Design and Implementation of Real-Time EP80579 Based Embedded System

JAREER H. ABDEL-QADER
Department of Computer Science and Engineering
University of Texas at Arlington
416 Yates St. Nedderman Hall, Rm 300, Arlington Texas
USA
+1-817-272-3640
Jareer.abdelqader@mavs.uta.edu

ROGER S. WALKER
Department of Computer Science and Engineering
University of Texas at Arlington
416 Yates St. Nedderman Hall, Rm 300, Arlington Texas
USA
+1-817-272-3640
Roger.walker@uta.edu
http://ranger.uta.edu/~walker/

Abstract: With the advances in integration of different units such as I/O controllers and network interfaces in a single chip, Intel introduced the low power EP80579 embedded processor. This processor is the first IA based system-on-chip (SoC) with an IA-32 processor core, North and South Bridges, and integrated Accelerator and network interface.

In this paper we will show main steps to design a real-time embedded system along with identifying the hardware and software requirements to implement such system. The embedded system introduced in this work will perform several tasks regarding road surface conditions based on multiple sensor readings. The sensor data will be processed in real-time to reconstruct the road profile and provide an estimate for the texture contents of the road surface. The EP80579 SoC will be used in the design of such a system. Modeling will be done with the aid of UML profile for modeling and analysis of real-time and embedded (MARTE) systems.

Key-words: System-on-a-chip; Modeling Real-time systems; parallel processing modeling; MARTE; Embedded Application; Road Profiler.

1 Introduction

Road profiler [1] is a system that is used to reconstruct the road profile from the data collected by laser and accelerometer sensors. With the aid of the latest SoC from Intel, the Tolapai embedded processor, we intend to design and implement a real-time road profiler system with capabilities of providing both road profile and road surface texture analysis. The final desired system to be implemented will be a Real-time road profiler/ texture measurement system.

The first step in the design of a new system is to provide a model for the system that shows the main parts of the system and how to integrate them together to reach the desired design. For this reason modeling tools are used. The general-purpose modeling language UML (Unified Modeling Language) is a standard used mainly for software systems. UML provides extensions and profiles that can help modeling embedded systems. Many of these profiles such as UML-RT, SysML, SPT and MARTE [2-5] are now standardized and in use for modeling embedded systems and the real-time operations. Also the new UML profiles introduce the tools to model both hardware and software systems and provide a way of allocating the software to the desired hardware unit. Modeling real-time systems either as hardware, software, or a combination of both has been the subject of several research projects. For instance, [10] models a network constructed from network-on-chip (NoC) systems with the aid of MARTE. In [11] a hardware model is introduced for the IP-XACT feature which is a standard to normalize interface of intellectual property (IP) from different vendors of SoC systems. [12] Models a

telecommunication system, GSM base transceiver station, using UML standard.

The following section discusses a number of UML profiles and extensions developed for embedded systems modeling. The third section of the paper describes briefly the road profiler/ texture system. Section 4 introduces the Tolapai (EP80579) SoC. Section 5 briefly introduces the software used for implementing such application. The modeling steps as well as the results obtained from the road profiler system are shown and analyzed in section 6. Finally, the paper is concluded in section 7.

2 UML Extensions for Modeling Real-Time Systems

In this section we are discussing some of the modeling profiles adopted and standardized as part of UML.

2.1 UML-RT

Real-time Object Oriented Modeling (ROOM) [5] is a modeling language used for modeling real-time systems. It has its own graphical notation set to model structures of real-time systems. A capsule stereotype was introduced by ROOM to represent a reactive object. A capsule can communicate with other capsules through ports, which are boundary objects, and a protocol associated with the port. ROOM also defines a connector which connects ports to provide transmission facility for supporting a particular protocol. ROOM is more oriented towards the actual implementation and physical design [8]. ROOM was integrated as part of UML to form what is known as UML-RT. Lack of usage and support is considered one of the limitations of the UML-RT.

2.2 SysML

The Systems Modeling Language (SysML) is a UML profile that is domain-specific Modeling language for systems engineering. SysML supports the specification, analysis, design, verification and validation of a broad range of complex systems [3]. SysML defines two types of diagrams, the Block Definition Diagram (BDD) and the Internal Block Diagram (IBD). The BDD is based on UML Class Diagrams and UML Composite Structure Diagrams. The role of a BDD is to describe the relationships among blocks, which are basic structural elements focusing on specifying hierarchies and interconnections of the system to be modeled. The SysML IBD allows the designer to refine the structural aspect of the model. The IBD is the equivalent of the composite structure in UML. SysML lacks the constructs for modeling time.

2.3 UML Profile for Scheduling, Performance and Time (SPT)

SPT defines a set of concepts useful for modeling real-time systems. Its purpose is to integrate notation used by existing real-time analysis techniques into UML in order to:

- Enable the construction of models that could be used to make quantitative predictions regarding these characteristics.
- Facilitate communication of design intent between developers in a standard way.
- Enable inter-operability between various analysis and design tools.

