

An MPSoC Architecture for the Multiple Target Tracking Application in Driver Assistant System

Jehangir Khan¹, Smail Niar^{1,2}, Atika Menhaj¹, Yassin Elhillali¹, Jean Luc Dekeyser²
¹University of Valenciennes France, ²INRIA Lille Nord Europe France
{jehangir.khan,smail.niar,atika.menhaj,yassin.elhillai}@univ-valenciennes.fr
jean-luc.dekeyser@inria.fr

Abstract

This article discusses the design of an application specific MPSoC architecture dedicated to Multiple Target Tracking (MTT). This application has its utility in driver assistant systems, more precisely in collision avoidance and warning systems. An Automotive-radar is used as the front end sensor in our application. The article examines the tradeoffs that must be taken into consideration in the realization of the entire MTT application in an embedded system. In our implementation of MTT, several independent parallel tasks have been identified and mapped onto a multiprocessor architecture to ensure the deadlines imposed by the application. Our study demonstrates that the joint utilization of reconfigurable circuits (namely FPGA) and MPSoC, facilitates the development of a flexible and efficient MTT system.

1. Introduction

The lives of drivers and passengers are put in danger by bad weather and low visibility conditions. Misjudgment of delicate situations on the part of drivers and physical or mental fatigue due to stressful driving conditions further aggravate this danger. Driver Assistant systems help drivers take correct and quick decisions in delicate situations. These systems provide the driver a realistic assessment of the dynamic behavior of potential obstacles well before it is too late to react.

The purpose of target tracking is to collect data from the sensor field of view (FOV) containing one or more potential targets of interest and to partition the sensor data into sets of observations, or tracks that are produced by the same source [1]. In context of driver assistance for road safety applications, a target tracking system detects and monitors the dynamic behavior of one or more obstacles in the way of the host vehicle. Since in a real traffic scenario, the host vehicle comes across more than one obstacle, typically other vehicles, the driver assistant system falls in the domain of Multiple Target Tracking (MTT).

Our MTT system has two main features. First, as it is radar-based, it has the advantages of longer range as

compared to vision based systems. It performs better in bad visibility conditions and has lower computational requirements [3]. Radar based multi-target tracking is achievable with relatively more ease and less complexity as compared to vision based tracking. Moreover, longer radar range helps detect obstacles from a longer distance and hence ensures longer reaction time for vehicle drivers.

The second feature i.e. reconfigurable & reprogrammable MPSoC architecture, capitalizes on advances in FPGA fabrication technology allowing a cost effective implementation of complex embedded applications. Charting of processor properties over the last three decades shows that the performance of a single processor has levelled off in the last decade. Dedicated hardware implementation may be useful for high speed processing but it does not offer the flexibility and programmability required for system evolution. Applications with stringent resource-consumption and runtime constraints are increasingly resorting to MPSoC architectures. The move to MPSoC design elegantly addresses the power issues faced on the hardware side. Creating multiple processors that execute at lower frequency, results in comparable overall performance in terms of instructions per second while allowing designers to slow down the clock speed, which is a major constraint for low power designs [13].

2. The application

2.1. Terminology

In context of target tracking applications, a *target* represents an obstacle in the way of the host vehicle. With every obstacle is associated a state which is represented as a vector that contains parameters defining target's position and dynamics in space e.g. its distance, speed, azimuth or elevation etc. A state vector with "n" elements is called "n-state" vector. A concatenation of target states defining the target trajectory or movement history at discrete moments over time is called a *track*. As detailed below in MTT, obstacle tracking uses 3 components: the *Observation*, which corresponds to the measurement of a target's

state by a sensor (radar) at discrete moments in time. It is one of the two representations of the true state of the target. The other representation is a calculated “*guess*” or ***prediction*** of the target’s true state before the observation arrives.

Taking into account the *observation* and the *prediction*, an ***estimate*** about the true target state is made. Estimate is the corrected state of the target that depends upon the variances of both observation and prediction. We use Kalman filter algorithm for state prediction and state estimation. Kalman filter is a recursive Least Square Estimation (LSE) algorithm with its origins in estimation theory. In this paper, the term ***scan*** is used to name the periodic sweep of radar field of view (FOV) giving observations of all the detected targets.

