

Dynamically Reconfigurable Architecture for a Driver Assistant System

Naim Harb, Smail Niar, Mazen A. R. Saghir[‡], Yassin El Hillali, Rabie Ben Atitallah
University of Valenciennes, France.

Email: {naim.harb, smail.niar, yassin.alhillali, rabie.benatitallah}@univ-valenciennes.fr

[‡]Texas A&M University at Qatar

Email: mazen.saghir@qatar.tamu.edu

Abstract—Application-specific programmable processors are increasingly being replaced by FPGAs, which offer high levels of logic density, rich sets of embedded hardware blocks, and a high degree of customizability and reconfigurability. New FPGA features such as Dynamic Partial Reconfiguration (DPR) can be leveraged to reduce resource utilization and power consumption while still providing high levels of performance. In this paper, we describe our implementation of a dynamically reconfigurable multiple-target tracking (MTT) module for an automotive driver assistance system. Our module implements a dynamically reconfigurable filtering block that changes with changing driving conditions.

I. INTRODUCTION

Application-specific programmable processors are widely used in embedded systems due to their customized instruction sets and micro-architectural features, which enable them to attain high levels of performance and energy efficiency. However, time-to-market pressures, high design and fabrication costs, and fast changing standards make them a costly and risky investment for embedded system designers. To overcome these problems, designers are increasingly relying on Field Programmable Gate Arrays (FPGAs) due to their high logic densities and rich sets of embedded hardware components, which enable the implementation of highly complex systems [1]. Although FPGAs are slower and consume more power than application-specific integrated circuits (ASICs), recent advances such as dynamic partial reconfiguration (DPR) [2] are helping bridge this gap. In this paper, we explore the use of DPR to implement a dynamically reconfigurable system for automotive multiple target tracking (MTT). MTT is commonly used in Driver Assistance Systems (DAS) that require real-time monitoring of the driving environment and tracking of other vehicles or obstacles on the road. As the driving conditions and environment change, the *type* of processing required for MTT also changes. For instance, when targets move closer to the radar, they should be tracked at higher levels of accuracy since they can potentially become more hazardous. On the other hand, when targets move further away, less accurate tracking can be used. In this paper we show how the functionality and accuracy of an MTT system can be automatically tuned to match the dynamics of moving obstacles on the road. We show how DPR can be used to release hardware resources for other uses, such as tracking more obstacles, accelerating other computational functions, or reducing power consumption.

II. SYSTEM DESIGN

The Multiple Target Tracking (MTT) system uses data from an external radar to keep track of the speed, distance and relative position of vehicles within its field of view. From the functional point of view, the MTT system can be divided into two main blocks: Data Association and Filtering&Prediction (Figure1) [11]. Figure 1 shows the functional blocks of a typical MTT system along with the radar signal processing blocks including our block of interest, the Filtering&Prediction block.

A. The MTT Application and its Filtering&Prediction Block

The MTT application tracks targets by processing data measured by an AC20 TRW radar realizing a scan every 25 ms [9]. This data is processed in three iterative stages. The MTT application reads linear speed, angular speed, distance, and azimuth angle data for each target in the first stage. Each of this data is represented by 32 bits. In the second stage, a filter is used to estimate the location of each target and in the last stage, a prediction is made for the next radar scan [11].

The MTT application can be implemented using a variety of adaptive algorithms. In our system, we have chosen to

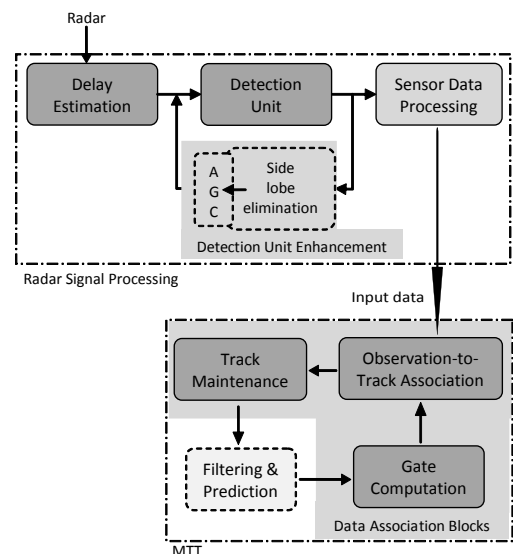


Fig. 1. Full system building blocks.