Thus, the SPT is defined to offer a common framework for real-time modeling that unifies the diversity of techniques, terminologies and notations existing in the real-time software community, while still leaving space for different kinds of specifications. In its present form, the main focus of SPT is on time and time-related concepts: performance, timelines, schedulability, etc.

SPT offers a terminology for modeling real-time systems: it defines a set of concepts - aiming to fit standard real-time modeling techniques - and some relationships between these concepts as allowed by the meta-modeling technique used for the definition of the SPT.

The use of SPT is justified because UML is lacking in some key areas that are of particular concern to real-time system designers and developers. In particular, the lack of a quantifiable notion of time and resources was an impediment to its broader use in the real-time and embedded domain. It was discovered that UML had all the requisite mechanisms for addressing these issues, in particular through its extensibility faculties [14, 15]. SPT is a standard way of using these capabilities to represent concepts and practices from the real-time domain.

One of the main guiding principles is that, as much as possible, modelers should not be hindered in the way they use UML to represent their systems just for the purpose of model analysis. That is, rather than enforcing a specific approach or modeling style for real-time systems, the profile should allow modelers to choose the style and modeling constructs that they feel are the best fit for their needs of the moment.

2.4 UML Profile for MARTE

MARTE is UML profile adopted by OMG in order to extend the capacities of UML for real-time modeling in embedded systems. Not only for the modeling and analysis, MARTE also provides support for specification, design, and verification/ validation stages. This new profile is intended to replace the existing UML Profile for Schedulability, Performance and Time. [4]

Because SPT's constructs were considered too abstract and hard to apply, and for the requirement of aligning SPT profile UML2.0, there was a need for upgrading or creating new profile.

MARTE profile is an evolution of the SPT profile with the purpose of upgrading this profile to UML2.0. It is made of

various packages: namely MARTE foundations, MARTE design model, MARTE analysis model and MARTE annexes. The profile is intended to be a fundamental tool in the design of real time systems. Both modeling and analyzing concerns are tackled leading to a complete instrument to improve the design phase. Within MARTE, the Software Resource modeling (SRM) framework provides modeling artifacts to describe software execution platform modeling. The SRM profile provides a broad range of modeling capabilities covering main multitasking framework such as dedicated real-time language libraries and real-time operating systems.

Besides software resources, MARTE allows us to model hardware resources. Due to its general purpose, UML lacks certain key native artifacts for describing concrete and precise hardware RTE execution platform. The UML profile for MARTE fills this lack with two sub-profiles: a generic resource modeling (GRM) profile and a hardware resource modeling profile (HRM). Both can be used to model hardware platform.

The HRM is composed of two views, a logical view that classifies hardware resources depending on their functional properties, and a physical view that concentrates on their physical nature. Both are specializations of the general model. The logical and physical views are complementary. They provide two different abstractions of hardware which could be simply merged.

2.5 Real-time Modeling Using UML and UML Profiles

Common UML and/ or UML profiles modeling practices suggests that

- Use of UML profiles only when standard UML cannot perform the task.
- Use SysML diagrams for general modeling of the system.
- Use MARTE for modeling the system's details.
- Use MARTE specifically to model the hardware system with all of its details along with modeling the software methods which are part of the application.

3 Road Profiler/ Texture System

The system to be designed will collect and analyze data regarding road surface condition in order to specify road roughness and usability. The purpose of the system is to determine the road profile and texture measurements from the data collected by laser and accelerometer sensors installed in a vehicle which is driven over a specified road section.

3.1 Profiler

The profiler is an instrument that is used to produce values related in a well-defined way to a road surface [1]. Profiler combines reference elevation, height relative to the reference and longitudinal distance to produce the true road surface profile.

Most profilers measure profiles for wheel paths traveled. For each wheel path an accelerometer is used to find inertial reference defining the height of the accelerometer at that moment after double integrating the acceleration measurements. A laser sensor is then used to obtain readings representing the height of the road surface to the reference, and a distance encoder provides the longitudinal distance as in Fig. 1.

The road profile is reconstructed from laser and accelerometer readings according to (1).

$$p(t) = \iint a(t)dtdt - H(t) \qquad (1)$$

Where

a(t) is the acceleration,

H(t) is the height measured by the laser sensor.

A high-pass filter is used to remove the effect of long wavelengths on the profile. These wavelengths represent the underlying grade and overall road curvature and are more difficult to measure with inertial profilers with the current configuration. There is also a distance sensor to measure the distance traveled. An optical sensor is often used to determine the start and end of some measured road sections.

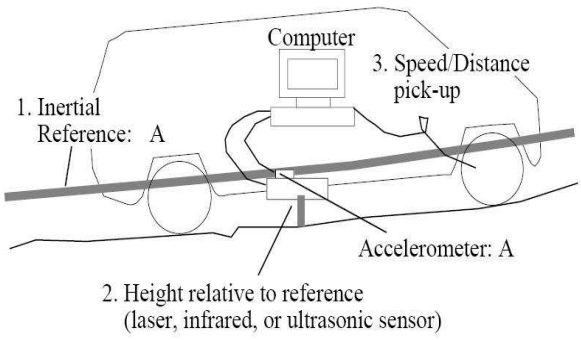

Fig. 1 Road profiler system [1]

3.2 Texture Analysis

ASTM E 1845 [6] standardizes the calculation of pavement texture from laser readings which represent the measured profile of the pavement macro-texture.

