Hardware/Software Exploration for an Anti-collision Radar System

Sébastien Le Beux, Vincent Gagné, El Mostpaha Aboulhamid, Philippe Marquet, Jean-Luc Dekeyser

Hardware/Software Exploration for an Anti-collision Radar System

S. Le Beux[1], V. Gagné[2], E.M. Aboulhamid[2], Ph. Marquet[1], J.-L. Dekeyser[1]

[1] Laboratoire d'informatique fondamentale de Lille
Université des sciences et technologies de Lille
{lebeux, marquet, dekeyser}@lifl.fr

[2] Département d'informatique et de recherche opérationnelle, Université de Montréal
{gagnevi, aboulham}@iro.umontreal.ca

Abstract

Anti-collision radars help prevent car accidents by detecting obstacles in front of vehicles equipped with such systems. This task traditionally relies on a correlator, which searches for similarities between an emitted and a received wave. Other modules can then use the information produced by the correlator to compute the distance and the relative speed between the moving vehicle and the obstacle. We implemented such a system using FPGAs. We used hardware blocks to implement the high demanding computing correlator and a soft-core processor to compute the distances and speeds. In order to improve the maximum detection distance reached by the correlation algorithm, we developed and tested a modified version of a Higher Order Statistics based algorithm. This work results in a detailed description of this new algorithm, its possible implementations and the corresponding FPGA synthesis results.

1. Introduction

Safety has become an important issue for car manufacturers. Some approaches target the safety of the passengers in a crash, while others try to anticipate them. Anticipating a delicate situation requires the knowledge of the surrounding context. It is the case of anti-collision radars, which use radar waves to detect obstacles. An embedded anti-collision radar is a system which emits a wave. When the emitted wave hits an obstacle (other vehicles, animals, etc.), it is re-emitted in the direction of the vehicle. The system compares the received wave with the emitted one, searching for similarities resulting from the presence of an obstacle in front of the car. Computing the time spent between the emission and the reception of the wave, the system can determine the distance between the car and the obstacle. By periodically computing this distance, the system can also estimate the relative speed of the car and the obstacle. Similarly, the system can determine the acceleration using two or more speed estimations.

Thus, detecting an obstacle comes down to one task: performing the correlation of an emitted and a received wave. A well-known correlation algorithm, described in Section 3, fails to satisfy certain constraints [9]: according to the characteristics of a certain radar, the maximum detection distance of such an algorithm is almost 100m in favorable conditions [5]. In this article, we propose a new algorithm, based on Tugnait's work [3,4], that increases the maximum detection distance. This algorithm is based on Higher Order Statistics formulation [3].

This obstacle detection is placed in an embedded system context, where resources are limited. This leads us to use FPGA technologies, which offer flexible solutions for the implementation of the modified Tugnait algorithm. Moreover, FPGAs allow us to implement a complete system, where intensive signal processing tasks are implemented as hardware blocks and sequential computations (distance, speed, etc.) are performed on a soft-core processor (see Figure 1).

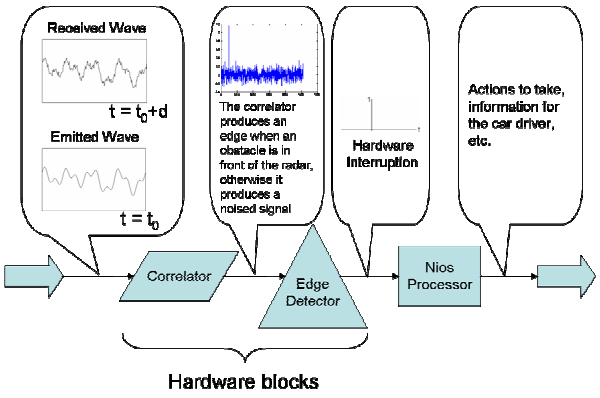

Figure 1: Schematic view of the obstacle detection system on FPGA

The paper is organized as follows: Section 2 justifies the need for a new anti-collision radar algorithm. Section 3 presents a well-known correlation algorithm and its FPGA implementations. From that, we extract simple mathematical formulas estimating the amount of resources needed for their implementation on a FPGA. The same formulas are used to estimate the number of resources taken by the modified Tugnait algorithm, presented in Section 4. Finally, Section 5 concludes this work.