implement both the Kalman and the α - β filters [10]. The Kalman filter is used because it is an important block in the whole MTT system [11]. A Kalman filter is designed to track a moving object having a constant velocity. However, real objects actually tend to move in variable accelerations and therefore different velocities. The choice of the α - β filter is due to its simple implementation and its consideration of target velocities in decision makings [10].

The Filtering&Prediction block (Figure 1) takes as input actual measured and estimated values of target’s characteristics: distance, linear velocity, angular velocity and azimuth angle. This block produces predicted values of distance and azimuth angle as output.

B. Radar Signal Processing and Enhancement Block

The radar signal processing blocks are responsible for analyzing captured signals and delivering them to the MTT system for decision making. The three main blocks can be seen in Figure 1. The Delay Estimation block is based on correlation and HOS (High Order Statistics) algorithms related to noisy radar signals or an FFT (Fast Fourier Transform) algorithm for continuous wave radars such as the one used in our system. The Detection Unit is an adaptive threshold based signal detector that is responsible for target identification and tracking [13]. The Detection Unit Enhancement block (Figure 1) is used to amplify and filter weak signals detected from distant targets and obstacles. The Sensor Data Processing block realizes data translation and interpretation in order to be delivered to the MTT MPSoC based system .

III. DYNAMICALLY RECONFIGURABLE MTT ARCHITECTURE

Figure 2 shows the architecture of our dynamically reconfigurable MTT system. Our system enables specific hardware blocks to be swapped on the fly. This system is used for validating the MTT’s functionality and measuring reconfiguration overhead.

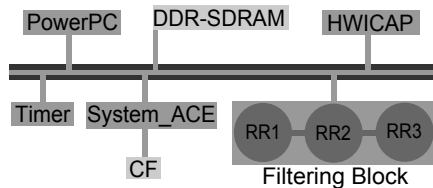


Fig. 2. Implemented MTT system’s testing architecture.

The external Double Data Rate Synchronous Dynamic Random Access Memory (DDR SDRAM) is used to store the instruction and data files used by the MTT algorithm. A timer block is used to measure the latency of the hardware filter block.

The Advanced Configuration Environment (ACE) controller is used to load the configuration bit stream files from the compact flash drive. The Hardware Internal Configuration Access Port (HWICAP) is responsible for configuring the

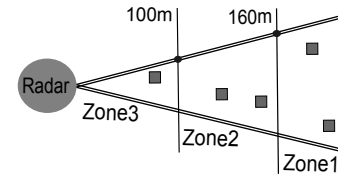


Fig. 3. Radar’s field of view and zones definition.

regions in the FPGA [14]. As we identified three functioning modes, we have three reconfigurable regions noted RR1 to RR3. These three regions constitute the filtering and prediction block of the MTT system.

A. Filter Implementations

Initially we implemented the Kalman filter as a dedicated hardware block in a previous work. Our next goal was to study the ability of implementing different types of filters in reconfigurable regions. The main advantage of this implementation is its ability to implement different filter architectures having different characteristics in response to changing operating conditions.

The three designed filters are used in three different configurations to support three different driving scenarios. These filters include a Kalman filter for angle estimation (KFA), and a Kalman filter for distance estimation (KFD) and an extension α - β filter. The two Kalman filters differ in their building blocks due to the different behavior of the input data. We also designed an extended α - β filter to provide even more accurate predictions for certain driving conditions. An α - β filter is used due to its smoothing characteristics of input data and ability to be mapped on a hardware block. Three reconfigurable regions, RR1, RR2 and RR3 are respectively mapped to three designed filters:

KFA Filter: The Kalman filter for angle estimation (KFA) is a version of the original Kalman filter hardware block. This filter performs only the prediction of the angle.

