IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

Model Driven Embedded System Testing Using Real-Time Data

Pravin Y. Karmore[1], Pradeep K. Butey[2]

Research Scholar, Dept. of Electronics & Computer Science, RTM Nagpur University, Nagpur, India [1]

Associate Professor, Department of Computer Science, Kamla Nehru College, Nagpur, India [2]

Abstract: Model driven development and testing of an embedded system is a great solution over the unnecessary wasting of time and money in the physical development and testing of embedded systems. This paper mainly focuses on the model driven development of an embedded system using the Simulink system development tool and testing of the modelled system to find out the better system using real-time data. The testing of embedded system before the actual development of hardware, software and testing using modelling approach is a very good advantage of the system developers. Today's embedded system development and testing becomes very complex due to increase in customer demands from household appliances to industry equipment.

Keywords: Embedded system testing, model driven development, image processing, real-time data.

I. INTRODUCTION

Testing is executed by conducting a program developed with test inputs and comparing the observed output with the expected one. Testing is a static and dynamic activity. It includes unit, interface, usability, functionality, performance, operability and security testing of software. Model Driven Testing (MDT) [1] [2] is executed by conducting verification with test inputs and comparing the observed output with the expected one. It is difficult to test the whole system, therefore selective components of the system are considered for testing. If the input space of the System Under Test (SUT) is quite very large, its testing has to be conducted with a small subset of test cases. Embedded systems generally use microprocessors that contain many functions of a computer on a single device (i.e. System-on-chip). Embedded software is often integrated in highly complex devices. Medical device software, automotive software, avionics software, military software and railway software are all used to control devices or vehicles on which people's lives depend. A fault in that software may not just be inconvenient, it could be disastrous.

II. BACKGROUND

A. Embedded System

An embedded system is a specialized microcontroller based computing device used as a part of another system or machine. Normally, an embedded system is built on a single microprocessor board with the software stored in ROM. Some common examples of applications of embedded systems are telecommunications, automobiles, consumer electronics, and plant control. Even though the application domains are dissimilar from each other, they have universal organization in functional configuration. A layered embedded system structure, including application programming interfaces, hardware-dependent software, application software, and hardware platform is shown in Fig. 1 [5][6]. Application program interfaces are essential for communication between the hardware-dependent software (System Software) and the application layer of software (Application programs). The hardware-dependent software is attached with the external physical hardware and network. Real-time Operating System (RTOS) and device drivers are closely attached to the hardware platform of the system. According to an application domain, performance and size are the constraints that usually influence the hardware platform. As per specific application, a processor and memory system must meet a minimum requirement.

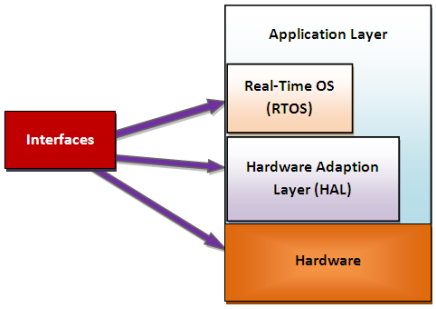

Fig. 1 Architecture of an embedded system

B. Designing Embedded System

An embedded system designing includes main four steps [3]. The basic steps are the following:

- Requirements specification
- Hardware and software partitioning
- Software design
- Hardware design
- Interface design
- System integration and test

System developers can derive required functions after evaluating system requirements. These functions are considered for allocation of hardware or software.

DOI 10.17148/IJARCCE.2016.56120

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

Development of hardware and software is done parallel with the interface design. After development of all required hardware and software components, they are integrated to build a system and go ahead for the testing of system. The system level design steps are shown in Fig. 2.

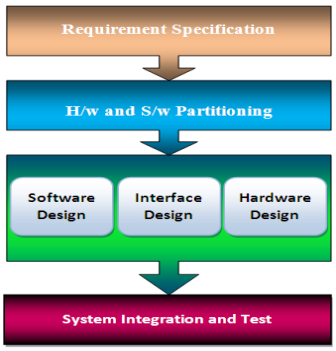

Fig. 2 System-level design processes

C. Model Driven Design

Model driven design (MDD) is based on the efficient use of models as a primary objective throughout the software engineering life cycle. The main objective of MDD is to provide a central role to functional models in the specification, design, integration, and validation of software. Model driven development uses models to represent a system's elements, the structural relationships between them and their dynamic interactions and behaviour. Modelling the structural relationship supports design exploration and system partitioning. The modelling behaviour and interactions are required to verify designs by verifying models and for code generation. But many embedded software developers hesitate to accept the generated code. The rejection of code by developers means loss of MDD advantages. Use of the MDD approach means accepting automatic code generation from models.

