

FPGA Implementation of Embedded Cruise Control and Anti-Collision Radar

Sébastien LE BEUX

Philippe MARQUET

Ouassila LABBANI

Jean-Luc DEKEYSER

Laboratoire d'informatique fondamentale de Lille
Université des sciences et technologies de Lille, France
Firstname.Lastname@lifl.fr

Abstract

The ModEasy project seeks to develop techniques and software tools to aid in the development of reliable micro-processor based electronic (embedded) systems using advanced development and verification systems. The tools are to be evaluated in practical domains such as the automotive sector for reactive cruise control and anti-collision radar. We choose to define specific IPs using FPGA techniques to cover this application domain. This paper presents the implementation of such a complex and safety application on a single FPGA. The target system is composed of a reactive cruise control, a detection radar and the associated treatments.

1 Introduction and Motivation

The ModEasy project [1] seeks to develop techniques and software tools to aid in the development of reliable microprocessor based electronic (embedded) systems using advanced development and verification systems. The tools are to be evaluated in practical domains such as the automotive sector for reactive cruise control and anti-collision radar but are envisaged to be applicable for generic embedded systems in any safety and mission critical applications in the wider industrial domain. The project seeks to reduce development and costs while maintaining existing high dependability and safety levels as embedded systems become more complex for many existing and new products. To cover this application domain, we foresee two different solutions: a System on Chip (SOC) implementation of different Intellectual Property (IPs), and a full integration on FPGA component.

SoC design covers a lot of different viewpoints including the application modeling by the aggregation of functional components, the assembly of existing physical components, the verification and the simulation of the modeled system, the synthesis of a complete end-product integrated into a single chip. As a rule, a SoC includes programmable pro-

cessors, memory units (data/instruction), interconnection mechanisms and hardware functional units (Digital Signal Processors (DSP) and Application-Specific Integrated Circuits (ASIC)). These components can be generated for a particular application; they can also be obtained from IP providers. For the safety systems we target, such solution seems inappropriate because of the cost and availability of IPs for this system. In ModEasy project, we choose to define specific IPs using FPGA techniques. Metamodel with behavior specifications has to be available from the higher level of modeling. Using model driven engineering we will automatically produce lower level (Register Transfer Level) via successive refinements of the models and compilation techniques.

With this particular approach a single low cost and flexible FPGA has to include IPs for dedicated pieces of the application, general purpose processors to execute elementary tasks and memory to store data. All these parts have to coexist inside the same component and interact to proceed the embedded system.

In this paper, we present the implementation of such complex and safety application on a single FPGA. The target system is composed of a reactive cruise control, a detection radar and the associated treatments.

The paper is organized as follows: section 2 describes the global system. Section 3 describes the automaton which controls and regulates the vehicle speed. Section 4 presents the obstacle detection algorithm we implemented. Section 5 introduces the restitution and the analysis of the environment issued from the radar detection. Moreover, we introduce a way to link both radar and regulation from the environment analysis. Finally, Section 6 concludes this article.

2 Global System Description

Anti-collision radar and the reactive cruise control require information external from the system. The driver has no direct access to the anti-collision radar system, which requires data from the radar (placed at the front of the vehicle). The anti-collision system can decide to slow down the

vehicle speed using brakes. Thanks to an interface placed on the vehicle board, the driver is allowed to switch between the several modes proposed by the cruise control. The cruise control permits the regulation of the vehicle speed via the accelerator pedal. In both cases, the vehicle behavior information (speed, acceleration...) is necessary. The study of a such system can be very important since it makes possible to considerably increase the drivers safety.

Figure 1 gives a simplified UML view on the studied FPGA component which mainly contains four basic elements. The Intelligent Cruise Control with GPS (ICCG), component regulates the car speed; the Detection component detects obstacles in front of the car; the Generator component generates radar waves and specifies the time base for the system; and the Processor component reconstitutes the car external environment provided by the radar, and analyzes this environment for making decisions and sending information to the ICCG system.