KFD Filter: The Kalman filter for distance estimation (KFD) is also a version of the original Kalman filter hardware block. KFD allows only the prediction of the distance.

ABF Filter: The α - β filter is a simple filter mainly used for data smoothing and control applications. This filter realizes the prediction of both the distance and angle with the help of linear and angular input velocities.

B. Filtering&Prediction Configurations

The Filtering&Prediction Block configurations are directly related to a set of different combinations among the filters KFA, KFD and ABF. These configurations are associated with three regions: Zone 1 between 160 meters and the maximum radar range, Zone 2 between 100 meters and 160 meters and Zone 3 between 100 meters and 0 meters as shown in Figure 3. These regions are defined according to their proximity to the radar and will be mapped to 3 configurations simultaneously:

Configuration 1 (C1): This configuration uses only the filter KFA. C1 is implemented when there are only targets tracked

far away from the radar or are in Zone 1. These obstacles are considered to have a negligible hazard level. Also the distance estimation for such far away obstacles is considered to have a very low priority. Estimating just the angle for such obstacles is considered as an initiation step in the detection system. This means that when there are targets only in that region, then RR1 will be configured to KFA while RR2 and RR3 will just forward their inputs to the outputs. In addition to this, this block will use different inputs for an enhancement block in the system. Only the signals related to the filtering block will be forwarded without any treatment.

Configuration 2 (C2): This configuration uses both KFA and KFD. C2 will be used when targets are observed in Zone 2. Targets in that region can potentially be more hazardous than those in Zone 1. Thus, a better estimation will be computed for such targets by using estimated of both their distance and angle. RR1 and RR2 will be configured to KFA and KFD, respectively.

Configuration 3 (C3): C3 includes the implementation of all KFA, KFD and ABF at the same time. If a target is spotted in Zone 3, it will be considered to have a high hazard level and a better prediction of its position is needed. This is why we consider ABF an enhancement over KFA and KFD that provides better distance and angle estimation of a target's data. In this configuration, RR1 will be configured as KFA, RR2 will be configured as KFD, and RR3 will be configured as ABF.

In configurations C1 and C2, the free reconfigurable regions are used as an enhancement unit related to the radar signal processing part as mentioned in the following subsection.

C. Detection Unit Enhancement Block

The Detection Unit Enhancement (DUE) block is used to improve the detection of weak target signals captured from far away targets. As mentioned in Section II-B, closer target signals tend to hide farther target signals. In potentially hazardous situations where targets are spotted in Zone 3, the signals from these targets are the most important and are strong enough for analyzing and delivering to the MTT system. Other signals are not considered very important since they are not within the hazardous zone (Zone 3). In situations where targets are only in Zone 2 and Zone 1, and with the free resources liberated from other reconfigurable regions, a better utilization of these resources can be achieved by implementing the DUE block in those regions.

D. Reconfiguration Heuristic

The MTT application can reconfigure the filtering block automatically. The measured distance of any obstacle forms the basis for our reconfiguration heuristic. This information, along with the zone definitions, will provide the decision to switch among the three configurations. For our experimental results we consider a scenario where a target is traveling at 33.3 meters/s (120 km/hr) speed towards the radar [15]. Since the reaction time of the driver should be around 2.7 seconds, a distance of around 90 meters is considered safe and thus defining Zone 3.