The pavement macro-texture is defined as the deviations of a pavement surface from a true planar surface with the characteristic dimensions of wavelength and amplitude from 5 mm and up.

In order to compute the mean profile depth (MPD), the measured profile is divided into segments each having a base-length of 100 mm. The slope, if any, of each segment is suppressed by subtracting a linear regression of the segment. The segment is further divided into two equal lengths of 50 mm. segments and the height of the highest peak in each half segment is determined. The difference between that height and the average level of the segment is calculated. The average value of these differences for all segments making up the measured profile is reported as the MPD. Check the illustration in Fig. 2.

The texture will be computed from laser sensor readings for both left and right wheel paths from the road profiler system as discussed in the previous subsection.

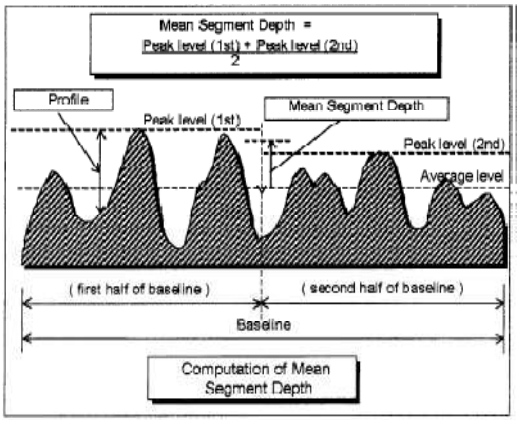

Fig. 2 Mean segment depth computation

3.3 Real-Time Implementation Requirements

In this section to the system requirements and the amount of data to be processed will be discussed.

The main requirement to operate the road profiler/ texture system is to collect data for any road section with the minimal distraction to the traffic, which means that the profiler vehicle should run in a speed range of 40- 60 mph (58- 88 ft/sec). The data from all 6 sensors are to be collected simultaneously via data acquisition system in order to compute the road profile values correctly. The minimum sampling rate required to construct road profile from sensor readings is 4 kHz; while for accurate texture estimation, laser readings should be sampled with at least 24 kHz. So, in order to implement the road profiler/ texture system sampling rate required is 24 kHz or more, since all sensor readings should be sampled with the same speed for consistency.

4 EP80579 Embedded Processor

EP80579 (code name Tolapai) [13] is a system-on-a-chip (SoC) embedded processor which includes an Intel architecture complex based on the Intel Pentium M processor, integrated memory controller hub, integrated I/O controller hub and flexible integrated I/O support with three Ethernet connections, two Controller Area Network (CAN) interfaces and a local expansion bus interface. The design also includes PCI Express, High Speed Serial1 (HSS) ports for TDM or analog voice connectivity, security accelerators for bulk encryption, hashing and public/private key generation.

The Intel QuickAssist Technology initiative consists of a family of interrelated Intel and industry standard technologies that simplify the use and deployment of accelerators on Intel platforms. The integrated accelerators in this processor support Intel QuickAssist Technology through software packages provided by Intel. These software packages provide the library structures to integrate security and/or VoIP functionality into the application, completely adjunct to the Intel architecture complex, freeing up CPU cycles to support additional features and capabilities. This provides the efficiency of customized hardware with the flexibility to design diverse applications with one platform.

Fig. 3 shows a block diagram of the Tolapai SoC.

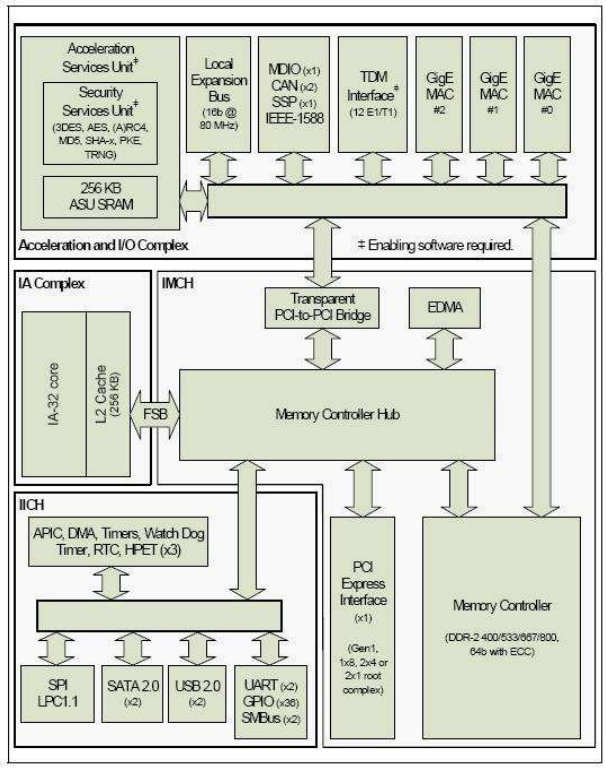

Fig. 3 Block Diagram of the Tolapai Embedded Processor [7]

