Exploring High-Level Synthesis Tools for Vehicle Perception Tasks

Mokhtar Bouain, Denis Berdjag, Rabie Ben Atitallah

Exploring High-Level Synthesis Tools for Vehicle Perception Tasks

Conference Paper · February 2018

CITATIONS
0

READS
11

3 authors:

Mokhtar Bouain
University of Valenciennes and Hainaut-Cambr…
8 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Denis Berdjag
University of Valenciennes and Hainaut-Cambr…
40 PUBLICATIONS **176** CITATIONS
SEE PROFILE

Rabie Ben Atitallah
University of Valenciennes and Hainaut-Cambr…
78 PUBLICATIONS **450** CITATIONS
SEE PROFILE

Exploring High-Level Synthesis Tools for Vehicle Perception Tasks

Mokhtar Bouain, Denis Berdjag and Rabie Ben Atitallah
University of Valenciennes, CNRS, UMR 8201- LAMIH, F-59313 Valenciennes, France
Email: {mokhtar.bouain, denis.berdjag, rabie.benatitallah}@univ-valenciennes.fr

***Abstract*—Building Intelligent Transportation Systems (ITS) requires the use of different types of sensors to improve the traffic safety and to ensure an efficient perception. These systems, integrate an increasing number of tasks and require a significant computing power. To develop a base computing platform that can be quickly modified and scaled to meet the cost and performance targets, FPGA platforms represent an appropriate solution that offers modular and scalable architecture. Moreover, with the complexity of applications and the delicacy of manual coding, the High-Level Synthesis (HLS) tools are used to accelerate the creation of the Register Transfer Level (RTL) designs. In this paper, we explore the HLS tools to generate the RTL designs for vehicle perception tasks based on the Zynq-7000 System On Chip (SoC).**

***Index Terms*—High-level synthesis, intelligent vehicles, sensor fusion, Zynq SoC.**

I. Introduction

Conventionally, for each intelligent transportation task, there is an electronic control unit (ECU). In practice, various sensors acquire the vehicle surroundings and hence the sensor data is analyzed and merged in the ECUs. These electronic devices are often simple micro-controllers which are not scalable and do not have the sufficient processing power to process the incoming data from different and heterogeneous sensor inputs such as cameras, radars, LIDAR sensors... Indeed, automotive systems now integrate an increasing number of features aiming at providing active safety and then full autonomy. For example, obstacle detection (e.g vehicle and pedestrian), lane detection and drivable surface detection are presented as three important applications for visual perception using camera sensors. The LIDAR sensors provide various applications such as detection and tracking of static and moving objects, localization...

Moreover, these applications require high performance systems. On the other hand, CPU, GPU, FPGA and ASIC are discussed as the major components and the appropriate solutions to build an efficient hardware platform for real-time operations.

Otherwise, very few works are addressed to merge these different applications in one embedded platform for autonomous vehicles. In particular, the applications address the detection and tracking of moving objects using multi-sensor data fusion. Furthermore, these tasks operate in uncertain and unknown environments. They should adapt their functioning mode to provide reliability, fault-tolerance, deterministic timing guarantees, and energy efficiency.

To develop a base computing platform that can be quickly modified and scaled to meet the cost and performance targets, FPGA platforms represent a potential solution that offer more features and benefits according to conventional architecture [1].

Otherwise, the conventional design methodology based on the manual encoding of systems has several constraints related to the cost, the risk of error/bug during the integration of the Hardware/Software (HW/SW) parts, verification, validation and development time.

Indeed, with the complexity of applications and the delicacy of manual coding, designers need a direct link between specification and implementation to satisfy the constraints of embedded systems, namely: the miniaturization of Time-To-Market (TTC), the computing performance, the cost of design and the energy consumption. To overcome these issues, several design approaches that are acting with the severe limitations of the conventional methods are used. These approaches, called HLS, permit to accelerate the Intellectual Property (IP) creation by enabling a single HW/SW specification to be directly targeted into programmable devices without the need to manually create the RTL designs.