2.2. Multiple target tracking building blocks

A simplified view of the basic building blocks of Multiple Target Tracking (MTT) system is shown in figure 1. The system can be divided into two main functions namely *Data Association* and *Tracking Filters*. The two functions work in a closed loop. The data association function is further divided into three sub-functions; “*Track maintenance*”, “*Observation-to-Track Assignment*” and “*Gate Computation*”. Detailed description of these sub-functions is given in the next section. The data association sub-functions are also explained in the next section.

In this project, we use a traditional Kalman Filter to implement the tracking filters. Equations (1) through (5) summarize Kalman filter. Kalman filtering is very popular estimation technique as it is optimal for linear systems having a Gaussian probability distribution for the process noise [4,5].

$$\hat{Y}_k^- = A\hat{Y}_{k-1}^- + Bu_k \quad (1)$$

$$P_k^- = A P_{k-1}^- A^T + Q \quad (2)$$

$$K = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (3)$$

$$\hat{Y}_k = \hat{Y}_k^- + K(Z_k - H\hat{Y}_k^-) \quad (4)$$

$$P_k = (I - KH) P_k^- \quad (5)$$

where

\hat{Y}_k = State estimation vector for time instant k

\hat{Y}_k^- = State Prediction vector for time instant k

K = Kalman gain matrix

Z_k = Observation vector at time instant k

H = Measurement (observation) matrix

A = State transition matrix

B = Control Matrix

u_k = Control signal

P_k^- = Prediction error covariance matrix

P_k = Estimation covariance matrix

R = Measurement noise covariance matrix

Q = Process noise covariance matrix

I = Identity matrix of the same dimensions as P_k

A very important part of data association function is the assignment solver. The assignment problem is summarized as follows: Given a cost matrix C of elements C_{ij} , find a matrix $X = \{x_{ij}\}$,

such that $C = \sum_{i=1}^n \sum_{j=1}^m c_{ij}x_{ij}$ is minimized.

$$\text{Subject to } \begin{cases} \sum_i x_{ij} = 1 \forall j \\ \sum_j x_{ij} = 1 \forall i \end{cases}$$

In our application, c_{ij} represents the cost of associating observation i with track j . This cost is the statistical distance between the observation i and prediction j . Here x_{ij} is a binary variable used for ensuring that an observation is associated with one and only one track and a track is associated with one and only one observation. This requires x_{ij} to be either 0 or 1 i.e. $x_{ij} \in \{0,1\}$.

We use Munkres algorithm to solve the assignment problem for observation to track assignment. The assignment problem and its solution with Munkres algorithm are detailed in [9]. Other techniques like auction algorithm may also be used.

2.3. Functioning of MTT

Assuming recursive processing as shown by the loop in figure 1, tracks would have been formed on the previous *scan*. Now the observations are received from the sensor and the processing loop is to be executed. Incoming observations are first considered by the “*Gate checker*” for the update of the existing tracks. Gating tests determine which possible “*observation-to-track*” pairings are reasonable, by attributing a cost to each pairing. The costs are calculated as the statistical distance between the *prediction* of the target states given by the filters and the *observed state* coordinates received from the radar. These costs are put together in a *cost matrix* which is then passed on to the *assignment solver* to determine the finalized pairings. The pairings are made in a way to ensure minimum total cost for all the pairings.

The finalized observation-track pairings are passed on to the tracking filters which use them for estimating the current states of targets and predicting the next states as well as the *error covariance* associated with these predictions.

The predicted states and predicted error covariance are used by the “*Gate compute*” function to define probability *gates* or windows around the predicted states. The dimensions of the gates being dictated by the prediction error covariance, these gates demarcate the probability boundaries for the next state coordinate measurements. The “*Gate Compute*” sub-function can be viewed as a first level of “*sorting out*” the unlikely target-track associations in case of multiple observations falling close to a single prediction or vice

new track while 3 consecutive misses out of 5 scans for an existing target prompts the deletion of its track. MTT is explained at length in [1] and [6].

The “*Tracking filters*” block in figure 1, requires special attention since the number of filters instantiated, is equal to the maximum number of targets to be processed. In our current work we have fixed this number at 10. In the final system we will increase it up to 20. This limit is being imposed by the radar we are using, which can