III. TEST FRAMEWORK FOR MODEL DRIVEN DESIGN

Using MDD, we can eliminate specification and design errors early in the development cycle where they are cheapest and easiest to rectify. It helps to increase the degree of automation that can be applied to the development process by means of automatic code generation. The test driven development of MDD[7] facilitate parallel hardware/software design by enabling system models to be tested using a simulated execution mechanism on development hosts before the target system is available. Such a development reduces the required testing effort by applying automated formal verification methods to the functional models in collaboration with simulating execution behaviour instead of relying solely on testing the implemented program code. The Fig. 3 describes the proposed approach of model driven development of embedded system and testing with the real-time data. The MDD platforms widely used in the safety, security and mission-critical domains such as rail transportation, aerospace, automotive and military applications is gradually increasing. If an MDD platform is used with sufficient verification and testing facilities on design models, the unit testing requirement can be reduced [8]. The MDD standards differentiate between the operational embedded software changed as part of the system from the modelling tools used to build that software. The different standards for safety-critical embedded software often needs the tool users to carry out an evaluation of the tool to classify it according to whether or not the tool itself can introduce errors into the operational embedded software and to perform an evaluation of the tool against the appropriate criteria for safe and secure use [4].

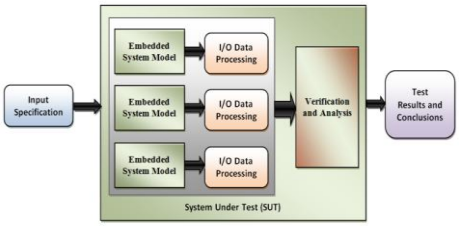

Fig. 3 Test framework for model driven design of embedded system

IV. MODELLING AND TESTING OF EMBEDDED SYSTEM

In this approach we used Simulink [9] for the development of model based embedded system. Here we developed two models of embedded system and combined them in one model for the processing and testing. The combined model first captures image from the run-time source. The first sub-model designed by using Simulink system generator blocks and second sub-model designed by hard coding in Matlab. The Fig. 4 depicts the embedded system model using Simulink. The plotting of image histograms is done using the Simulink tool. The histogram equalization is the process which increases the dynamic range of the gray-level in a low-contrast image to cover a full range of gray-levels. The comparative analysis of histograms and error graph is done. This process repeated for the other run-time images. The final conclusion drawn from the analysis that, which model is best for the development of embedded system hardware and software. The following sections describe the overview of the used components and basic methodologies used for this research work.

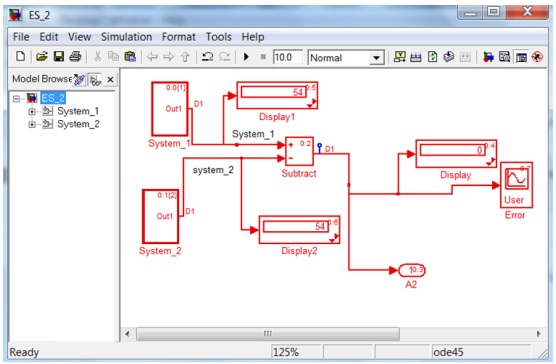

Fig. 4 Simulink model for embedded system

DOI 10.17148/IJARCCE.2016.56120

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

A. Overview of MATLAB Simulink

Simulink allows automatic code generation, simulation and continuous testing and verification of embedded systems. Simulink has a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. Its integration with MATLAB, allows you to integrate MATLAB algorithms into models and also export results after the simulation to MATLAB for further analysis. According to MathWorks, Simulink coupled with their other products to automatically generate C source code for real-time implementation of systems. Due to Simulink flexibility and capacity for quick iteration it is widely used tool for embedded system designs and production systems. Simulink provides Embedded Coder (EC) to generate efficient code for embedded systems. It also allows verification and validation of models through requirement traceability, modeling style checking and model coverage analysis. Simulink Design Verifier (SDV) capable of identifying design errors such as division by zero, integer overflow and dead logic. It also generates test case scenarios for model checking within the environment. The Simulink tool TPT used to perform a formal verification and validation process to stimulate Simulink models. It is also used during the development phase where the developer generates inputs to test the system [9][10].