2.1 ICCG Component

The ICCG system represents a significant automatic contribution in the automotive field. It allows the control of a car speed depending on the radar information and on its position given by a GPS. The ICCG system can be seen as an electronic device which facilitates the control of a car. It informs the driver about the various changes of speed limits and, in some cases, obliges him to respect them by controlling the accelerator pedals `Car_Throttle`.

2.2 Generator Component

The Generator component generates the emitted wave, send to the radar emitter, according to a particular reference code which is transmitted to the Detection component. The comparison of this reference code with the future received wave will be allow to detect obstacles. The wave is periodically generated according to a time base (`counter` in the figure 1) which is recovered by Processor. The Generator component is not fully described in this paper, it is based on works done in [7].

2.3 Detection Component

An embedded detection radar is a system which emits a wave (according to the Generator stage in our case). When the emitted wave hits an obstacle (other vehicles, animals...), 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.

Thus, detecting an obstacle comes down to one task: performing the correlation of an emitted and a received wave.

A well-known correlation algorithm fails to satisfy the following constraint [5]: according to the characteristics of the radar, the maximum detection distance of such an algorithm is almost 100 m in favorable conditions [7]. In this paper, we propose a new algorithm, based on Tugnait's work [13, 14], which increases the maximum detection distance. This algorithm is based on Higher Order Statistics formulation [13].

2.4 Processor Component

By 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. The periodically computation of this distance allows the system to estimate the relative speed of the car and the obstacle. Similarly, the system can determinate the acceleration using two or more speed estimations. All these tasks permit the environment reconstitution from the radar point of view.

Using this reconstitution, it is feasible to prevent collision and to bypass it by slowing down (braking) the car. Moreover, a tracking algorithm may compute speed of the others car and permits to safely follow the previous car, in a convoy mode. These tasks remain in the `EnvironmentAnalysis` sub-stage of the figure 1.

2.5 FPGA Implementation

The obstacle detection and the ICCG are 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 different stages composing the whole of the system. Moreover, FPGAs allow us to efficiently implement a complete system, where intensive signal processing tasks are implemented as hardware blocks and sequential computations (distance, speed...) are performed on a soft-core processor (see Figure 1).

3 Intelligent Cruise Control with GPS

Globally, the ICCG system combines both control and data processing, and it is composed of two main parts: `ControlAutomaton` and `ModeComputation`. The `ControlAutomaton` part on figure 2 represents the automaton which controls the ICCG system by choosing the appropriated running mode to activate. The `ModeComputation` part represents the different running modes of the system which can operate in six different modes: `Alarm`, `Limit`, `Cruise`, `Limit_GPS`, `Cruise_GPS`, and `Cruise_Tracking`, as specified in the dashed box of the figure 2.

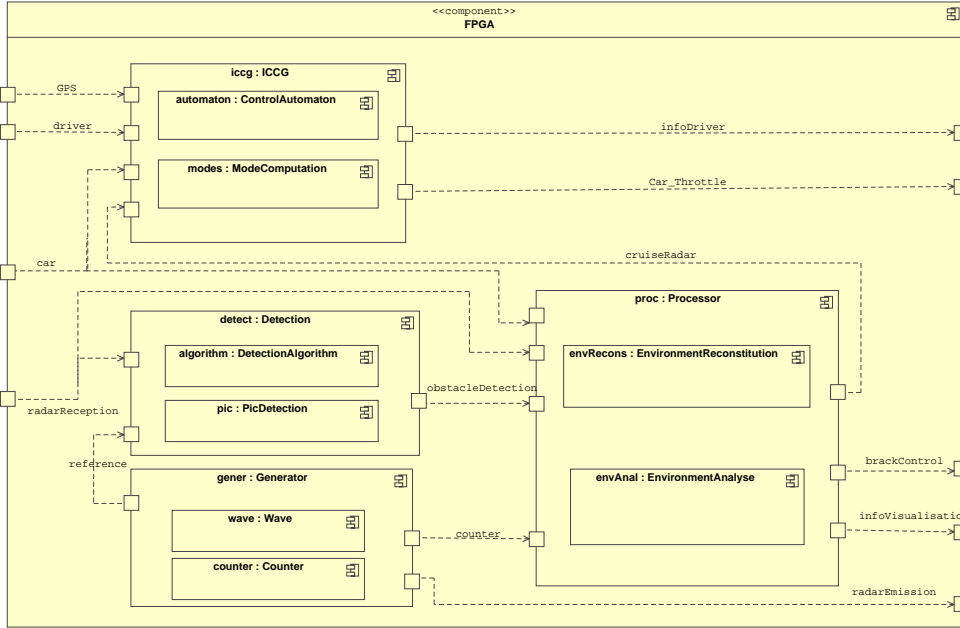


Figure 1. Schematic view of the whole system on FPGA

The interaction with the system and the activation or deactivation of the different modes are done according to input information provided by the environment. These information are given by the GPS system, the driver, the car position, and the radar system. As result, the ICCG system reacts and provides information on the car speed. In order to keep readable the automaton, all transactions are not shown on the figure 2.

3.1 Control Automaton

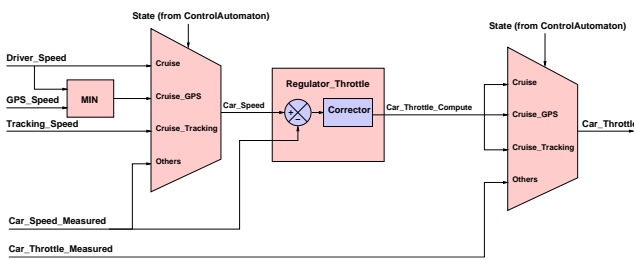


Figure 3. Computation Mode of the states using cruise regulation

The automaton shown on figure 2 permits the transaction between the several modes of the ICCG system, and activates the associated mode. The behavior of these modes

is described in the following.

The Alarm mode is only an informative mode. It indicates the driver, by an audio or luminous signal, that the allowed speed limit, given by a GPS, is exceeded. Nevertheless, the driver controls the car by accelerating or braking, independently of its speed. The Cruise mode maintains the car at a constant speed given by the driver. This speed is not controlled via the accelerator pedal but rather by using a set of buttons. The Limit_GPS mode is similar to Limit mode, where the speed limit is the minimum of the speed required by the driver and the one given by a GPS. The Cruise_GPS mode behaves similarly as the Cruise mode, but the speed limit is calculated in the same way as in Limit_GPS mode. The Cruise_Tracking mode recovers information produced by the radar system and manipulated by the processor to adjust the speed of the car according to that of the car which precedes it. This mode is particularly useful in the convoy case.

The others states of the automaton, represented outside the dashed box, are used to guarantee the safety of the system in some particular cases such as Limit_GPS_Fail, Cruise_Tracking_Fail and Limit_StdB.

3.2 Mode Computation

We have defined the different computation modes according to the design methodology described in [9] and [8]. We focus our study on the behavior of some

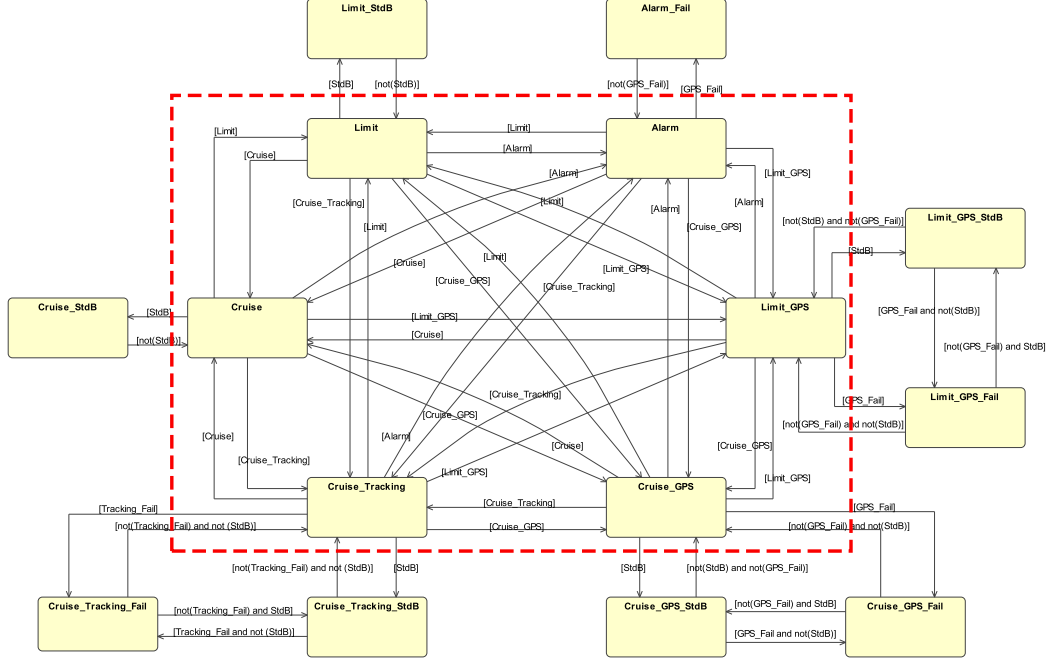


Figure 2. Control automaton of the ICCG system

modes which permit the regulation of the car speed. The concerned modes are Cruise, Cruise_GPS and Cruise_Tracking.

The main part of this regulation system is the Regulator_Throttle, which computes the percentage Car_Throttle_Compute necessary to reach the Car_Speed from the Car_Speed_Measured. In other words, it computes the pressure on the accelerator pedal required to reach or to maintain the vehicle at a given speed.

The current state of the control automaton stage is evaluated in order to provide Car_Speed and Car_Throttle. The Car_Speed is the target speed of the vehicle which can be Driver_Speed, GPS_Speed, Tracking_speed or Car_Speed_Measured. The MIN function is not submitted to control and allows to choose the minimum speed of that provided by the GPS and the driver, in order to ensure the driver safety.

Some experimental simulation and verification processes proved [10] the correct functionality of the ICCG system, including both automaton and computation parts. Currently, we are interested in its implementation on a FPGA.

4 Obstacle Detection

According to the Detection step presented on figure 1, detecting an obstacle is possible thanks to the comparison between the received wave and the emitted one. A

common correlation algorithm [4] performs this comparison and allows the detection of an obstacle located at 100 m 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. One of them is the modification of the detection algorithm.

We decide to implement a modified version of the third Higher Order Statistic algorithm [13, 14]. We aim at increasing the SNR and, in doing so, increasing the maximum detection distance till 150 m. Other algorithms [3, 6, 15] that propose to reduce the noise within the emitted signal may be considered in the future.

4.1 Mathematical Formulation

The mathematical formulation of the Higher Order Statistics (HOS) [13] algorithm is given by:

$$J_{32}(i0) = \sum_{j=0}^{L-1} \left[\frac{1}{N} \sum_{i=0}^{N-1} y(i) \cdot c(i+1) \cdot c(i+j) \right] \cdot \left[\frac{1}{N} \sum_{i=0}^{N-1} y(i) \cdot c(i+1) \cdot y(i+j+i0) \right] \quad (1)$$

The $\frac{1}{N}$ factor normalizes the results. We do not take into account this factor while implementing the algorithm. Thus,

a reduced formula is:

$$J_{32}(i0) = \sum_{j=0}^{L-1} C_{ycc}(j) \cdot C_{ycy}(j + i0) \quad (2)$$

$$C_{ycc} = \sum_{i=0}^{N-1} y(i) \cdot c(i + 1) \cdot c(i + j) \quad (3)$$

$$C_{ycy} = \sum_{i=0}^{N-1} y(i) \cdot c(i + 1) \cdot y(i + j) \quad (4)$$

- **N**: The number of samples in the reference code. In our case, we fix N to 1023 based on previous work [7].
- **L**: The maximum distance between peaks generated by C_{ycc} and C_{ycy} . In our particular case, L is directly equal to the maximum distance detection (in meter) we are targeting: 150.
- **c**: The reference code [12] used to create the emitted wave (to be more precise, the radar signal is modulated with a reference code, and is continuously emitted) generated by a Linear Feedback Shift Registers [11] contained in the Generator step of the figure 1.
- **y**: The received wave (only noise if there is no obstacle, reference code in addition of the noise otherwise). 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 is used N times [7].