On the other side, since vehicle perception tasks require a significant computing power and often need updates, the HLS tools can be an appropriate solution allows generating C or VHDL codes for embedded real-time applications. In this paper, we explore the HLS tools to generate the RTL designs for vehicle perception tasks based on the Zynq-7000 SoC.

The rest of this paper is organized as follows. Section II details the existing work in this area. Section III describes the classification of the HLS approaches based on the input language. In section IV we present the multi-sensor data fusion architecture. In section V, we present our design space exploration and we conclude this paper in section VI.

II. Related works

To achieve both high performance and low cost, researchers have proposed their vehicle perception tasks based on three kinds of HW platforms: simple homogeneous SoC such as microcontrollers, FPGA based platforms [2], and heterogeneous SoC platforms [3]. Moreover, with the complexity of the vehicles perception applications and the limitations of the manual coding, developers need a direct link between specification and implementation. To overcome to these constraints

and limitations, the HLS tools appear as potential solutions. A general survey about the commercial and academic HLS tools are given in [4]. It aims to present comprehensive analysis of the recent HLS tools methodology and evaluate them based on a common set of C langage benchmark.
Otherwise, the works presented in [5] evaluate and compare the HLS and the Hardware Description Language (HDL)-based design methodologies by simulating and synthesizing an automotive RADAR signal processing algorithm for Xilinx Virtex 7 FPGA. The used tool is Vivado HLS. The RADAR signal processing algorithm allows to detect the positions and estimates the velocities of targets. The chosen criteria for the evaluation and comparison are TTC, speed (timing analysis), and area (resource utilization). For their HLS implementation, they achieved an overall speed up of about 2X when compared to the HDL-based design (manual coding) which is a significant improvement while keeping the overall resource utilization at under 5% with respect to the available resources. The authors of [3] explore a Multiple Target Tracking (MTT) implementation for an automotive RADAR based on the Zynq-7000 SoC. The MTT system processes multiple target observations obtained from the radar during each scan. For each observation a Kalman Filter is associated. These works emphasize the usage of Vivado HLS to obtain a reduction of more than 65%, both in execution time and energy consumption.
The works done by [2] use Vivado HLS to implement the Semi-Global Matching (SGM) algorithm, which is frequently used in stereo vision systems, e.g. for automotive applications. The HW implementation is based on the Xilinx Virtex 7 FPGA. The works describe how the HLS tool permits to obtain very different RTL designs based on various alternatives. The design cycle is reduced for different performances while maintaining a better quality of results. In [1] we present an embedded multi-sensor data fusion design for vehicle perception tasks. The presented works use Vivado HLS to generate the RTL design of the Sum of Absolute Difference (SAD) stereo matching algorithm in the navigation context. This algorithm allows to create the disparity map. It is used for the object detections surrounding an intelligent vehicle. The proposed design is based on the Zynq-7000 SoC.
According the previous cited works, the HLS tools are used for the ADAS tasks. In our works, we are interesting to use them for the autonomous vehicle.

III. OVERVIEW OF HIGH-LEVEL SYNTHESIS TOOLS

HLS tools start from a software programmable High-Level Language (HLL) (e.g., C, C++) to automatically produce a circuit specification in HDL that performs the same function. HLS tools offer benefits to software developers by using the HW architectures without having to build up HW expertise [4]. Based on the design input language, we distinguish three major categories for the academic and commercial HLS tools which are: **(1) C-Based Design [1], [3]:** This approach allows to software engineers; based on their HW knowledges; to produce the HW modules directly from C/C++ language such as Vivado HLS. **(2) Model Driven Architecture (MDA)[6] :** This approach is employed in the development process which is based on the models written in UML language such as Gaspard. **(3) Model-Based Design (MBD) [7]:** The generation of the VHDL or C codes are provided from a block model tool such as Matlab/Simulink or Labview. Matlab includes two main tools which are Embedded Coder allows to generate C code and Hdl-coder ensures the generation of the VHDL/Verilog code.