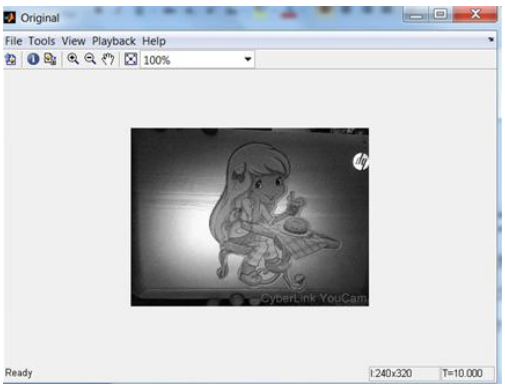

Fig. 4 Original image captured from web camera

B. Histogram

The histogram is statically representation of frequency of intensity values in the image. It is used to improve the visual appearance of an image and also be used to determine what type of processing has been applied to an image. Histograms can be used to identify a wide range of image defects, such as saturation, spikes and gaps, impact of image compression. The histogram of a digital image with L total possible intensity levels in the range [0, G] is defined as the discrete function:

$$h(r_k) = n_k$$

where, r_k is the k^{th} intensity level in the interval [0, G]
n_k is the number of pixels in the image whose intensity level is [0, G]

G: [255 for images of class uint8, 65535 for images class uint16 and 1.0 for images of class double].

Normalization of histograms can be obtained by dividing all elements of by $h(r_k)$ the total number of pixels in the image.

$$P(r_k) = \frac{h(r_k)}{n} = \frac{n_k}{n} \qquad \text{k=1, 2, ..., L}$$

n=total number of pixels

C. Histogram Equalization

It is a method which increases the dynamic range of the gray-level in a low-contrast image to cover a full range of gray-levels. Histogram equalization is achieved by having a transformation function (TF) T_r, which can be defined to be the Cumulative Distribution Function (CDF) of a given Probability Density Function (PDF) of a gray-level in a given image.

For intensity levels that are continuous quantities normalised to the range [0, 1].

Let $P_r(r)$ – probability density function (PDF) of the intensity levels. The following transformation on the input levels to obtain output levels, s:

$$s = T_r = \int_0^r P_r(\omega) d\omega$$

$$\omega - \text{dummy variable of integartion}$$

The general histogram equalization formula is:

$$h_v = round\left(\frac{cdf(v) - cdf_{min}}{(M * N) - cdf_{min}} \times (L-1)\right)$$

$$\text{cdf}_{\text{min}} - \text{minimum value of cdf}$$

$$(\text{M} * \text{N}) - \text{image's number of pixels. (M} - \text{width, N} - \text{height)}$$

L - Number of gray scale levels.

D. Equalization Transformation Function

In histogram equalization, the spreading out of the frequencies in an image (or equalizing the image) is a simple way to improve dark or washed out images. An original image is converted into 'hist eq' image using a transformation function (see Fig. 5).