4.2 Hardware Implementation

It has been shown in [2] that a hardware implementation of such an algorithm is much more efficient than the software one. Thus, only the hardware implementation description is presented in this article.

The evaluation of the formula 5 is needed to compute both $C_{ycc}(j)$ and $C_{ycy}(j)$. To optimize the implementation, this evaluation is done only once and then sent to both C_{ycc} and C_{ycy} modules, as shown on figure 4.

$$s(i) = y(i) \cdot c(i + 1) \quad (5)$$

Three elements are required to compute C_{ycc} : $y(i)$, $c(i + 1)$, $c(i + j)$. The last two are coded using 1 bit, and can easily be implemented by a multiplexer. The last one, $y(i)$, uses 4 bits.

C_{ycy} structure is similar to the C_{ycc} 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 C_{ycc} .

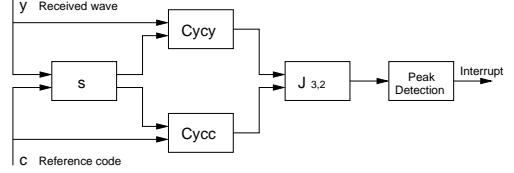


Figure 4. Schematic view of the modified Tug-nait algorithm

$J_{3,2}(i0)$ 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 large design.

Figure 5 shows the schematic view of our hardware implementation of formula 1. The input point, marked **1** on the figure 5, is the received wave, composed of the reference signal and the noise. Mark **2** is the signal produced by the formula 4, while mark **3** is the result of the formula 3. Both mark **2** and **3** permit output computation of the full algorithm, marked **4**. This signal is encoded on 40 bits. It is not directly efficiently usable by a processor and a software task, since the signal contains noise.

As we are targeting software tasks, it is easier to handle interrupt which are directly associated to the obstacle detection. The role of the last stage of the figure 5 is to find the right peak in the noise, and to generate an interrupt, marked **5** once found [7].

We have developed VHDL code corresponding to the presented algorithm using compound approach. The compound approach allows us to get a useful hierarchical view of the system, and makes it possible to re-use some hardware bloc, like the multiplication or the addition blocs, described by a generic way in VHDL.

4.3 Algorithm Behavior

Simulation processes allow us to quickly evaluate the algorithm's behavior. In order to simulate the radar's inputs, we have created an input signal, marked **1** in both figures 5 and 6, composed of the reference code and of a noise which increase with time. The noise function used is based on a classical Gaussian form. Reference code and noise are cyclically emitted, as in real conditions. This fact explains the edges on mark **2** (produced by C_{ycy}), **3** (from C_{ycc}) and **4** ($J_{3,2}$, the correlation results between C_{ycy} and C_{ycc}). The mark **5** is a clean peak that can be sent to the processor via an interrupt.

4.4 FPGA Implementation

The step following the algorithm behavior validation is the implementation on a FPGA. We have to ensure that the