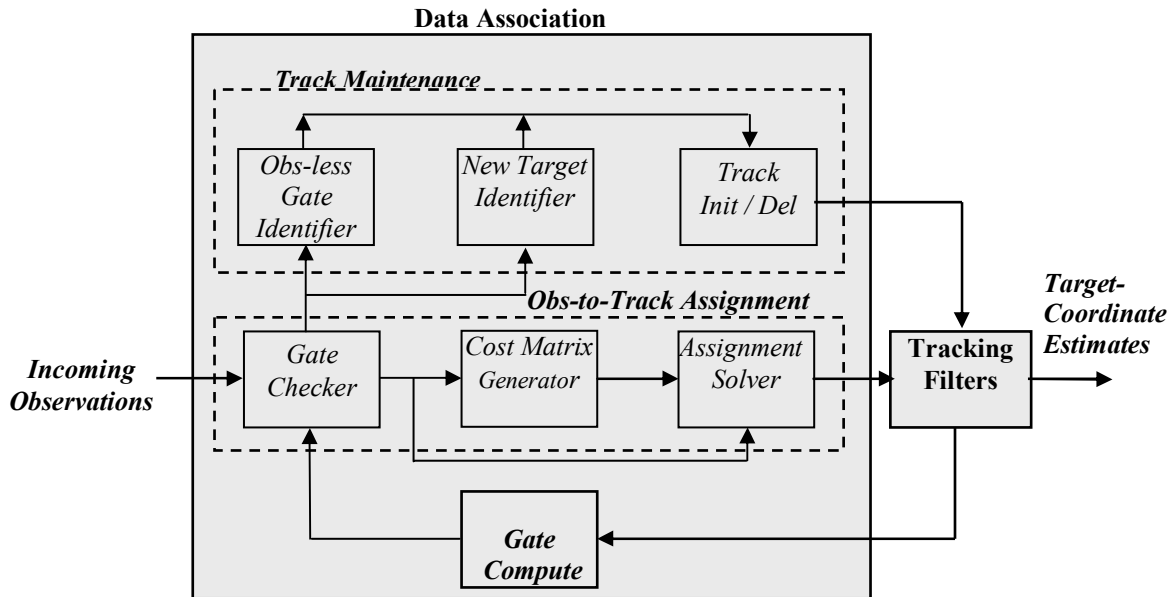


Figure 1: Elements of MTT

versa. In the second level of “*sorting out*”, namely observation-to-track assignment, a strictly one-to-one coupling is established between observations and tracks.

The “*Track Maintenance*” sub-function consists of three blocks. The “*obs-less Gate Identifier*” identifies the gate where no observation falls. This indicates a probable disappearance of an already known target and hence the deletion of its track after confirmation. The “*New Target Identifier*” detects observations that fall outside all the gates. These observations are potential candidates for initiating new tracks after confirmation. The “*Track Init/Del*” block initiates new tracks or deletes existing ones when needed.

Observations not assigned to existing tracks can initiate new tentative tracks. A tentative track becomes confirmed when the number and quality of the observations included in the track, satisfy confirmation criteria. Similarly, *low-quality* tracks, as usually determined by the *update history*, are deleted. *Track quality* may be defined as a criterion for initiating a new track or deleting an existing one. In context of this work, 3 observations out of 5 scans for the same target initiate a

measure the coordinates of a maximum of 20 targets. Hence this block will use 20 similar filters in final system.

3. Implementation

The application is distributed over different processors as defined distinct functions which exchange information among them. We are mapping separate defined functions/sub-functions to individual processors rather than running a single task on several processors in parallel. Similar considerations have been proposed in [10, 11, 12].

The two major parts of the MTT system are the Kalman filters and the Assignment algorithm. Kalman filters, as mentioned earlier above, is recursive algorithm looping around prediction and correction steps. Both these steps involve matrix operations on floating point quantities. This makes the filter a strong candidate to be mapped onto a separate processor. Thus for processing a maximum of 10 targets at a time, we

need 10 identical processors executing 10 Kalman filters.

The assignment-solver is a “step algorithm” consisting of six distinct iterative steps [9]. The algorithm goes through each of these steps several times before it comes out of the loop with the final observation-track pairings. Looping through the algorithm steps demand a long execution time hence this function cannot be combined with any of the other functions. So the assignment algorithm is another strong candidate to be mapped onto a separate processor

The “Gate Compute” block regularly passes information to “Gate Checker” which in turn, is in constant communication with “Cost Matrix

Generator”. We grouped these three blocks together and mapped them onto a single processor to minimize inter-processor communication which would have required extra buffers and would have added to the complexity of the system. Avoiding unnecessary inter-processor communication is also desirable for saving power (figure 2).