5 Embedded Software

Parallel processing implementation requires an operating system capable of dealing and managing parallel tasks (threads), and a programming language with ability to

create, run, and terminate threads. Recently there are many operating systems that can support parallel processing. Selecting an operating system is one of the most fundamental decisions as a device manufacturer you must make. In the case of choosing an operating system for embedded systems, such operating system should have certain characteristics, like having a small foot print since usually embedded systems do not have large storage devices. Also the operating system should be able to communicate and control any special peripheral designed specific embedded applications. Since embedded systems are mainly designed with time constraints or what is known as real-time application, the operating system must be able to meet the real-time requirements for the implemented application or the whole system will be useless.

An operating system dedicated for embedded systems is known as a real-time operating system (RTOS). RTOS is an operating system that guarantees a certain capability within a specified time constraint. Some real-time operating systems are created for a special application and others are more general purpose. For our application we selected one the Windows based operating systems dedicated to the embedded systems. The Windows Embedded Standard.

5.1 Windows Embedded Standard

Windows Embedded Standard allows developers to get access to embedded specific tools that work in the familiar developer environment of Visual Studio allowing them to rapidly conFig., build, and deploy devices that are more secure, reliable, and manageable.

The main practice of parallel processing involves splitting a single problem into separate parts that are solved simultaneously, allowing more work to be done faster due to the fact of using parallel execution instead of serial execution. Those parts are implemented in program units called threads, and using several threads as in the case of parallel processing is known as multi-threading.

Multi-threading is implemented within a single program, running on a single system. It requires a multi-thread capable operating system, which allows a program to split tasks between multiple threads of execution. On a machine with multiple processors, these threads can execute concurrently, potentially speeding up a given task significantly. Also multi-threading requires programming languages that provide abstractions for expressing the parallelism explicitly.

5.2 Parallel Programming Models

In order to split a single problem into separate parts, or create a parallel programming model from a serial programming model, developers should identify the parts (workloads) and the dependencies among them in the given program; then divide the workload to multiple threads. This division is known as decomposition. There are three major forms of decomposition [18, 19]; the first type is the task (Functional) decomposition where the division is based on the type of work assigned to different thread. The second decomposition type is the data decomposition which is based on dividing the data among a number of threads that perform the same functions but processing different data block. And finally, the third one is the data flow decomposition where the output of one thread is the input of another one. This decomposition type is used when there is dependency among threads which occur in what is known as a producer/ consumer problem which is the case in pipeline or wave-front programming patterns.

5.3 Thread Implementation

Programming embedded multiprocessor systems can be difficult. They are generally programmed at a relatively low level using C, C++, or even assembly language.

Thread implementation requires special support from the operating system and the implementation language with the support of special libraries that helps creating and manipulation threads.

OpenMP is one of the well known and widely used threading technologies. OpenMP is an application programming interface (API) for parallel programming which consists of compiler directives and library of support functions that is conFig.d to work with FORTRAN, C, or C++ programming languages. OpenMP supports both data and functional Parallelism on a shared memory system.

OpenMP provides support for concurrency, synchronization, and data handling while hiding the need for explicit thread management.

6 Modeling Road Profiler with MARTE

In this section we will show and discuss the modeling process used to model the real-time road profiler/ texture system based on UML profile for MARTE. Also we will show the results from implementing such a system using an EP80579 based embedded board.

6.1 Road Profiler/ Texture System Modeling

This subsection focuses on how to model real-time embedded systems, sensor hardware, and multi-core processors using real-time UML extensions, as a first step in the designing and building a fully integrated real-time system that implements profiler/ texture system. We used a combination of UML and MARTE to model the system, in which we use MARTE only when UML has limited support. The first step in modeling with UML is to provide a description of a system's behavior which is done with the aid of a use case diagram. Fig. 4 illustrates the use case that describes the profiler/ texture system. The Fig. shows two actors. The road surface itself under test, and the system's operator that is responsible for driving and running the measurement vehicle. This use case is described by a series of events that occur regarding operating the profiler/ texture

system. The road surface will be measured to determine the reference elevation from accelerometer sensor and height relative to the reference from laser sensor of the road surface. A combination both data sets will be used to compute profile, while the texture will be estimated using the height readings alone.