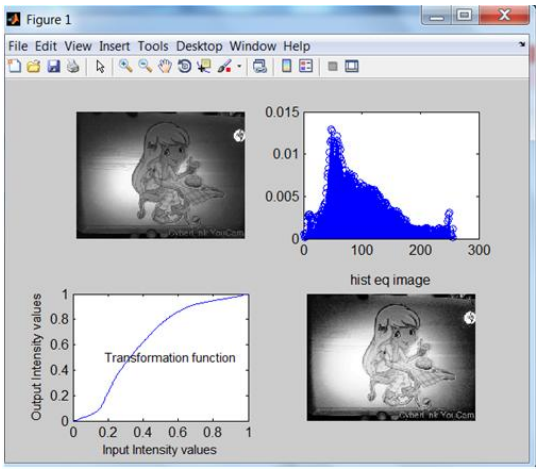

Fig. 5 Original image captured from web camera

DOI 10.17148/IJARCCE.2016.56120

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

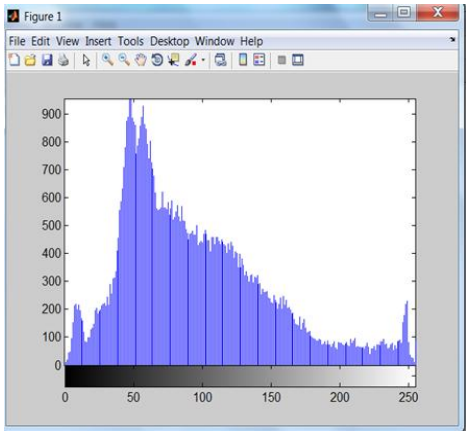

Fig. 6(a) Histogram of original image

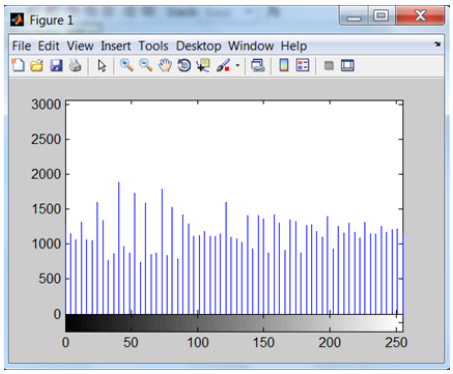

Fig. 6(b) Histogram of equalized image

E. Binning of Histogram

The binning of histogram is done to limit the large value and further process on histogram makes easy. Binning is done to reduce a large number of computations in high resolution image. For example If an image is of 32 bit size, then "32-bit image = 232 = 4,294,967,296 columns". Ultimately, process calculations require higher number of the bit size of computation and time require for execution automatically increased.

So, given the image $I : \Omega \rightarrow [0, K - 1]$, the binned histogram for I is the function

$h(i) = \text{card}\{(u,v) | a_i \leq I(u,v) < a_{i+1}\}$

where $0 = a_0 < a_1 \ldots < a_B = K$

$\text{card} \rightarrow$ Number (size of set)of pixels

$I(u,v) \rightarrow$ Pixel's intensity is between a_i and a_{i+1}

Bin size = (Number of distinct values of image/ Number of bins)

To create 256 bins from 14-bit image use the following:

$$\text{Bin size} = \frac{2^{14}}{256} = 64$$

$h(0) \leftarrow 0 \leq I(u,v) < 64$

$h(0) \leftarrow 64 \leq I(u,v) < 128$

$h(0) \leftarrow 128 \leq I(u,v) < 192$

. . . .

. . . .

. . . .

$h(j) \leftarrow a_j \leq I(u,v) < a_{j+1}$

. . . .

. . . .

$h(255) \leftarrow 16320 \leq I(u,v) < 16384$

F. Calculations and Results

The original Image size is 240 x 320.

Let us consider [M,N]=[8 8] of original image for showing the result. An 8-bit gray-scale image has the following values as shown in Table 1.

TABLE I
VALUES OF $I(u,v)$

22	18	15	16	20	24	26	25
22	20	19	21	24	26	25	24
22	22	23	25	27	27	25	23
24	24	25	25	26	26	25	25
26	26	24	23	23	24	26	28
26	26	25	23	22	24	28	31
24	26	27	27	25	26	30	33
22	26	30	31	30	29	31	34

Fig. 7 Histogram equalization on same image (Histogram Equalized3 is most accurate among the others)

The histogram for this sub image is shown in the Table 2 below. Pixel values that have a zero count are excluded.

TABLE II
PIXEL VALUES AND COUNT

Value	Count	Value	Count	Value	Count
15	1	22	6	29	1
16	1	23	4	30	3
17	0	24	9	31	2
18	1	25	10	32	0
19	1	26	12	33	1
20	2	27	4	34	1
21	1	28	2	35	0

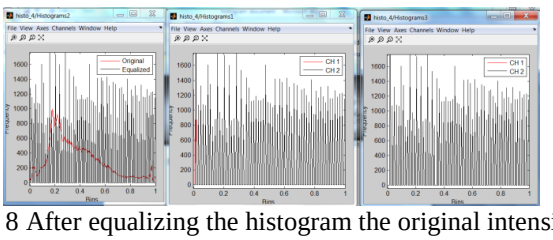

Fig. 8 After equalizing the histogram the original intensity range is vanished as shown in Fig.

The Fig. 9 shows the histograms and error plotting of model-1 and model-2 after processing on the same image. The comparison between these results is helpful to identify

DOI 10.17148/IJARCCE.2016.56120

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 6, June 2016

the best model for the further development of a specified embedded system.