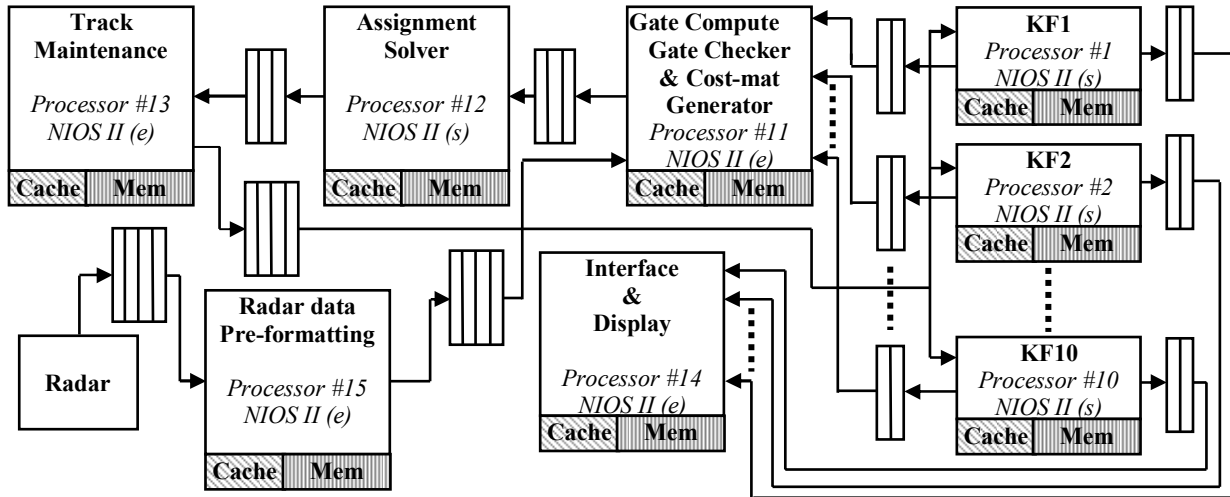


Figure 2 : The proposed architecture

The three blocks of the “Track Maintenance” sub-function individually don’t demand heavy computational resources, so we grouped them together for mapping onto a processor. We have two other processors in the system; one for radar data pre-formatting and the other for human interface and display functions. The first of these two processor works as an interface between the radar and the rest of the system. The second one presents the system output in a human readable form. Thus we use 15 processors for the complete application.

The three blocks of the “Track Maintenance” sub-function individually don’t demand heavy computational resources, so we grouped them together for mapping onto a processor. We have two other processors in the system; one for radar data pre-formatting and the other for human interface and display functions. The first of these two processor works as an interface between the radar and the rest of the system. The second one presents the system output in a human readable form. Thus we use 15 processors for the complete application.

4. Proposed architecture

Figure 2 shows a simplified view of our proposed architecture for the system. As a first implementation, a traditional SMP architecture with a large shared memory has

processors do not have any drastic effect on the overall system as long as their interfaces remain unchanged.

Using Altera’s Quartus II and SOPC design tools, we implemented the system with the NIOS II processors. The interconnections among various system components are accomplished through the Avalon switch fabric provided by Altera. The local memory and cache sizes as well as queue depths are specifically tailored up to the requirements of the individual functions running on different processors. Several experiments have been conducted to find the good balance between queue sizes. Nevertheless, generally the faster a function is, the deeper the queues on its processor output ports are. The processors running the filters and assignment algorithm are the standard NIOS II(s) processors while those running all the remaining functions are NIOS II light or economical (e) processors. These two types of processors differ in the number of logic elements they consume. NIOS II(s) also includes branch prediction, a separate instruction cache and the option to use hardware multiply and hardware divide.

The choice of using NIOS II(e) or NIOS II(s) is driven by the results of several tests done on the codes.

Inserting time stamps and performance counters into the code at various points, we came to know that the filter spent more than 90% of the time in multiply/divide operations. So the optional “hardware multiply” and “hardware divide” are included with the standard processors for the filters to augment their ALU’s. This augmentation is termed as “*custom instruction*” in NIOS II literature. From the software point of view, custom instructions appear as machine generated assembly macros or C functions [14]. Their inclusion is helpful in speeding up the filtering functions because the filters involve numerous multiply/divide operations on floating point numbers. Figure 4 shows the effects of custom instruction on run times for different processors. The application was coded in ANSI C and compiled with NIOS II IDE before downloading it into processor memories.