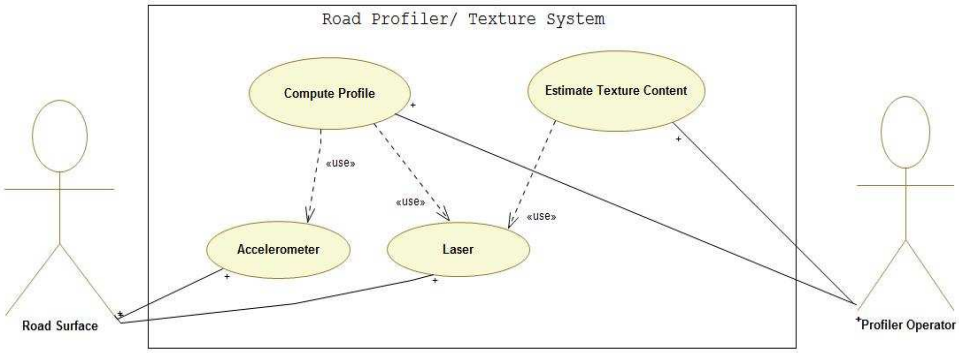

Fig. 4 Road Profiler Use Case

The activity diagram is used to show workflows (flowchart) in a step by step manner for the activities and actions, with support for choice, iteration and concurrency. State (activity) diagram is another diagram of the UML standard that shows the step-by-step workflows of the activities and actions. The activity diagram of the system here is shown in Fig. 5. The main activity this system carries is to continuously collect sensor data then distribute the readings between two computational tasks processed concurrently, profile construction and texture estimation. The concurrent operations are represented in the activity diagram with the aid of fork, join states. The fork pseudo-state is a connector that branches a single input transition into multiple outgoing transitions to different states that will be activated concurrently. The join pseudo-state joins together multiple incoming transitions into a single transition. Once the data collection is over, and the system is ready to stop working, all data points will be saved in output files for offline analysis and archiving processes.

Next, the state machine is introduced (Fig. 6). There are two states. The idle/ pre-section state is where the system starts running but doesn't perform any computation and the real section state where the system collects and processes the data. UML models parallelism in two ways. First, all objects are considered to be parallel entities. Second, a single object entity exhibits itself a concurrent behavior. This means that the object's state-machine is specified as a set of concurrent components. The real section state is supposed to perform three operations separated by dotted lines. The first operation is to collect the sensor readings, while the second operation is to perform texture analysis once enough data is obtained (4 inch worth of data); the third and final operation is to compute profile for every 1 inch. Those three operations are intended to be performed concurrently as stated in the state machine.

In order to model the Tolapai embedded processor we created new stereotypes. One stereotype is for the QuickAssist technology and the other is for the Tolapai processor itself adding the capability of customizing them towards individual application domains. The new stereotypes will appear in diagrams as <<QuickAssist>> and <<Tolapai>> respectively. Fig. 7 shows the <<Tolapai>> stereotype, which extends three of the MARTE meta-classes namely hw_processor to represent the IA-32 processor in the Tolapai, the hwI_O for the I/O controller part, the hwMMU for the memory management unit. It also extends the <<QuickAssist>> stereotype.