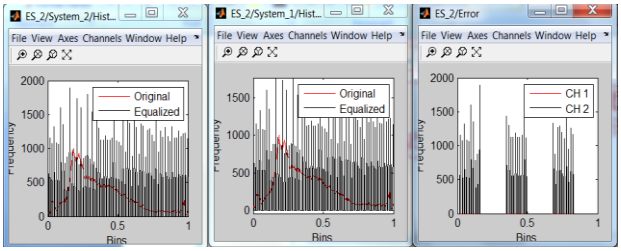

Fig. 9 Histogram and error histogram

V. CONCLUSION AND FUTURE WORK

Embedded systems are the integral part of our day-to-day life. The various devices such as medical instrumentation, bank machines, industry control systems, household machines and many more are now depending on the embedded system mechanism. The testing of embedded devices is becoming crucial day-by-day due to the increasing device complexities and safety and security requirements for human beings. The model based testing approach for embedded system is a great solution over the building of physical systems and testing. The modelling tools and languages are helpful to avoid unnecessary wasting of time and money in the physical development of embedded systems. In this approach real-time data accepted by the system and tested according to the specifications. The error verification using the plotting of histograms makes it easy to understand the behaviour of the model. Furthermore, such type of testing can be applied to the continuous streaming of the images and the system behaviour can be tested. This type of work is required for the testing of multiple models and for the heterogeneous designs of embedded system models. The combination of image processing with the Neuro-fuzzy approach may be the future approach in the way of model based embedded system testing.

REFERENCES

[1] Hartman, Alan. "Model-Based Test Generation Tools". http://www.agedis.de, 2011.
[2] Pravin Karmore and Pradeep Butey, "Analysis of Model-based Testing Methodology for Embedded Systems", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 6, Issue 5, pp. 308-314, May- 2016,.
[3] Byeongdo Kang, Young-Jik Kwon, Roger Y. Lee, "A Design and Test Technique for Embedded Software", Proceedings of the 2005 Third ACIS IEEE Int'l Conference on Software Engineering Research, Management and Applications (SERA'05), 2005.
[4] Pravin Y. Karmore and Pradeep K. Butey, "Technical View of Testing Methodologies for Diverse Designs of Embedded System", International Journal on Information Technology Management, vol. 2, ISSN2277 8659, 2013, pp. 123–131.
[5] Alberto Sangiovanni-Vincentelli and Grant Martin, "Platform-Based Design and Software Design Methodology for Embedded Systems," IEEE Design & Test of Computers, November-December, 2001, pp.23-33.
[6] M. Sgroi, L. Lavagno, and A. Sangiovanni Vincentelli, "Formal Models for Embedded Systems Design," IEEE Design & Test of Computers, April-June, 2000, pp.2-15.
[7] David Astels, "Test Driven Development: A Practical Guide", Upper Saddle River, NJ: Prentice Hall PTR, 2003.
[8] Pravin Y. Karmore, Pradeep K. Butey, "Test Driven Development of Model Driven Embedded Systems", International Journal of Science, Engineering and Technology Research (IJSETR), ISSN: 2278 – 7798, vol.5, Issue 6, pp. 1941-1945, June. 2016.
[9] Agam Kumar Tyagi, "Matlab and Simulink for Engineers", Oxford publication.
[10] James B. Dabney, Thomas L. Harman, "Mastering Simulink", Prentice Hall publication.

BIOGRAPHIES

Pravin Y. Karmore pursuing Ph.D. in Computer Science from RTM Nagpur University, Nagpur. He is obtained Master in Computer Applications degree from RTM Nagpur University, Nagpur. And M.Phil. degree in Computer Science from Alagappa University, Karaikudi. At present he is working as Assistant Professor at Dept. of Computer Applications, Shri Ramdeobaba College of Engineering and Management, Nagpur. His research area is Software Engineering, Embedded Systems, Artificial Neural Networks and Fuzzy Logic.

Dr. Padeep K. Butey obtained M.Sc. degree and PGDCS&A from RTM Nagpur University, Nagpur. He obtained his Ph.D. degree in Computer Science from RTM Nagpur University, Nagpur. Now he is working as Associate Professor and Head at Dept. of Computer Science, Kamla Nehru College, Nagpur. He has published more than 40 research papers in various national and international conferences and journals. His research area includes RDBMS, Data Mining, Artificial Neural Networks and Fuzzy Logic.