2. Motivation and justification

A well-known detection algorithm, presented in Section 3, allows the detection of an obstacle located at 100m or less in front of the radar. Increasing the algorithm's signal-to-noise ratio

(SNR) is equivalent to increasing its maximum detection distance. This in turn helps to better anticipate a collision. Several solutions exist to enhance the SNR: increase the power of the emitted wave, increase the sample resolution of the received wave, increase the length of the reference code and changing the detection algorithm. We decided to implement a modified version of the third Higher Order Statistic algorithm [3,4]. We aim to increase the SNR, and in doing so increase the maximum detection distance. Other algorithms [6,7,8] that propose to reduce the noise within the emitted signal may be considered in the future.

3. Correlation algorithm

Detecting an obstacle is possible thanks to the comparison between the received wave and the emitted one. A common correlation algorithm [2] performs this comparison. In this section, we give three views of this algorithm: mathematical, software and hardware.

3.1 Mathematical formulation

A mathematical formulation of the correlation algorithm is given by:

$$C_{cy}(j) = \frac{1}{N}\sum_{i=0}^{N-1} c(i)\cdot y(i+j) \quad (1)$$

N: The number of samples in the reference code. In our case, we fix *N* to 1023 based on previous work [5].

c: The reference code [1] used to create the emitted wave. Each sample of *c* is encoded using 1 bit ('1' for +1, '0' for -1).

y: The received wave. Each sample of *y* is encoded using 4 bits (previous studies have shown that it is a good trade-off between precision and resources used) and it is used *N* times [5]. A reduced formula is:

$$C_{cy}(j) = \sum_{i=0}^{1022} c(i)\cdot y(i+j) \quad (2)$$

3.2 Software implementation

A software implementation of the correlation algorithm given in formula (2) is immediate. Figure 2 contains such an implementation in C.

```
while (1) {
  OutCorrelation = 0;
  y[0] = *ReceivedSignal;
 for(i=1022; i=>0; i--) {
    y[i+1] = y[i];
  }
  for(i=0; i<1023; i++) {
    OutCorrelation += c[i] * y[i];
  }
  EdgeDetection(OutCorrelation);
}
```

Figure 2: A C implementation of the correlation algorithm

When using an AMD processor (32 bits, 1200 MHz, 256 kB cache size, 1 GB RAM memory), the maximum runtime frequency is approximately 63 kHz. Using another AMD processor (64 bits, 2200 MHz, 512 kB cache size, 2 GB RAM memory), we obtain a maximum runtime frequency of 122 kHz. Those results are far away from the required frequency, which is 100 MHz.

In a software implementation, operations are mainly done in a sequential way. However, the code given in Figure 2 is highly parallelizable, suggesting that a hardware implementation would give better results.

3.3 Hardware implementation

When implementing the correlation algorithm (2) on a FPGA, one needs to understand how its design choices will affect the performance of the resulting circuit. The objective of this section is to clearly define which kind of implementation is efficient and which one is not. Therefore, we synthesized each one of the proposed implementations.

Moreover, in order to re-use the synthesis work done in this section, mathematical formulas estimating the number of resources used by any given implementation were developed. The formulas shall allow us, in Section 4 and in future work, to easily estimate the impact of future hardware implementations on FPGA.

The formula (2) can be decomposed into three steps: shift registers, multiplication and addition.

Shift registers: In the correlation algorithm (2), each received wave sample is used 1023 times. Since we need to compare the last 1023 received samples with the reference (which is fixed), each newly received sample is sent to a shift register, as shown in Figure 3.

$$E_{Shift\ Registers} = N \cdot n \quad (3)$$

E represents the estimated number of resources used and *n* represents the number of bits used to encode each received sample. In our case, $n = 4$ and $N = 1023$.