IV. MULTI-SENSOR DATA FUSION FRAMEWORK

A. Multi-sensor data fusion (MSDF) architecture

Figure 1 shows the structure of MSDF generic framework for n given sensors. Each sensor provides a list of positions of the detected objects in surrounding according to its Filed Of View (FOV) and characteristics. Our goal is to combine all the lists. This structure consists of the following tasks : **(1) Sensor alignment process (off-line):** The inputs of this process are sensors data while the outputs are the calibration parameters (rotation matrix and translation vector). This process allows finding the transformation between the different sensor frames. **(2) Object detection process (on-line):** In this step, there are n processing chains, each of them provides a list of the detected objects. With the conjunction of the calibration parameters obtained from sensor alignment process, we are able to combine the data to better detect the objects in the surroundings. For more details, the interested reader is referred to [1]. **(3) Tracking module (on-line)**: This module allows to track the static and moving objects. The tracking module includes the following steps : **(1)** The data association between the measurements (new detected object) and the existing tracks. The goal of data association is to select the measurements which will be used to update the target state estimate. We quote the Multiple Hypotheses Tracking (MHT) approach which is used to solve this problem [8]. **(2)** Maintenance of the detected objects and taking the decision about the tracks: created, confirmed or deleted. **(3)** The tracking method precise how we track the objects of the fused list. The Kalman filter (KF) is an appropriate solution [8] for this task.

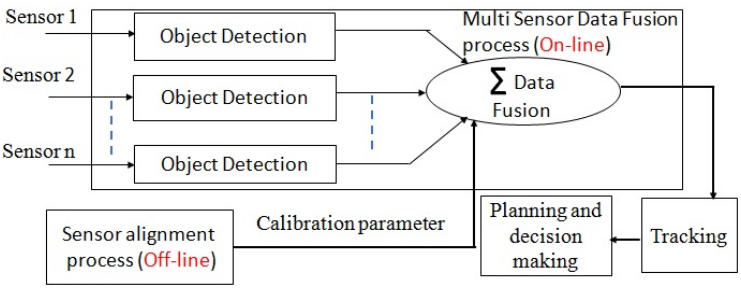

Fig. 1: Multi-sensor data fusion architecture

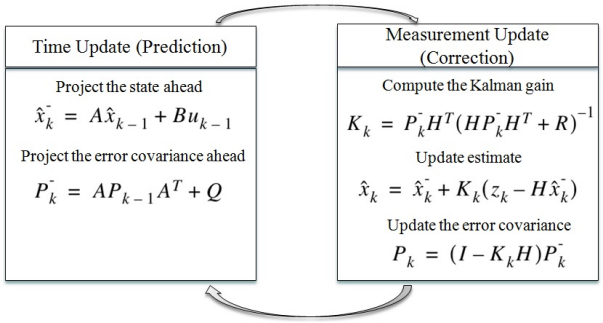

Fig. 2: Operation of KF [9]: prediction and correction

B. Kalman filter

An autonomous vehicle is located in uncertain and unknown environments. Consequently, it is surrounded by different types of objects such as pedestrians, cars, trucks, bikes... These objects are moving in different directions using different motion models. Moreover, our tracking module is based on the KF and since there are different motion models, we have to use different KF to handle these models. Without going into too much detail, there different types of the KF (e.g. discrete, extended) for different motion models [9], [8]. In this work, we are interesting to the discrete KF for linear motion model to track the obstcales. We choose the KF to be implemented as a HW accelerator because it uses diffeents matrix operations which need high performance computing. Figure 2 shows the operation of KF where: $\widehat{x} \in \mathbf{R}^n$ is the state, $z \in \mathbf{R}^m$ is the measurement, $u \in \mathbf{R}^l$ is the optional control input, A is $(n \times n)$ transition matrix, B is $(n \times l)$ control matrix, H is $(m \times n)$ observation matrix , P is $(n \times n)$ covariance matrix, Q is the process noise covariance matrix and R is the measurement noise covariance matrix [9]. Moreover, the 'super minus' is a priori state and k is the time step. The KF involves two stages: prediction and measurement update [8]. Note that the state $\widehat{x}$ is used to estimate the position and the velocity of the objects in 2D trajectory.

V. Design space exploration

A. Platform for implementation

We synthesise and implement our design using the Zynq-7000 SoC. It is about an heterogeneous SoC integrates dual-core ARM Cortex-A9 based on processing system (PS) and programmable logic (PL) in a single device. The communication between the PS and PL is achieved by the Advanced eXtensible Interface (AXI) bus.