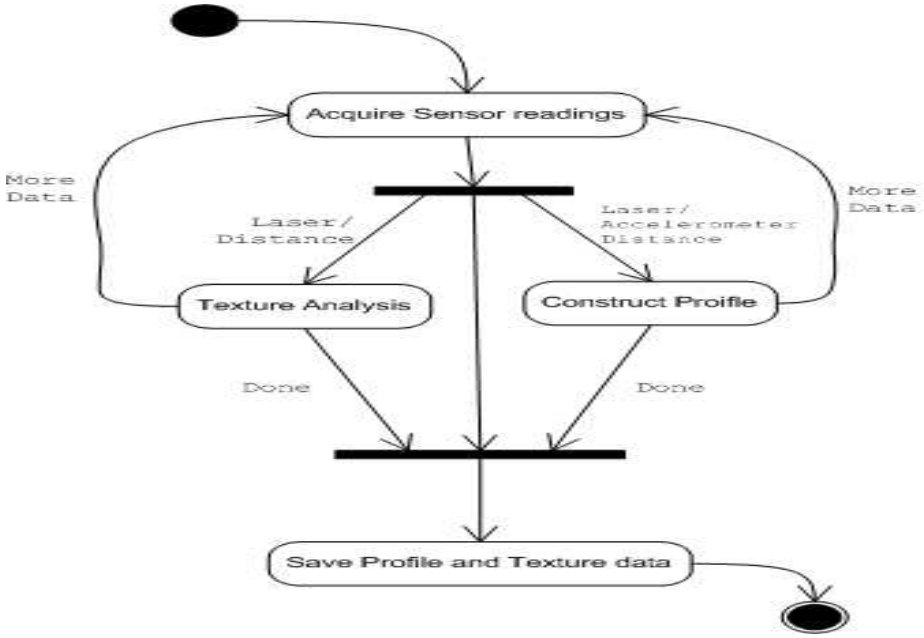

Fig. 5 Activity Diagram

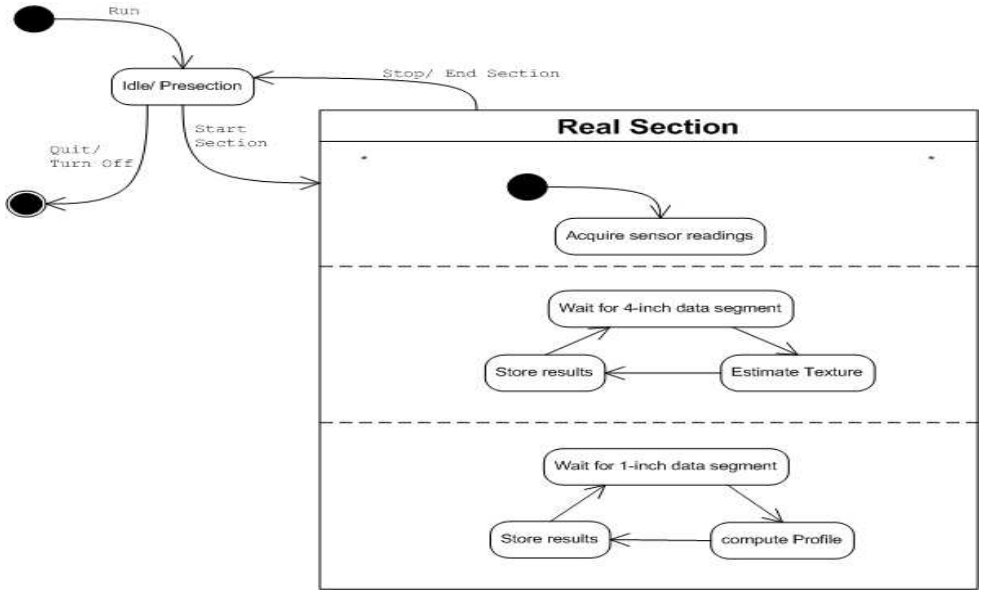

Fig. 6 State-Machine Diagram

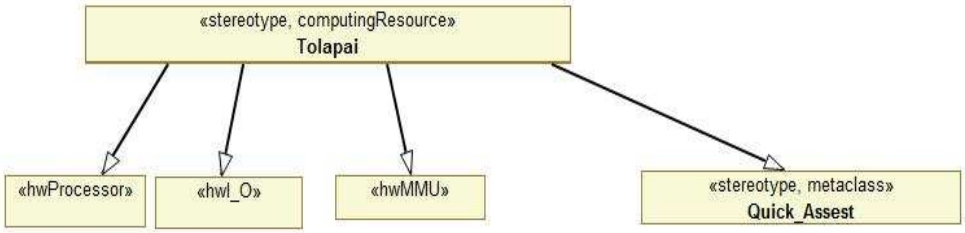

Fig. 7 Tolapai Stereotype

Fig. 8 illustrates the HRM model of the road profile/ texture system. Using the stereotypes defined by MARTE for different hardware components plus the newly created stereotype for the Tolapai processor we were able to model and specify the hardware devices and the controllers important to build such as system. On the other hand, the software functions and methods to be used are modeled with the aid of SRM profile as in Fig. 9.