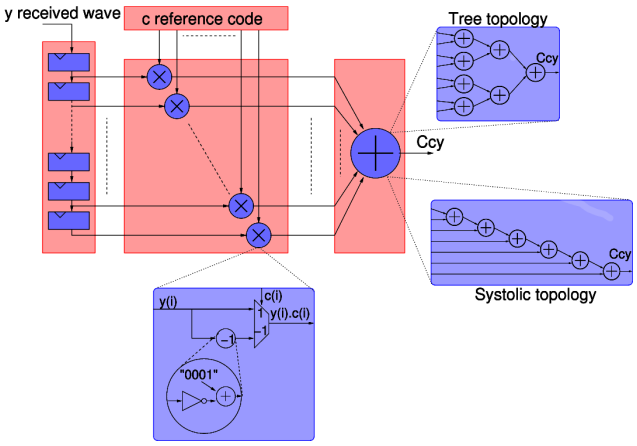

Figure 3: Schematic view of the hardware implementation of the correlation algorithm

Multiplication: 1023 multipliers are required to perform the global summation.

$$E_{Simple\ Multiplication} = N \cdot n \quad (4)$$

Addition: There are two main hardware implementations for a summation: a tree topology or a systolic topology.

3.3.1 Tree topology: A well-known solution to implement a

summation is the tree topology. To allow better segmentation of the addition tree (which has 1023 inputs), a neutral value is added. Thus, the addition tree has 1024 values and can easily be decomposed (see Figure 3).

Formula (5) estimates the number of FPGA resources required for the implementation of the tree.

$$E_{Addition\,Tree} = N \cdot (n+2) - \lg(N) - n - 2 \quad (5)$$

In order to use fewer resources, it is possible to perform averages instead of additions. Of course, an intermediate solution (using both additions and averages) can be used.

Formula (6) estimates the number of resources used when performing averages.

$$E_{Average\,Tree} = (N-1) \cdot (n+1) \quad (6)$$

3.3.2 Systolic topology: The systolic topology is another well-known structure. The estimation formulas can be expressed by:

$$E_{Addition\ Systolic} = N \cdot (\lg(N) + n - 1) - n + 1 \quad (7)$$

$$E_{Addition\ Systolic\ Pipeline} = 2 \cdot E_{Addition\ Systolic} \quad (8)$$

Similarly to the tree topology, the systolic topology can perform averages instead of additions. The formulas become:

$$E_{Average\,Systolic} = (N-1) \cdot (n+1) \quad (9)$$

$$E_{Average\,Systolic\,Pipeline} = (N-1) \cdot (2 \cdot n + 1) \quad (10)$$

Table 1: Synthesis and estimation results for the correlation algorithm on a Stratix pro board

Implementations details			Max. frequencies (in MHz)	Used Resources	
				Synthesis	Estimation
Addition	Tree	Full pipeline	229	14 362	14 312
		One cycle	45	11 873	14 312
	Systolic	Full pipeline	151	36 736	34 802
		One cycle	5,5	12 957	21 493
Average	Tree	Full pipeline	232	13 296	13 299
		One cycle	49	10 718	13 299
	Systolic	Full pipeline	180	15 338	17 391
		One cycle	0.61	11 761	13 299

3.4 Comparison and summary

Using the formulas (1) to (10) and synthesis results, we created Table 1, summarizing the synthesis and estimation results of different correlation algorithm implementations. The estimations errors do not exceed 20%, except for the addition, systolic one cycle solution. Indeed, this solution can be highly optimized by the FPGA manufacturers' tools. Examining Table 1, we find that the tree topology is the most efficient implementation in both addition and average cases. The tree topology shall be used in more complex algorithm implementations.

The synthesis results shown in Table 1 were obtained using an Altera Stratix pro.

4. Modified Tugnait algorithm

In this section, we present and implement a new algorithm, based on Tugnait's research [3, 4]. The objective of this new algorithm is double: increasing both the maximum obstacle detection distance and the SNR.