4. Discussion and results

In the very early stages of the work, we implemented a 2-state Kalman filter in RTL hardware using VHDL. It not only took enormous design efforts but also consumed 23,327 LUTs on the FPGA. In fact one 2-state Kalman filter needs about 48% of the Stratix II 2S60 FPGA resources. In our current design we are using ten 4-state filters apart from the other system components. Implementing a fully hardware system with ten 4-state filters would have been simply infeasible. The NIOS II (e) consumes 600 to 700 logic elements (LEs) whereas NIOS II (s) consumes 1200 to 1400 LEs. The whole system, for 10 targets, consumes almost 30,000 logic elements which

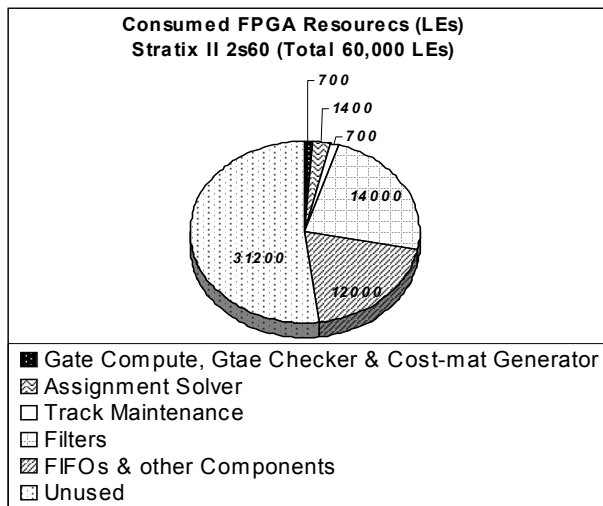


Figure 3: FPGA Resource consumption

are about 50% of the reconfigurable resources available on Stratix II 2S60 FPGA. So we can easily add other processors and components when needed, using the existing FPGA.

The radar sensor we are using has a Pulse Repetition Time (PRT) of 20 ms. This requires us to go round the “data association-filtering” loop for all the 10 targets in less than 20 ms.

Using 100 MHz clock, the slowest function in the application i.e. the filter takes 15 ms to complete the prediction-correction loop discussed earlier. This is well below the 20 ms threshold set by the radar PRT. Since all the filters run concurrently on separate processors, this run time remains unchanged whatever the number of targets and filters. Our initial experiments show that the execution time of the assignment algorithm increases exponentially with the number of targets. Using a single a processor for executing the assignment algorithm would not be advisable if we were to scale the system up for higher number of targets. This is because for more than 15 targets the execution time for the assignment algorithm would overshoot the radar PRT cut-off. But the good news is that this algorithm is highly modular and can be distributed over several processors to bring its execution time down. This is what we intend to cater for higher number of targets.

5. Related work

To our understanding, comprehensive literature about implementation of complete MTT system in FPGA does not exist. Some work has been done on different isolated components of the MTT system. For example in [2] the authors propose an FPGA implementation of the estimation part of MTT system. Apart from being limited only to estimation, this is a fully hardware implementation. As mentioned before, fully hardware designs lack the flexibility and programmability needed for the ever evolving modern day applications. Moreover, the authors don’t explicitly say whether floating point computations are taken into account in their solution. A fully hardware implementation of the Kalman filter only, is also proposed in [3]. This implementation also has all the above mentioned limitations for the same reasons. An attempt to implement an MTT system in hardware for maritime application is documented in [7]. In addition to being a completely hardware implementation, the work presented here is inconclusive. The data association aspect of MTT has been dealt with nicely in [8] but the physical implementation of the system is not a consideration here. Moreover the use of Multiple Target Tracking with radar for road safety is rarely dealt with. The bulk of the MTT work is carried out in context of military and civilian air traffic control applications.

Our work is unique in several aspects. In contrast to the works mentioned above, we consider a complete MTT system implementation. Our reconfigurable MPSoC architecture of the system is inherently flexible,

programmable and scalable. Thus it can evolve very easily with advances in technology and improvement in application algorithms. The use of several concurrently running processors meets the overall real time deadlines. Several low frequency processors running concurrently consume less power compared to a single processor with a high frequency [13]. The reconfigurability of the processors and other components allow for customizing them according to application and further economizing FPGA resources. The system we propose is a complete plug-and-play solution that can easily be integrated with the existing electronic systems onboard a vehicle.