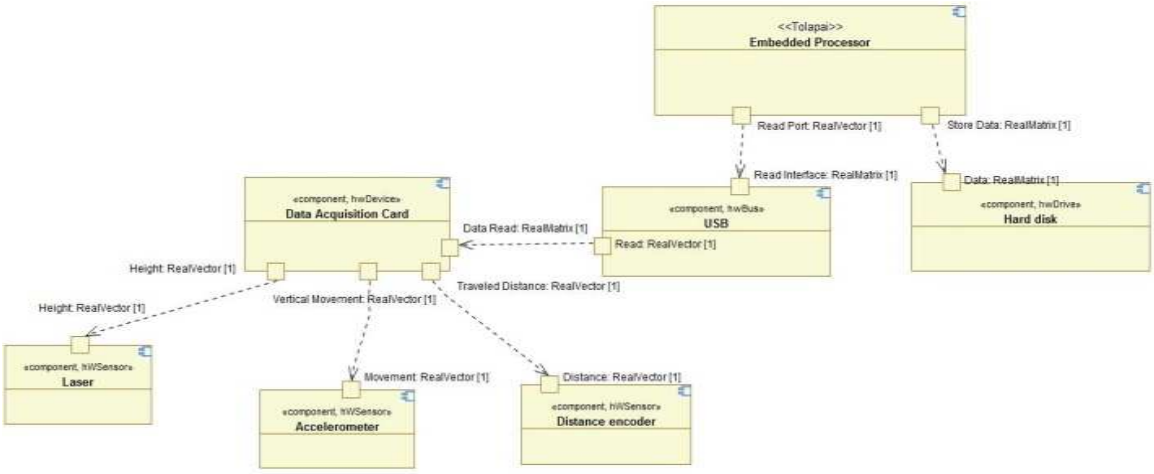

Fig. 8 Road Profiler HRM Model

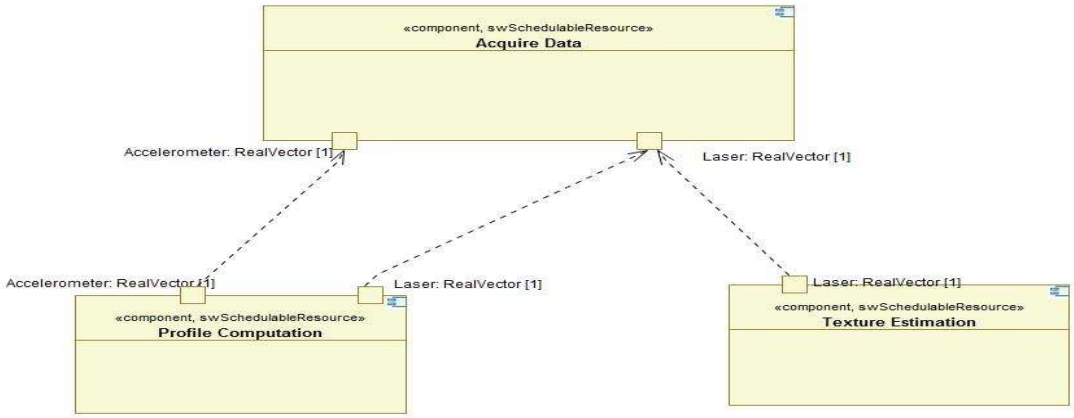

Fig. 9 Road Profiler SRM Model

6.2 Applicability of Tolapai for the Road Profiler/ Texture System

The profiler program was next used with the Tolapai development system to simulate real-time data collection. A portable instrument package used for obtaining raw sensor data for a given wheel path, Fig. 10, was used to obtain sensor data from a typical road section. The desired plan is to use the Tolapai inside the instrument module, processing and sending the computed profile, texture and other pavement performance characteristics via a network connection to a client computer. Using this data the multi-threaded profiler program was run on the Tolapai development system to compute profile for one wheel path, simulating the real-time data measurement process. The computed results matched with the real-time measurements using the current measurement system. Fig. 11 represents the road profile while Fig. 12 shows the MPD values for texture estimate.

In terms of software implementation, the application is programmed in C with the aid of OpenMP as a threading technology. The threads created according to the thread mapping obtained from decomposing the given application as illustrated in the activity and state-machine diagrams (Fig. 5 and Fig. 6) and discussed in the previous subsection. The tasks to be processed are mapped according to the dependency and the amount of work to be done in each stage. The road profiler/ texture system problem fits the data flow decomposition and more precisely, the producer/ consumer model. From Fig. 3, there are three tasks that can be processed in parallel; the main task is data acquisition. The second task is the profile construction, and the third one is texture analysis. Both second and third task depends on the first task and the data passed to them from the first task, in which the relationship among them fits the producer/

consumer model where the data acquisition represents the producer and both the texture analysis and profile construction are the consumer part in this relationship. This way of decomposition requires creating three different threads each of them is assigned to one of the tasks. Fig. 13 lists the main part of the code that starts with setting the number of threads to be used (3 threads) then the program will call the `ADC()` function to initialize the data acquisition unit and any other device that needs initialization) . After that the program will initialize and run the created threads with the aid of section pragmas in OpenMP were each one of them will execute one task assigned by the function calls in each section.