4.1 Mathematical formulation

The mathematical formulation of the Higher Order Statistics (HOS) [3] algorithm is given by:

$$J_{3,2}(io) = \sum_{j=0}^{N-1} \left[\frac{1}{N} \sum_{i=0}^{N-1} y(i) \cdot c(i+1) \cdot c(i+j) \right] \cdot \left[\frac{1}{N} \sum_{i=0}^{N-1} y(i) \cdot c(i+1) \cdot y(i+j+io) \right] \quad (11)$$

Similarly to the correlation, we are not interested by the normalization. Thus, the implemented algorithm is:

$$J_{3,2}(io) = \sum_{j=0}^{N-1} C_{ycc}(j) \cdot C_{ycy}(j+io) \quad (12)$$

$$C_{ycc} = \sum_{i=0}^{N-1} y(i) \cdot c(i+1) \cdot c(i+j) \quad (13)$$

$$C_{ycy} = \sum_{i=0}^{N-1} y(i) \cdot c(i+1) \cdot y(i+j) \quad (14)$$

y: Received wave
c: Reference code
N: Length of reference, = 1023

$J_{3,2}(io)$ is a correlation. Since the two entry points of this correlation are themselves results of previous correlations, we can expect the synthesis to produce a much larger design.

Three elements are required to compute *Cycc*: $y(i)$, $c(i+1)$, $c(i+j)$. The last two are coded using 1 bit. The first one, $y(i)$, uses 4 bits.

Cycy structure is similar to the *Cycc* one. However, its implementation is much more complex. The difference is that $y(i+j)$ is needed. Since $y(i+j)$ is a previous value of the input signal (encoded using 4 bits, as opposed to the 1-bit encoding of $c(i+j)$), synthesizing this part of the algorithm requires more resources than synthesizing *Cycc*.

The evaluation of $y(i) \cdot c(i+1)$ is needed to compute both $Cycc(j)$ and $Cycy(j)$. To optimize the implementation, it is evaluated only once and then sent to both modules:

$$s(i) = y(i) \cdot c(i+1) \quad (15)$$

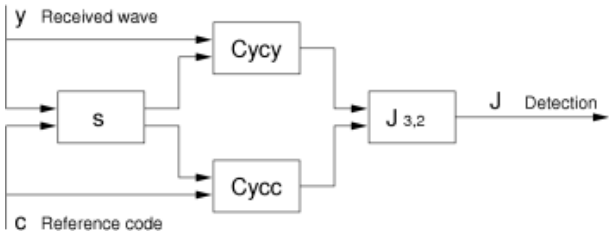

Figure 4: Schematic view of the modified Tugnait algorithm

Figure 4 shows the schematic view of our hardware implementation of formula (11).

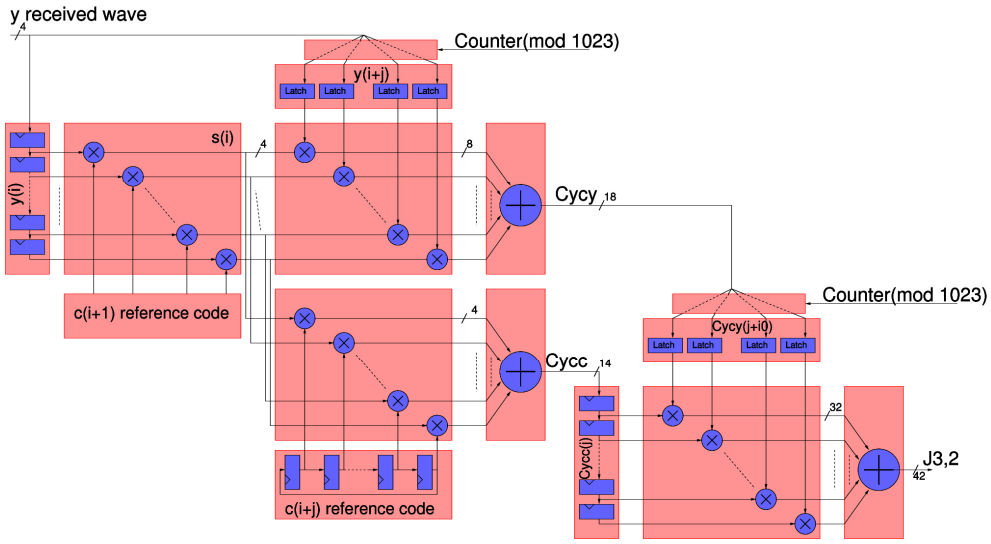

Figure 5: Schematic view of the modified Tugnait algorithm hardware implementation

4.2 Comparison with the correlation algorithm

According to the mathematical formulations of both correlation and HOS algorithms, we can evaluate their differences from a computation point of view. We choose to use an input wave composed of integers ranging from -7 to 7. Moreover, after analyzing the results presented in Table 1, we chose the pipelined tree solution performing additions as the reference correlation algorithm. Results are shown in Table 2.