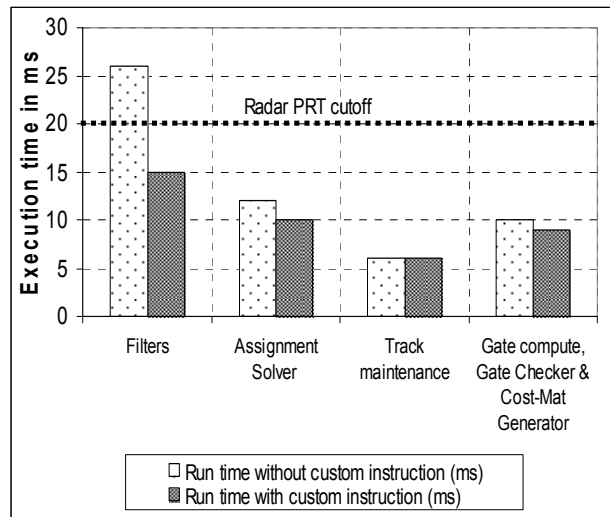


Figure 4: Run time & custom instructions

6. Future work

The system has tremendous potential for growth on several fronts. On the application side, we intend to take dynamically varying number of targets into account. This would require us to dynamically vary the number of filter configurations on the FPGA. Tracking precision can be improved through better / newer algorithms e.g. extended Kalman Filter (EKF), auction algorithm for assignment, radar-signature aided data association etc.

On the architecture side, new processor configurations and interconnections shall be evaluated for run time and power consumption improvement. With the continuous improvement in sensor development, it is expected that the radar PRT will drop down in the near future. This will require us to further decrease the execution time of the application whereas power saving is always a sought after feature of embedded systems. We plan to carry out power consumption analysis on the system in the next phase of the work. To integrate the system with other electronic safety systems on board a vehicle, we shall add a CAN (Controller Area Network) interface to the system.

7. Conclusions

In this paper we presented an application-specific MPSoC architecture for MTT based driver assistant system. We implemented the system in FPGA with Altera's SOPC builder and NIOS II processors. We meet the real time deadlines of the application. The system uses 50% of the reconfigurable resources on a Stratix II 2s60 FPGA which contains 60,000 logic elements. The system currently caters for a maximum of 10 targets. In the next stage of the work the number of targets being processed will be increased up to 20. The system is easily evolvable and scalable. It is not only a complete embedded MTT solution but is also economical and power efficient. We plan to take the system further up the evolution hierarchy in the near future.

References

- [1]. Samuel Blackman and Robert Popoli, "Design and analysis of modern tracking systems", Artech House Publishers 1999, ISBN 1-58053-006-0.
- [2]. Zoran Salcic and Chung-Ruey Lee, "FPGA-Based Adaptive Tracking Estimation Computer" IEEE transactions on aerospace and electronic systems, 2001.
- [3]. Salcic, Z. et al. "Scalar-based direct algorithm mapping FPLD implementation of Kalman filter." IEEE Transactions on Aerospace and Electronic Systems, 2000.
- [4]. Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems", Transaction of the ASME--Journal of Basic Engineering, pp.35-45(Mar 1960).
- [5]. Welch, G and Bishop, G. 2001. "An introduction to the Kalman Filter", <http://www.cs.unc.edu/~welch/kalman/>
- [6]. Brookner, E. "Tracking and Kalman filtering made easy" John Wiley & Sons Inc. 1998.
- [7]. Boismenu, Y. "Etude d'une carte de tracking radar---" Thèse de doctorat 2000, Université de Bourgogne, France.
- [8]. P. Konstantinova et al. "A study of Target Tracking Algorithm Using Global Nearest Neighbor Approach" CompSysTech' 2003.
- [9]. Munkres' Assignment Algorithm, Modified for Rectangular Matrices <http://cslab.murraystate.edu/bob.pilgrim/445/munkres.html>
- [10]. Joost, R.; Salomon, R. "Advantages of FPGA-based multiprocessor systems in industrial applications" Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conf. of IEEE Volume, Issue, 6-10 Nov. 2005.
- [11]. Epson's Breakthrough REALOID Printer SOC Powered by Multiple Xtensa Processors http://www.tensilica.com/products/lits_success-stories_epson.htm
- [12]. Hannu Penttinen, Tapio Koskinen, Marko Hännikäinen."Leon3 MP on Altera FPGA" Altera Innovate Nordic 2007, Final Project Report 8/28/2007
- [13]. Frank Schirmermeister Imperas, Inc." Multi-core Processors: Fundamentals, Trends, and Challenges" Embedded Systems Conference 2007 ESC351.
- [14]. Altera Corporation. "NIOS II processor reference handbook". www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf