is also obtained for $p_e = 0.5$. As p_b and p_e can be seen as

equivalent, the probability of error will be denoted as p in the following.

The probability density function (p.d.f.) for the sum of 4 binary values is $P(f_k = a) = \binom{4}{a} p^{4-a}(1-p)^a$ i.e. $\{p^4, 4p^3(1-p), 6p^2(1-p)^2, 4p(1-p)^3, (1-p)^4\}$. In the system functional domain, when synchronised, p is significantly lower than 0.5, otherwise p is around 0.5. Thus, when synchronised, $P(f_k = 1) = 4p^3(1-p)$ can be considered as negligible compared to $P(f_k > 1)$ and it is not useful to discriminate the cases where f_k is equal to 0 or 1. The function f'_k expressed in eq. (5) is chosen for implementation.

$$f'_k(i_0, i_1, i_2, i_3) = \max\left(0, f_k(i_0, i_1, i_2, i_3) - 1\right). \tag{5}$$

The new p.d.f. of f'_k is $P(f'_k = 0) = P(f_k = 0) + P(f_k = 1)$ and $P(f'_k = a) = P(f_k = a + 1)$ for $a = 1, 2, 3$. This function takes its value between 0 and 3: 2 bits f'^1_k and f'^0_k are needed to code the result. This function can be implemented using only 2 LUTs. This represents a significant lower complexity than the optimal solution. Note that the proposed approximated match score $c^a(n)$ takes its value between 0 and $\frac{3}{4}N$.

3 Theoretical performance for the proposed implementation

The performance can be evaluated theoretically according to the bit error probability p. In the case of the exact computation, the p.d.f. of c^e is given by $P(c^e_p = s) = \binom{N}{s} p^{N-s}(1-p)^s$. In the case of the approximate function, the exact expression of the p.d.f. is complex to derive. Nevertheless, it is possible, for a given error probability p, to determine numerically its p.d.f. In fact, c^a is the summation of $N/4$ random variable (r.v.). Thus its p.d.f. function is the convolution of the p.d.f of those r.v. In practice, the p.d.f.

of c^e is also obtained with the same method. Fig. 1 shows the p.d.f. for unsynchronised and synchronised input (with $p = 0.3$ and $N = 64$) for the exact and the approximated schemes.

A detection threshold $s_t(p)$, an integer between 0 and N or $\frac{3}{4}N$, defines whether or not a detection occurs. From such a threshold, probability of false alarm P_{fa} and probability of nondetection P_{nd} are defined in eq. (6) and eq. (7), respectively. The receiver operating characteristic (R.O.C.) can be derived from these expressions: Fig. 2 presents the R.O.C. drawn (with $p = 0.35$) for the two schemes. The two R.O.C. do not show any significant difference. The two curves are modified in the same manner for any variation of p.

$$P_{\text{fa}}(p) = P(c_{0.5}(n) > s_t(p)) = \sum_{k=s_t(p)+1}^{N} P(c_{0.5}(n) = k) \tag{6}$$

$$P_{\text{nd}}(p) = P(c_p(n) \leq s_t(p)) = \sum_{k=0}^{s_t(p)} P(c_p(n) = k) \tag{7}$$

An optimal threshold is defined as expressed in (8). It varies with p and is different for each computation scheme.

$$s_t^o(p) = \max_{s_t} \left(P(c_p(n) = s_t) \leq P(c_{0.5}(n) = s_t)\right) \tag{8}$$

Fig. 3 collects the probability P_{d} of correct detection ($P_{\text{d}} = 1 - P_{\text{nd}}$) and false alarm using $s_t^o(p)$ as a function of p. The sequence length is $N = 64$. Once again, Fig. 3 does not show significant differences between the two computational schemes.

4 Implementation results

Both the exact and approximated correlations have been implemented with a FPGA Xilinx Virtex2-2000 [4]. For realistic implementation, the pseudo-random sequence is a Chu sequence [7] using a single precision bit for quantification. This well-known sequence is a CAZAC (Constant Amplitude Zero AutoCorrelation) sequence like the ones used in [8]. Typically, over a multiple path channel [9], with a detection threshold of SNR = 10 dB, p_b is around 0.3 when synchronised and 0.5 when not synchronised. Over an additive white gaussian noise channel with the same detection threshold, p_b is around 0.05 when synchronisation is achieved and 0.5 when not.

The exact correlation core of Xilinx (Bit Correlator from the CORE Generator tool [10]) is used as a state of the art design (denoted as Ref.). A generic VHDL component implementing both exact and approximated correlator has been written: the final addition can be chosen as an optimized Carry-Save Adder (CSA) or a classical tree adder [11]. Table 1 presents the place&route results (in term of logic (LUT), registers (DFF) and clock frequency) for the exact (reference, component with and without CSA) and approximated correlators (component with and without CSA). The targeted clock frequency is 280 MHz with no other specific timing contraint given. The implementation of the approximated scheme shows about 15 % (resp. 30 %) of lower complexity in term of LUT (resp. DFF) compared to the reference design, associated with a larger clock frequency of about 30 %.

5 Conclusion

We have proposed a suboptimal binary correlator algorithm which simplifies FPGA implementation. The decrease of complexity (compared to a state

of the art design) is around 15 % in LUT count, 30 % in register count and a better clock frequency without any noticeable performance degradations. Note that this decrease in complexity also involves a reduction of power consumption. Moreover, this method can easily be generalized. For example, if 5-input LUTs are available, the cases when f_k equals 0, 1 and 2 can also be agregated. Finally, this method for aggregating least significant bits can also be efficiently applied by any ASIC designers.

Acknowledgment

This work has been supported by the Région Bretagne (PRIR n° 06003125 RITMO : *Recherche et Intégration d'algorithme dans le domaine des Techniques multi antennes (MIMO)*). Matlab code and VHDL code are freely available from the authors (http://web.univ-ubs.fr/lester/~boutillon/correlator/correlator.html).

References

[1] A. Fort, J.-W. Weijers, V. Derudder, W. Eberle, and A. Bourdoux, "A Performance and Complexity Comparison of Auto-correlation and Cross-correlation for OFDM Burst Synchronisation," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 2, Apr. 2003, pp. 341–344.

[2] S. Kulkarni, P. Mazumder, and G. I. Haddad, "A High-speed 32-bit Parallel Correlator For Spread Spectrum Communication," in *VLSI Design, 1996. Proceedings, Ninth International Conference on*, Jan. 1996, pp. 313–315.

[3] K. Wang, J. Singh, and M. Faulkner, "FPGA Implementation of an OFDM-WLAN Synchronizer," in *Electronic Design, Test and Applications, 2004. DELTA 2004. Second IEEE International Workshop on*, Jan. 2004, pp. 89–94.

[4] *Virtex-II Platform FPGAs: Complete Data Sheet*, Xilinx, Inc., Mar. 2005, [online:] http://www.xilinx.com.

[5] *Stratix II Device Handbook, Volume 1*, Altera, Dec. 2005, [online:] http://www.altera.com.

[6] "Corporate Overview," Lattice Semiconductor Corporation, 2006. [Online]. Available: http://www.latticesemi.com/

[7] D. C. Chu, "Polyphase Codes With Good Periodic Correlation Properties," *IEEE Trans. Inform. Theory*, vol. 18, no. 4, pp. 531–532, July 1972.

[8] I. Computer Society and I. Microwave Theory Techniques Society, Eds., *802.16 IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems*. IEEE, 2004.

[9] B. Porat and B. Friedlander, "Blind Equalization of Digital Communications using High-Order Moments," *IEEE Trans. Signal Processing*, vol. 39, no. 2, pp. 522–526, Feb. 1991.

[10] *CORE Generator Help*, Xilinx, Inc., 2005, [online:] http://www.xilinx.com/support/sw_manuals/xilinx7/.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*. The MIT Press, 1990.

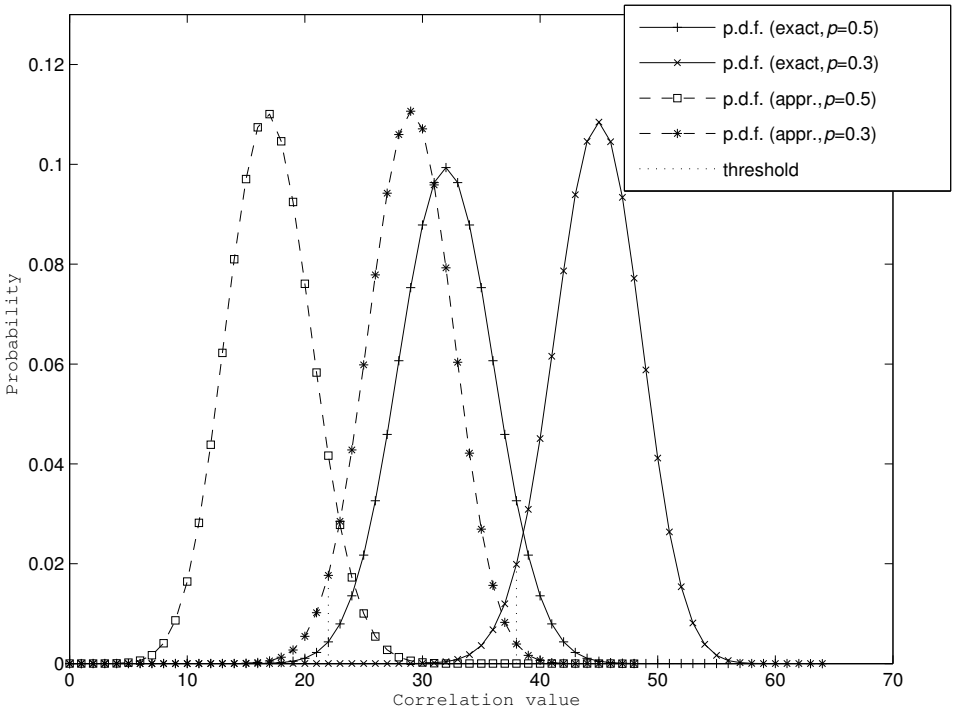

Figure 1: Probability density functions for both exact and approximated computation schemes for $N = 64$. P.d.f are represented for a nonsynchronised input ($p = 0.5$) and a synchronised input with error probability $p = 0.3$. The thresholds defined in (8) are represented for the two schemes.

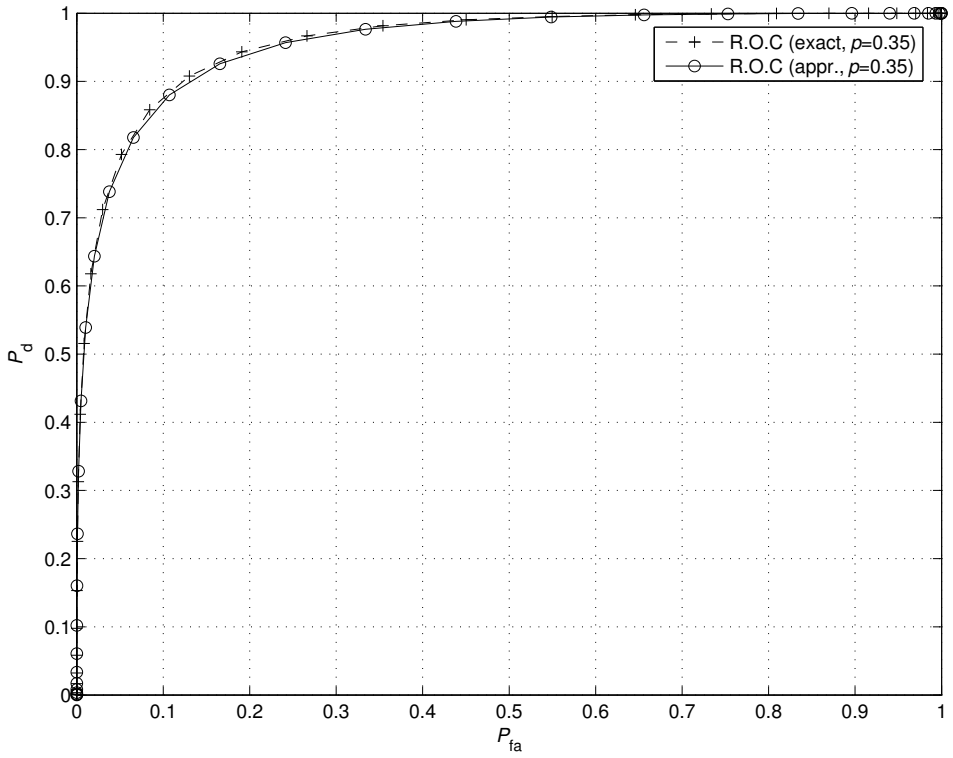

Figure 2: Receiver Operating Characterics for the two computational schemes ($N = 64$, $p = 0.35$): probability of detection $P_{\mathrm{d}} = 1 - P_{\mathrm{nd}}$ as a function of probability of false alarm.

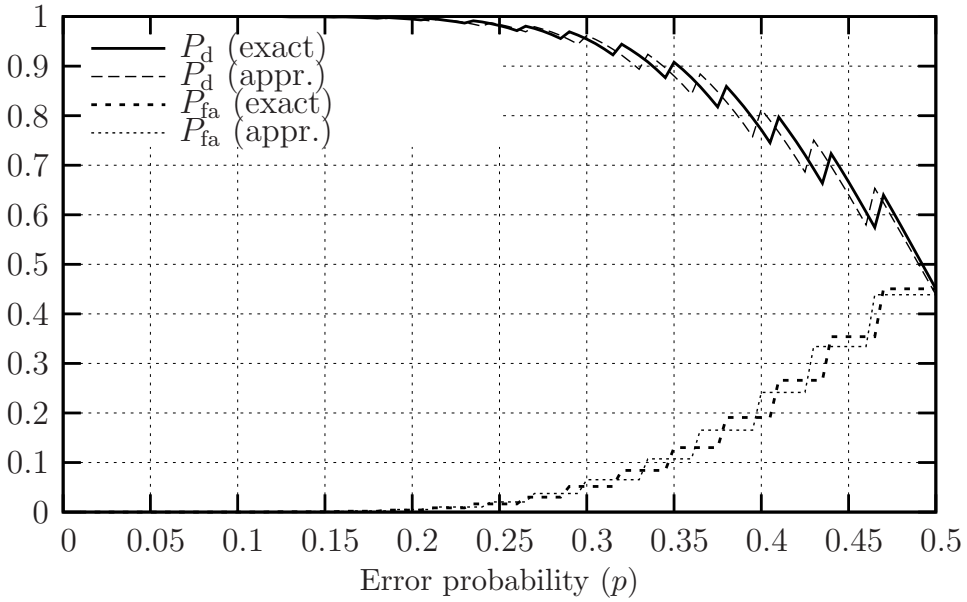

Figure 3: Probability of synchronisation as a function of the probability of error on the input sequence ($N = 64$). Probabilities of correct detection (P_d) decrease as p increases and probabilities of false alarm (P_{fa}) increase with p.

Table 1: Implementation results on a Xilinx Virtex2-2000 for logic (LUT), registers (DFF) and clock frequency after place&route (MHz)

		Exact			appr.	
		Ref.	Add.	CSA	Add.	CSA
$N = 256$	LUT	439	659	508	376	377
	DFF	810	1003	794	493	540
	MHz	219	214	272	231	281
$N = 128$	LUT	216	325	252	185	185
	DFF	396	493	397	239	271
	MHz	219	210	281	253	281
$N = 64$	LUT	105	159	124	90	89
	DFF	190	239	210	113	133
	MHz	280	265	281	285	285
$N = 32$	LUT	50	76	60	43	42
	DFF	88	113	103	51	63
	MHz	286	280	283	283	287