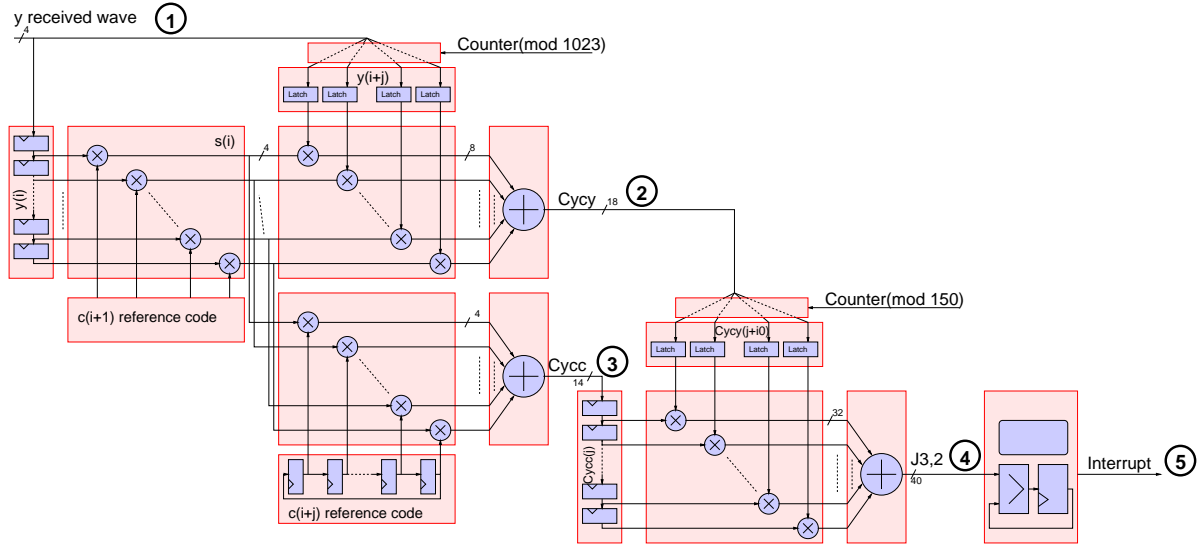


Figure 5. Schematic view of the modified Tugnait algorithm hardware implementation

whole algorithm fits on the targeted FPGA, an Altera Stratix 2, containing 60,000 equivalent logic elements (LE).

The synthesis results of the full algorithm implementation are given on table 4.4. It can be observed that the design does not fit into the targeted FPGA. One of the main reasons is the complexity of the multiplications. The computation of the $J_{3,2}$ module from HOS algorithm requires 150 multiplications of a 15-bit operand by a 19-bit operand. DSP blocks, included in FPGA, can be used to implement some of the multiplications, but not all of them.

Table 1. Synthesis results of the modified Tugnait algorithm implementation

Detection step	Equivalent logic elements used	
s	8 184	100 953
C_{ycc}	11 243	
C_{ycy}	33 749	
J_{32}	47 689	
Peak detection	88	

In order to implement the algorithm on the FPGA, we developed a reduced version, based on average instead of addition. This reduced version introduces a performance decrease in the detection, which has not been estimated. The reduced version requires approximately 40 000 equivalent logic elements (LEs). However, for other FPGA, big enough to support the whole algorithm, will be acquired in order to do testing.

5 Software Tasks

This section describes the part of the global system presented in figure 1 which is implemented on the processor (itself implemented on a FPGA). In this processor, two kinds of applications are implemented. The first one reconstitutes the environment of the vehicle, and mainly use radar detection information. The second one analyzes the environment in order to detect danger and to prevent collision.

5.1 Environment Reconstitution

The algorithm presented in the previous section produces a signal in which an edge is generated whenever an obstacle is detected. According to this detection, distances and speeds can be computed in a sequential way. A soft-core processor and a C program are enough efficient to do this work.

Three elements are needed to compute the distance:

- an edge detection module, which generates interrupts on the processor (Detection module developed in section 4);
- a counter, which computes the wave course time (Generator from figure 1);
- the radar direction, creating a 2-dimensional view of the context in front of the vehicle equipped with the radar.

Interrupts are used to detect edges. For each position, edges and their associated distances are stored in mem-