B. Measurements of power consumption

The Zynq-7000 SoC is a good candidate for building energy efficient systems due to their low power consumption. We measure the power consumption of the Zynq-7000 SoC using the Texas Instruments power measurement adaptor that connects to the PMBus of the Zynq-7000 SoC board. Figures 3 and 4 show ;respectively; the average value of power consumption of the PS and the PL (Standby mode). It is about

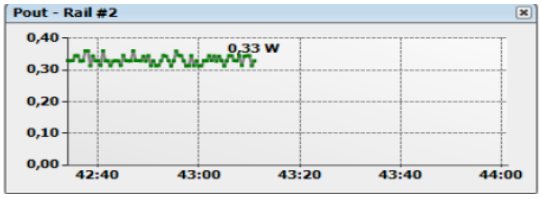

Fig. 3: Power consumption of PS during the execution of the embedded LINUX operating at 667 MHz (Standby mode)

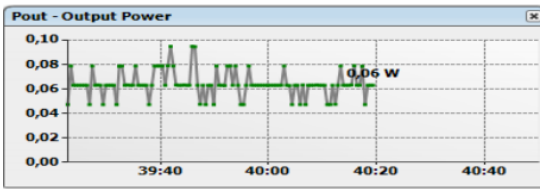

Fig. 4: Power consumption of FPGA operating at 200 MHz

0.33 watt operating at 667 MHz for PS and 0.06 watt operating at 200 MHz for PL.

C. Design flow of Matlab/Simulink framework: The KF case-study

Figure 5 shows the design flow of Matlab/Simulink. The Embedded Coder tool is used to generate the C code while the HDL Coder tool is used to generate the VHDL code. Therefore, we obtain the same application which can be executed either on the processor or on the FPGA giving us more flexibility and choice.

D. IP core user design using AXI4-Stream

Figure 7 shows a streaming data transfer between the PS (ARM processor) and PL (FPGA) into the Zynq-7000 SoC platform. The data transfer uses the AXI4-Stream interface which allows a high-speed streaming data. It is coupled with the Direct Access Memory (DMA) controller to transfer a

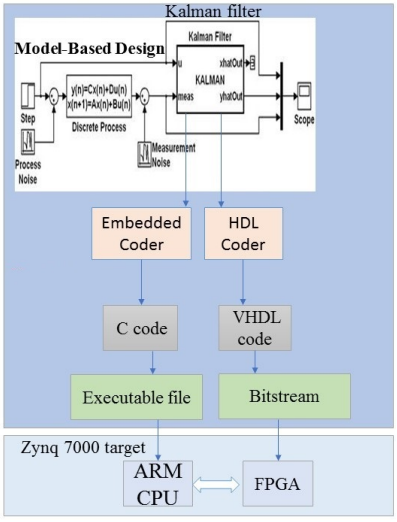

Fig. 5: Design flow of Matlab/Simulink tool

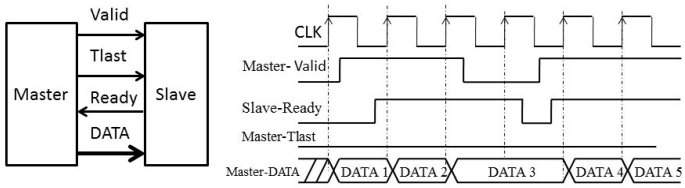

Fig. 6: The I/O of Master/Slave devices and their timing diagram

large amount of data from the PS to PL. The DMA controller reads the data from memory, and sends it using the stream mode to the IP core user (KF) through the AXI4-Stream interface. In our case, the PS sends the measurments and the PL sends state $\widehat{x}$. On the other hand, the AXI4-Lite bus is used for simple or low-throughput memory-mapped communication such as control signal or status registers.