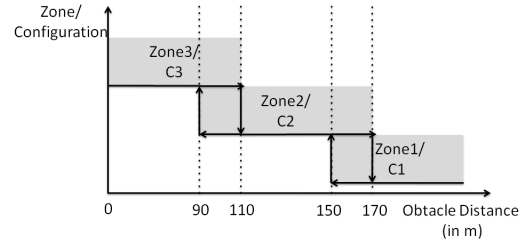


Fig. 4. Reconfiguration triggering distance supposing a speed of 120 km/hr and the 3 configurations

To verify the presence of an obstacle in a given zone, we rely on a window of 12, consecutive 25-msec-radar sweeps. This number is obtained through experimental testing. The value 12 is directly related to the driver's relative linear velocity and the driver's reaction time and is therefore calculated frequently. Twelve sweeps provide an additional margin of 10 meters around each zone threshold. For example if a target is initially spotted in Zone 1, 12 consecutive radar sweeps will be monitored to ensure that this target remains in Zone 1, upon which C1 will be instantiated. If the target moves closer and enters Zone 2, 12 consecutive radar sweeps will also be considered to ensure the target's position as well. These sweeps will be triggered for counting after the target crosses the 160 meters distance measured. Considering that the target is traveling at a speed of 120 Km/hr [15] and that each radar scan is 25 msec long, this will make our target ready to be tracked using C2 at a distance of $160-10=150$ meters. The same applies if the target is entering or leaving Zone 1, Zone 2 or Zone 3. In summary, Figure 4 shows the reconfiguration triggering process based on measured distance.

Regarding the DUE unit, one region (RR3) out of the 2 free regions (RR2 and RR3) in configuration C1 will be implemented to as a DUE hardware unit. The same region will remain a DUE unit when the system changes into configuration C2 since RR3 will not change its configuration but only RR2. In configuration C3, the DUE unit will be replaced by the ABF filter.

IV. RESULTS AND ANALYSIS

In this section, we present our experimental results and compare different MTT implementations in terms of accuracy, latency, reconfiguration overhead and resource utilization. On a Virtex4 FX60 FPGA, the Filtering&Prediction block needs 15 ms to execute from a total execution time of 70 ms (22%). This block also occupies 26,000 logic elements and 4 KByte for its instruction cache [11].

A. Accuracy

Our first set of experiments were designed to measure the accuracy of different filter implementations. We use actual radar distance, angle, linear velocity and angular velocity measurements with different filtering block configurations. Here we present only the results for distance estimation with the 3 configurations C1, C2 and C3. We compared the

distance of a target captured by the radar (system's input), its actual position, the output when implementing C2 (KFA+KFD output) and that when implementing C3 (KFA+KFD+ABF output).

When implementing C1, the output distance from the filtering block is identical to the measured distance. On the other hand, our results show that the angle estimation error can reach a maximum of 1.8 degrees (15%). This is similar to the angle estimation error achieved when implementing C2. However, since C2 filters the measured distance, it results in a distance estimation error that reaches a maximum of 6 meters (4%). Implementing C3 results in the lowest distance and angle estimation errors, which reach a maximum of 3.2 meters (2%) and 0.7 degrees (7%), respectively. This shows that the extended ABF reduces distance and angle estimation errors by 50%.

B. Latency and Reconfiguration Overhead

Latency analysis in driver assistant systems is very important to improve safety and ensure that drivers have adequate time to react to changing conditions. Starting from this point, we base our analysis on scenarios where targets travel towards the radar at 33.3 meters/s (120 km/hr).

Since the reaction time of the driver should be approximately 2.7 seconds [15], this results in a $33.3 \times 2.7 = 90$ meter safety buffer that we use to define Zone 3.

Regarding the implementation of the KFA, KFD and ABF filters, we note that they each have a latency of less than 50 ns. Since these filters are in turn used to implement each of the three configurations, we note that the latency of configuration C1, which only uses KFA, is 50 ns. Similarly, the latency of configuration C2, which uses filters KFA and KFD in parallel, is also 50 ns. On the other hand, the latency of configuration C3, which uses the ABF to process the outputs of the KFA and KFD filters is 100 ns. Although pipelining can be used to maintain a 50 ns clock, the additional hardware resources and system complexity are not necessary. Since the data transfer latency from a soft processor core to the filtering block is on the order of micro-seconds, the 100 ns latency of the Filtering&Prediction block is not on the critical path. The times needed to perform partial reconfigurations to KFA, KFD and ABF are shown in Table I.