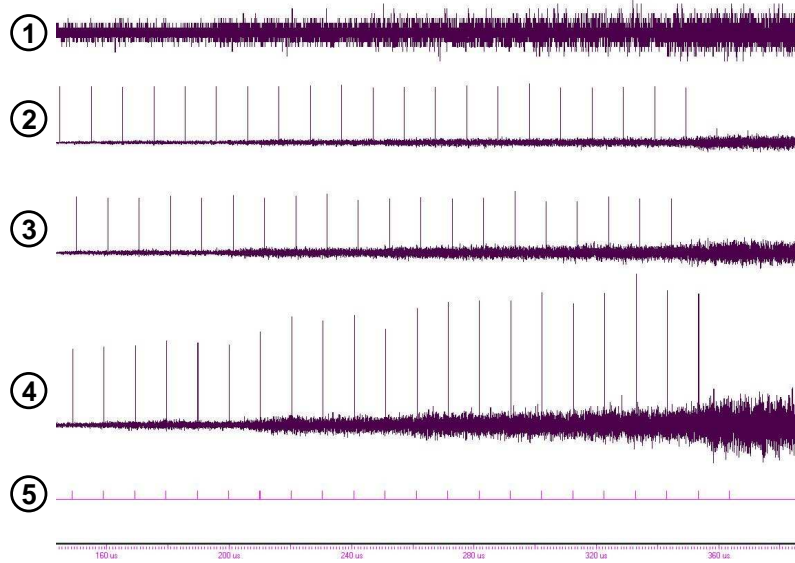


Figure 6. Simulation results of the modified Tugnait algorithm

ory. When the radar position changes, an average distance, based on previously stored data, is computed.

When an object is detected, according to the algorithm described in section 4, the processor stores relevant data and compute distance. These distance information are written to an output VGA screen and are directly associated to the radar orientation. Sixteen different orientations of the radar are possible and the distance is the only information reported on the screen. However, precision of the environment restitution has been developed in order to get an adaptive precision concerning orientation (it is easy to use 64 different orientations for example).

The figure 7 shows an example of what is expected in real condition tests. For these tests, a radar emulator generates a received wave y according to the picture shown on the top of the figure. On the bottom of the figure, all objects placed on the virtual line at a maximum of 150 meters are detected. On the screen output, the column comprised between -1° and 2° corresponds to the car on the road in front of the system detection. Others details appear, like the car situated on the orientation limit of the radar.

The basic exploitation of the whole system including the hardware and the software modules in this visualization on a VGA screen allows to partially validate the behavior of the global system.

In this section, we have shown that we are able to use a processor to analyze the outputs of the algorithm presented in section 4 and to reconstitute the environment. However, we are still not able to anticipate a collision, because of no analysis of the environment has been done. This concept is treated in the following section.

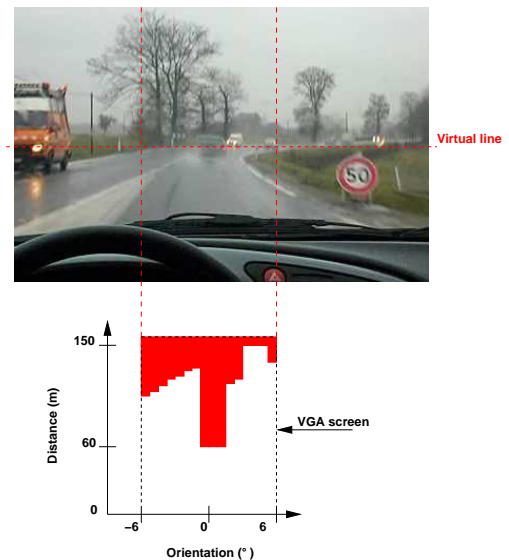


Figure 7. VGA output screen

5.2 Environment Analysis

From the environment reconstitution, no decision can be taken because no interpretation is done. In our system, the interpretation of the environment results on the tracking of different detected obstacles. These obstacles can be immovable, like a tree, or movable, like a car.

In order to anticipate a collision, prediction of the environment must be computed, using complex filtering trans-

formation. These filters require an important computation power. A way to fit their needs is to implement them on a multiprocessor architecture. We do not treat the filtering aspect on this paper, we are currently working on this subject. We suppose that we are able to detect potential collision and to prevent it by pushing the brake pedal.

Using tracking, computed by the processor, we can easily identify an object the car is following. If the object is another car the driver is following, it is interesting to offer to the driver the possibility to adjust the car speed according to that of the previous car. This kind of system is useful in convoy context, where all vehicle share the same destination, and where security distance are rarely respected. Thus, the algorithm we are currently working on shall be able to provide the speed allowing to respect this distance. This speed is computed in order to always keep a sufficient safety distance, and its value is sent to the automaton, as shown on the figure 1. The CruiseRadar signal contains the Tracking_Speed and the Tracking_Fail information (in case that no tracking is feasible, the signal is activated).