4.3 Software implementation

We implemented the algorithm described in (11) using the C language:

```
While(1) {
  J32 = 0;
  y[0] = *ReceivedSignal;
  for(i=1022; i=>0; i--) {
    y[i+1] = y[i];
  }
  for(i=0; i<1023; i++) {
    s = y[i] * c[i+1];
    Cycc[io] += s * c[i+io];
    Cycy[io] += s * y[i+io];
  }
  for(j=0; j<1023; j++) {
    J32 += (Cycc[j] * Cycy[(j+io) % 1023]);
  }
  io = (io+1) % 1023;
}
```

The resulting frequencies are 13 kHz for the 1200 MHz AMD processor and 32 kHz for the 2200 MHz 64-bit AMD processor.

4.4 Hardware implementation

4.4.1 Full version: The hardware implementation of the HOS algorithm requires much more resource than the correlation one.

One of the main reasons is the complexity increase in the multiplications. The HOS algorithm requires 1023 multiplications of two 4-bit operands and 1023 multiplications of a 15-bit operand by a 19-bit operand. DSP blocks can be used to implement some of the multiplications.

Table 2: Comparison between the correlation and the modified Tugnait algorithms

Algorithm	Formula	Estimated number of logical cells	
Correlation	(2)	14 312	
HOS	(15)	8 184	378 480
	(13)	11 243	
	(14)	33 749	
	(12)	325 304	

The computations of *Cycy* and $J_{3,2}$ elements require $y(i+j)$ and $Cycy(j+io)$. Therefore, we cannot use shift registers to store the values of $y(i+j)$ and $Cycy(j+io)$. In order to limit the number of connections, we have to immediately put the received sample in the right place. Looking at the algorithm, we find that they can be loaded inside the array $y(i+j)$ cyclically, using enabled latches and a time base counter. Thus, since one part of the correlator uses shift registers and the other uses the counter solution, an efficient correlation is realized. Figure 5 shows a described view of the hardware implementation of (11).

Simulations allow us to quickly evaluate the system's behavior. Figure 6 compares the two algorithms. In order to simulate the radar's inputs, we have created an input signal identical to the reference code, but attenuated with noise. Reference code and noise are cyclically emitted, as in real conditions. This fact explains the edges on the output sides,

which mean that an object has been detected.

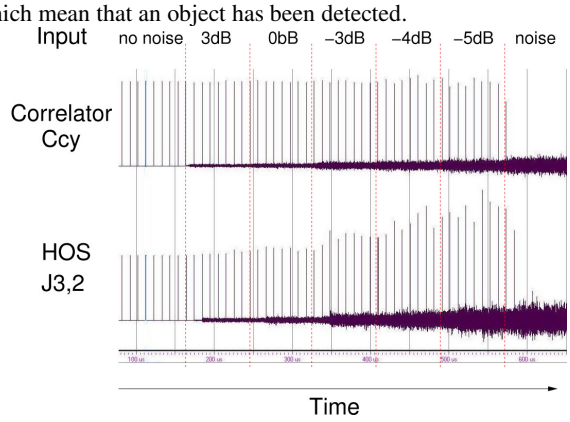

Figure 6: Simulation results comparing both algorithms

As expected, the output noise generated by the modified Tugnait solution is higher than the correlator's one. However, the modified Tugnait solution produces higher edges when used in a high noised environment, showing a performance improvement over the correlator solution.

4.4.2 Reduced version: The full version of the algorithm, with no reduction, requires at least 375 000 logical cells. The Altera board we are targeting is a Stratix pro edition, which contains 40000 logical cells. The full circuit cannot be implemented on it. In that version, the inputs are encoded using 4 bits and the results use 44 bits. However, many reductions are possible, and allow a reduced algorithm implementation on the target FPGA, with a performance reduction.

Two reductions are performed: a) the first uses the average version rather than the addition one; b) the second reduces the sample resolution of the inputs.