These results enable us to infer the number of radar sweeps that can be missed without filtering while reconfiguration is under way. The number of missed radar sweeps also enable us to validate the defined zones and the heuristic used to perform reconfiguration. For example, the AC20 radar has a maximum detection range of 200 meters. When a target is detected in

Zone 1 at a distance of 200 meters traveling at 33.3 meters/sec (120 Km/hr), four radar sweeps (100 ms) will be skipped due to the reconfiguration process. This still enables configuration C1 to be ready while the target is at $200 - 3.33 = 196.67$ meters. Using the same approach, our system will run on C2 when the target is at distance 146.67 meters and on C3 when the target is at 85.005 meters, if the target is traveling at 120 Km/hr respectively.

V. CONCLUSIONS AND FUTURE WORK

In this work a dynamically reconfigurable filtering hardware block for multiple target tracking applications in Driver Assistant Systems (DAS) was presented. Our system shows that there will be no reconfiguration overhead because the system will still be functioning with the original configuration until the system reconfigures itself. The free reconfigurable regions can be implemented as improvement blocks for other DAS system functionalities. For example all RRs can be KFAs for three radars attached to the MTT system. For future work we will look at optimizing the implementation of reconfigurable hardware blocks. We will also attempt to formalize the heuristics used to switch between different hardware implementations and develop more sophisticated algorithms that can assess various trade-offs (e.g. latency vs. energy consumption) in real time.

REFERENCES

- [1] K. Parnell, "The Changing Face of Automotive ECU Design", *XCell Journal*, Xilinx, Second Quarter, 2005.
- [2] J. Becker et. al., "Dynamic and Partial FPGA Exploitation", *Proceedings of the IEEE*, pp. 438-452, Vol. 95, No. 2, Feb, 2007.
- [3] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control", *IEEE Transactions on Intelligent Transportation Systems*, Sept. 2003.
- [4] "The European FP7, Prevent-Intersafe project", http://www.prevent-ip.org/en/prevent_subprojects/intersection_safety/intersafe/.
- [5] J. Saad, A. Baghdadi and F. Bodereau, "FPGA-based Radar Signal Processing for Automotive Driver Assistance System", *IEEE International Workshop on Rapid System Prototyping, RSP 2008*.
- [6] <http://www.mobileye-vision.com>
- [7] C. Claus et Al., "Using partial-run-time reconfigurable hardware to accelerate video processing in driver assistance system", *Proceedings of the conference on Design, automation and test in Europe 2007*.
- [8] C. Claus, R. Ahmed, F. Altenried, W. Stechele, "Towards rapid dynamic partial reconfiguration in video-based driver assistance systems", 6th International Symposium on Applied Reconfigurable Computing, ARC 2010.
- [9] "Cognitive Safty Radar Systems", www.trw.com.
- [10] S. Blackman, R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House Publishers, 1999.
- [11] J. Khan, S.Niar, M.Saghir, Y.El-hillali, A.Rivenq, "An MPSoC Architecture for the Multiple Target Tracking Application in Driver Assistance System", *Proc. of the 19th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pp. 126-131, IEEE, 2008.
- [12] M. Liu, W. Kuehn, Z. Lu, A. Jantsch, "Run-time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration". *FPL 2009*.
- [13] M. A. Richards, *Fundamentals of Radar Signal Processing*, McGraw-Hill, 2005.
- [14] *OPB HWICAP Data Sheet*. DS 280. <http://www.xilinx.com>.
- [15] X. Ma and I. Andreeasson, *Driver reaction time estimation from real car following data and application in GM-type model evaluation*, In Proceedings of the 85th TRB annual meeting, Washington D.C., 2006.

	Reconfiguration Time
0-C1 / C1-0	87.3 ms
C1-C2 / C2-C1	82.8 ms
C2-C3 / C3-C2	130.1 ms

TABLE I

RECONFIGURATION TIMINGS FOR DIFFERENT CONFIGURATIONS.