6 Conclusion

One objective of the ModEasy project [1] is to develop safety applications in the automotive domain. Within this project, we have defined a reactive system which takes into account four elements: driver controls, vehicle behavior (speed...), radar wave and GPS signals. From these inputs, we target a partial control of the vehicle behavior, mainly an emergency braking or a speed regulation. We have identified the different stages necessary for a safe behavior of the system.

Braking the vehicle is possible thanks to the knowledge of the immediate environment, provided by a radar. The chosen detection algorithm allows a maximum distance detection of 150 meters, whereas 100 meters is the maximum distance offered by a classical one. The algorithm implementation needs important computation power and requires a meticulous implementation. Based on this detection, the environment reconstitution is possible. Using filters, the environment can be analyzed, allowing the anticipation of a potential collision or the tracking of the vehicle followed by the driver.

The complete system also includes an Intelligent Cruise Control with GPS, implemented with an automaton. The operating modes are accessible by the user, but there also exist other modes which only depend on the car behavior. The ICCG system permits a limitation or a regulation of the vehicle speed, potentially using GPS signals. An original operating mode takes into account radar wave interpretation and allows the driver to regulate the car speed according to the previous one. This operating mode is particularly

interesting in a convoy context.

We have validated each part of the global system and implemented the whole on a single FPGA. The implementation of several modules requires both software and hardware development. We are currently validating the global system behavior by simulation, and plan to carry out tests in real conditions.

References

- [1] ModEasy project home page. <http://www.lifl.fr/modeasy/>.
- [2] S. L. Beux, V. Gagné, E. M. Aboulhamid, P. Marquet, and J.-L. Dekeyser. Hardware/software exploration for an anti-collision radar system. Research Report 5820, INRIA, Jan. 2006.
- [3] R. P. C. L. Nikias. Time delay estimation in unknown gaussian spatially correlated noise. In *IEEE trans. on acoustics, speech and signal processing*, volume 36, nov 1988.
- [4] B. Edde. *Radar Principles, Technology, Applications*. Prentice Hall, Sept. 1992.
- [5] B. Fremont, A. Menhaj, P. Deloof, and M. Heddebaut. A cooperative collision avoidance and communication system for railway transports. In *3rd IEEE Conference on Intelligent Transportation Systems*, Oct. 2000.
- [6] C. L. N. H. Chiang. A new method for adaptative time delay estimation for non-gaussian signals. In *IEEE trans. on acoustics speech and signal processing*, volume 38, pages 209–219, Feb. 1990.
- [7] Y. E. 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.
- [8] O. Labbani, J.-L. Dekeyser, and P. Boulet. Mode-automata based methodology for scade. In *Hybrid Systems: Computation and control, 8th international workshop*, LNCS series, pages 386–401, Zurich, Switzerland, March 2005. Springer.
- [9] O. Labbani, J.-L. Dekeyser, P. Boulet, and Éric Rutten. Introducing control in the gaspard2 data-parallel metamodel: Synchronous approach. Research Report 5794, INRIA, Jan. 2006.
- [10] O. Labbani, J.-L. Dekeyser, and Éric Rutten. Separating control and data fbw: Methodology and automotive system case study. Research Report 5832, INRIA, Feb. 2006.
- [11] NASA. Linear feedback shift registers. Technical report, NASA, 2002.
- [12] W. Peterson and E. Weldon. *Error-correcting codes*. MIT press, 1972.
- [13] J. Tugnait. On time delay estimation with unknown spatially correlated gaussian noise using fourth-order cumulants and cross cumulants. In *IEEE transaction on signal processing*, volume 39, pages 1258–1267, June 1991.
- [14] J. Tugnait. Time delay estimation with unknown spatially correlated gaussian noise. In *IEEE transaction on signal processing*, volume 42, pages 549–558, feb 1993.
- [15] M. R. W. Zhang. Nonparametric bispectrum-based time delay estimations for multiple sensor data. In *IEEE transaction on signal processing*, volume 39, pages 770–774, Mar. 1991.