Fig. 10 Portable Profiler Instrument Module

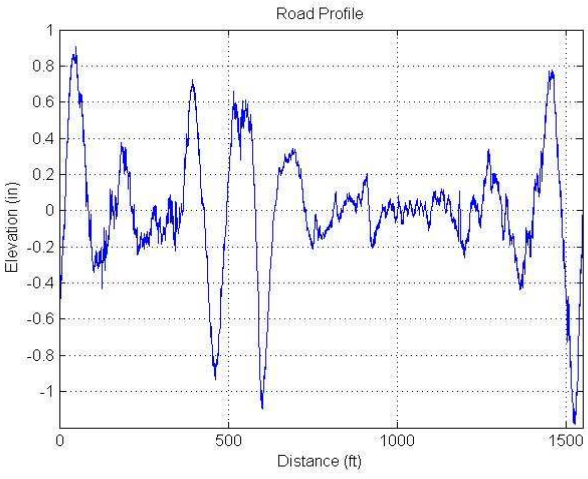

Fig. 11 Road Profile measurement for tested section

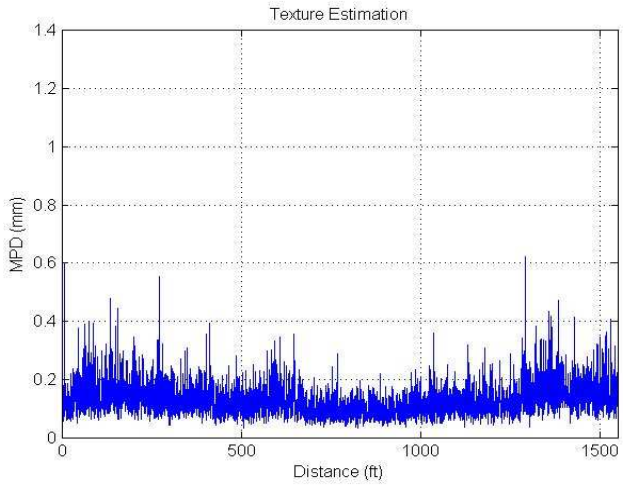

Fig. 12 MPD Values for Estimating Texture Contents of the Tested Section

```
#include "stdio.h"
#include "math.h"
#include <time.h>
#include <omp.h>
...
void Set_ADC(); //Initialize the ADC
Device
void Data_Acquisition();
void Construct_Profile();
void Texture_Analysis();
int main()
{
omp_set_num_threads(3);
Set_ADC();
#pragma omp parallel sections
{
 #pragma omp section
  Data_Acquisition();
#pragma omp section
  Texture_Analysis();
 #pragma omp section
  Construct_Profile();
 }
 printf("Program End...\n");
}
```

Fig. 13 C Code for Implementing the Road Profiler/ Texture System in with OpenMP Support

7 Conclusion

In this paper we discussed how to design and model a real-time embedded system that performs several tasks regarding road surface conditions based on multiple sensor readings which will be processed to reconstruct the road profile and to provide an estimate for the texture contents of the road surface. The Tolapai embedded processor will be used in the design of such a system. A comparison between different UML modeling profiles was introduced and accordingly we decided on a combination of UML and MARTE to model the system. We use MARTE only when UML has limited support. UML was used to show the general model of the system and MARTE was used specifically to model the hardware system with all of its details along with modeling the software methods which are part of the application.

ACKNOWLEDGMENT

This research has been funded in part by an Intel funded research project on" investigating the use of an EP80579 platform for real-time monitoring of pavement surface and bridge structure characteristics in a mobile measurement system" and a project with the Texas department of transportation and federal highway administration on development of a portable profiling system.

References:

[1] M. W. Sayers and S. M. Karamihas, *the Little Book of Profiling*, University of Michigan Transportation Research Institute, September 1998.

[2] SysML, http://www.omgsysml.org/

[3] MARTE, http://www.omgmarte.org/

[4] http://www.omg.org/technology/documents/formal/schedulability.htm

[5] B. Selic, J. Rumbaugh, *"Using UML for modeling complex real-time systems"*, ObjecTime, March 1998

[6] ASTM E1845-01,"*Standard practice for Calculating Pavement Macrotexture Mean Profile Depth*", 2005

[7] http://www.intel.com/design/intarch/ep80579/index.htm

[8] A. Staines, *"A Comparison of Software Analysis and Design Methods for Real Time Systems"*, PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 21 MAY 2007

[9] J. Kebrle, R. Walker, "*Texture Measurement and Friction Estimation Using Laser Data Acquisition and Neural Networks*", Proceedings of Mathematical and Computational Methods in Science and Engineering, WSEAS, Trinidad and Tobago, November 5-7, 2007.

[10] I. Quadri, P. Boulet, S. Meftali, J. Dekeyser, *"Using an MDE Approach for Modeling of Interconnection Networks"*, Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks 2008, Sydney, May 2008

[11] C. André, F. Mallet, A. Khan, R. de Simone *"Modeling SPIRIT IP-XACT with UML MARTE"*, Proc. DATE Workshop on Modeling and Analysis of Real-Time and Embedded Systems with the MARTE UML profile, 2008

[12] R. Jigorea, S. Manolache, P. Eles, Z. Peng, *"Modeling of Real-Time Embedded Systems in an Object-Oriented Design Environment with UML"*, 3rd IEEE Int. Symposium on Object-Oriented Real-Time, Distributed Computing, Newport Beach, CA, March 2000.

[13] http://edc.intel.com/Platforms/EP80579/#platform-overview-content=platform-overview-toggle~~visible-content

[14] S. Turki, T. S. Lismma, A. Sghaier, *"A SysML profile for mechatronics integrating Bond Graphs"*, 9th WSEAS International Conference on Systems, Athens, Greece, July 11-13, 2005.

[15] M. E. Cambronero Gregorio D´ıaz J.Jos´e Pardo Valent´ın Valero Fernando L. Pelayo, *"RT-UML for modeling Real-Time Web Services"*, Proceedings of the IEEE Services Computing Workshops (SCW'06)

[16] K. Ito, S. Matsuura, *"Model Driven Development for Embedded Systems"*, Proceedings of the 9th WSEAS international conference on Software engineering, parallel and distributed systems, Cambridge, UK, Feb20-22, 2010

[17] A. R. Mahajan , M. S. Ali, *"Optimization of memory system in Real-Time Embedded Systems"*, Proceedings of the 11th WSEAS International Conference on Computers, Agios Nikolaos, Crete Island, Greece, July 26-28, 2007

[18] Multi-Core Programming Increasing Performance through Software Multi-threading, Shameem Akhter, Jason Roberts, Intel Press, 2006

[19] Software Development for Embedded Multi-core Systems: A Practical Guide Using Embedded Intel Architecture, Max Domeika, Newnes, 2008