Using the formulas developed in Section 3, we can easily estimate the number of resources required for any given implementation. Results are given in Table 3. According to it, the target FPGA (Stratix pro containing 40 000 logical cells) only supports the last solution, which uses the average methodology and receives signals comprised between -1 and 1. Still, various tests run on the implemented reduced FPGA version proved its validity.

In this section, we have presented a modified Tugnait algorithm. From its mathematical formulation, we have made a software implementation, and a hardware one. According to simulation, we have validated the algorithm's behavior and, based on the results given in the correlation algorithm section, we have chosen the right solution and implemented it on the target FPGA.

Table 3: Estimation results for the modified Tugnait algorithm

Implementation choice	Number of logical cells	Possible target
Correlator addition $-7 \geq y \geq 7$	14 312	Stratix II EP2S30
Tugnait addition $-7 \geq y \geq 7$	375 000	none
Tugnait average $-7 \geq y \geq 7$	150 000	Stratix II EP2S180
Tugnait average $-3 \geq y \geq 3$	100 000	Stratix II EP2S130
Tugnait average $-1 \geq y \geq 1$	32 000	Stratix II EP2S60

5. Conclusion

A way to increase the sensibility of an anti-collision detection system is to change the algorithm in charge of the evaluation of the similarities between emitted and received waves. According to simulation, it has been shown that the algorithm based on Higher Order Statistics (HOS) computation is an efficient solution. For a well-known correlator algorithm, we analyzed several FPGA implementations results. From those results, simple mathematical formulas estimating the resources needed by the FPGA were created. According to the estimations, we came up with an efficient hardware implementation of an efficient anti-collision algorithm. However, this algorithm requires too much resource for the target FPGA. A reduced version was then developed. We compared the several implementations of this algorithm, in terms of FPGA needed resources and signal-to-noise ratio. The full system, based on a reduced version, was implemented on FPGA and coupled to a processor. The processor is needed to compute the distance and the speed, and is independent of the chosen algorithm. We are planning on running simulations using real conditions to test HOS algorithm. However, our experimentations allowed us to estimate that a reduced algorithm could be implemented on a FPGA containing 130 000 basics cells. Real test conditions shall allow us to determine the maximum detection distance offered by the implemented algorithm.

6. Acknowledgment

We would like to acknowledge J. Zaidouni and A. Menhaj-Rivencq, from the University of Valenciennes, for the fruitful discussions covering the correlation algorithm. M. Le Beux was supported by ModEasy, an Interreg III A project.

7. References

[1] W. Peterson, E. Weldon, *Error-correcting codes*, MIT Press, 1972.

[2] Byron Edde, *Radar Principles, Technology, Applications* Prentice Hall, 1992.

[3] J.K. Tugnait, "On time delay estimation with unknown spatially correlated gaussian noise using fourth-order cumulants and cross cumulants", *IEEE transaction on signal processing*, vol. 39, n°6, pp. 1258-1267, June 1991.

[4] J.K. Tugnait, "Time delay estimation with unknown spatially correlated gaussian noise", *IEEE transaction on signal processing*, vol. 42, n°2, pp. 549-558, Feb. 1993.

[5] Y. El Hillali "Etude et réalisation d'un système de communication et de localisation, basé sur les techniques d'étalement de spectre aux transports guidés", PhD thesis, University of Valenciennes, 2005.

[6] C. L. Nikias, R. Pan, "Time delay estimation in unknown Gaussian spatially correlated noise", *IEEE trans. on acoustics, speech and signal processing*, vol. 36, n°11, pp. 1706-1714, Nov. 1988.

[7] H. Chiang, C. L. Nikias, "A new method for adaptative time delay estimation for non-gaussian signals", *IEEE trans. on acoustics speech and signal processing*, vol. 38, n°2, pp. 209-219, Feb. 1990.

[8] W. Zhang, M. Raghuveer, "Nonparametric bispectrum-based time delay estimations for multiple sensor data", *IEEE transaction on signal processing*, vol. 39, n°3, pp. 770-774, March 1991.

[9] B. Fremont, A. Menhaj, P. Deloof, M. Heddebaut, "A cooperative collision avoidance and communication system for railway transports", *3rd IEEE Conference on Intelligent Transportation Systems*, 2000.