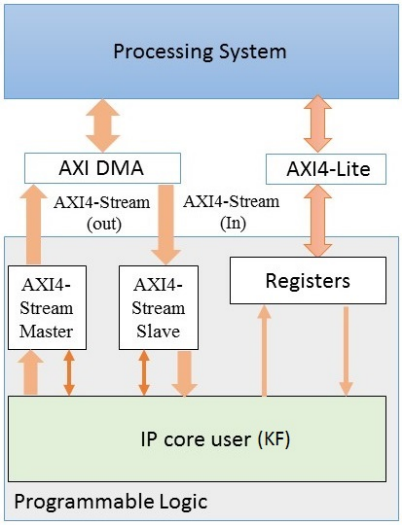

Fig. 7: IP core user (KF) design using AXI4-Stream interface

E. Implementation of Kalman filter using HDL Coder

Based on the design presented in Figure 7, we aim to implement the KF using the HDL coder, the HLS tool of Matlab. The ARM processor sends the data to the IP core (KF) through the AXI4-Stream interface guaranteeing a high-speed streaming data. Figure 6 shows the communication between the two modules (master and slave) as well as their timing diagram. Note that there are others I/O signals but we illustrate the main of them.

The AXI4-Stream interface contains data and control signals. The main control signals are: **(1) Valid signal:** When the data signal is valid, this signal is asserted. **(2) Ready signal:** Indicates whether the slave device can accept new data. **(3) TLAST signal:** This signal permits to indicate the last data.

We use the HDL Coder of Matlab R2017b. The operating frequency of PL is 200 MHz. The synthesis results for the proposed design are illustrated in Table I. According to this table, the HW resource utilization is low which allows to integrate more functionalities. Also, the maximum power consumption of the SoC (PS+PL) is about 1.2 watt which is still lower than the power consumed by the other industrial solutions such as PCs.

Resources	DSP slices	FF	LUT	BRAM36
Usage	211	10233	8435	5

TABLE I: SYNTHESIS RESULTS

VI. Conclusions

With the progress of ITS, vehicles are able to detect surroundings using a variety of proprioceptive and exteroceptive sensors. In this paper, we present a MSDF architecture for embedded systems. Our goal is to build an intelligent vehicle perception implemented on the heterogeneous Zynq platform using the HLS tools. Indeed, with the complexity of applications and the delicacy of manual coding, the HLS tools permit to accelerate the creation of the RTL designs. We explore the HDL Coder tool through the KF which is often used for different tasks (tracking, fusion...). Our future work will be focused to deploy different perception tasks using LIDAR, RADAR and camera sensors and combine the different data to improve the detection and tracking of moving objects.

References

[1] M. Bouain, K. M. A. Ali, D. Berdjag, N. Fakhfakh, and R. Ben Atitallah, "An embedded multi-sensor data fusion design for vehicle perception tasks," *JCM (Journal of Communications)*, vol. 13, no. 1, pp. 8–14, 2018.

[2] A. Qamar, F. B. Muslim, and L. Lavagno, "Analysis and implementation of the semi-global matching 3d vision algorithm using code transformations and high-level synthesis," in *Proceedings of the 81st Vehicular Technology Conference (VTC Spring), Glasgow, United Kingdom*. IEEE, 2015, pp. 1–5.

[3] G. Zhong, S. Niar, A. Prakash, and T. Mitra, "Design of multiple-target tracking system on heterogeneous system-on-chip devices," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4802–4812, 2016.

[4] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi *et al.*, "A survey and evaluation of fpga high-level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1591–1604, 2016.

[5] S. Luthra, "High level synthesis and evaluation of an automotive radar signal processing algorithm for fpgas," 2017.

[6] A. Gamatié, S. Le Beux, É. Piel, R. Ben Atitallah, A. Etien, P. Marquet, and J.-L. Dekeyser, "A model-driven design framework for massively parallel embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 10, no. 4, p. 39, 2011.

[7] L. Zhang, M. Glaß, N. Ballmann, and J. Teich, "Bridging algorithm and esl design: Matlab/simulink model transformation and validation," in *Languages, Design Methods, and Tools for Electronic System Design*. Springer, 2015, pp. 189–206.

[8] H. Durrant-Whyte and T. C. Henderson, *Multisensor data fusion*. Springer, 2008.

[9] G. Bishop and G. Welch, "An introduction to the kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, no. 27599-23175, p. 41, 2001.