International Journal of Electrical and Computer Engineering (IJECE)
Vol. 5, No. 3, June 2015, pp. 491~502
ISSN: 2088-8708

Software for Simplifying Embedded System Design Based on Event-Driven Method

Maman Abdurohman*, Arif Sasongko**
*School of Computing, Telkom University, Indonesia
** School of Electrical and Informatics Engineering, Bandung Institute of Technology, Indonesia

Article Info

Article history:

Received Mar 1, 2015
Revised Apr 19, 2015
Accepted May 3, 2015

Keyword:

Embedded System
Event Driven
Software
State Chart

ABSTRACT

Complexity of embedded system application increases along with the escalation of market demand. Embedded system design process must be enhanced to face design complexity problem. One of challenges in designing embedded system is speed, accuracy, and flexibility. The design process is usually conducted recursively to fulfill requirement of user and optimization. To solve this problem, it needs a system design that is flexible for adaptation. One of solutions is by optimizing all or some of the design steps. This paper proposes a design framework with an automatic framework code generator with of event driven approach. This software is a part of a design flow which is flexible and fast. Tron game and simple calculator are presented as a case study. Test result shows that this framework generator can increase speed of design's flexibility.

Corresponding Author:

Maman Abdurohman
School of Computing
Telkom University
Jln. Telekomunikasi no. 1 Bandung 40257, Indonesia
Email: abdurohman@telkomuniversity.ac.id, m_abdurohman@yahoo.com

1. INTRODUCTION

The increase of embedded system's complexity affects the design process. It is becoming more complex and time-consuming. Developing embedded system is not only designing software and hardware parts but also designing and integrating both them at the same time [1].

The embedded system design process engages many experts who have different point of views to the system. Not all the experts are capable in variety of fields; therefore, we need a standard form for them. It needs a framework that is agreed by a team which whole interaction among algorithms including the hardware and software parts.

In designing embedded system, one of problems that must be handled is the optimization between software or hardware implementation and conformance to the user requirement. Sometimes, this process needs repetition in defining framework design. Detail implementation step is conducted after its design really fulfills the optimization either from the side of speed, budget, or function. Regarding the easiness of modification of architecture for algorithm implementation, it needs a flexibility of framework. Generating framework automatically is one of good alternatives to come to this flexibility. There are many implementation of software for automation and adaptive system [2-5]. The flexibility is needed to handle the facts that product age is very short. Sometimes, new version of one product must be launched several months later after the previous product is launched. It needs short time to change previous design into the newest model, and certainly, it will need a flexible framework. Many efforts/researchs have been done for design improving in the field hardware and software and event driven context [6-9].

This paper proposes a well-defined new design method that is started from the first model. Characteristics of this model are: easy to understand, precisionto use in generating code, and flexible to

modify. The quite similar method that has been proposed is [10]. This method generates framework code from UML diagram. The difference of the method from this paper is [10] in terms of implementation that is specified on a platform or OS, while in this paper, the method of generating framework code for software and hardware does not depends on a specific platform.

2. RESEARCH METHOD

The method of developing embedded system using traditional way has much weakness such as 1) We need someone who understands big image from the system that will be built; 2) We need a lot of coordination among planners to guarantee the similarity of perception and process among sub systems.

Designing process and coordination are conducted manually. The weakness in big scale is the inability to support reusable concept. This paper comes up with a new designing process with event-driven method approach. It is taken because naturally, the system works reactively to an event. Its design is started by defining the system as several interacting subsystem in form of Finite State Machine with event-driven based communication. Based on the design that is defined correctly, the next step is generating framework automatically for whole application. Figure 1 shows the framework design concept.

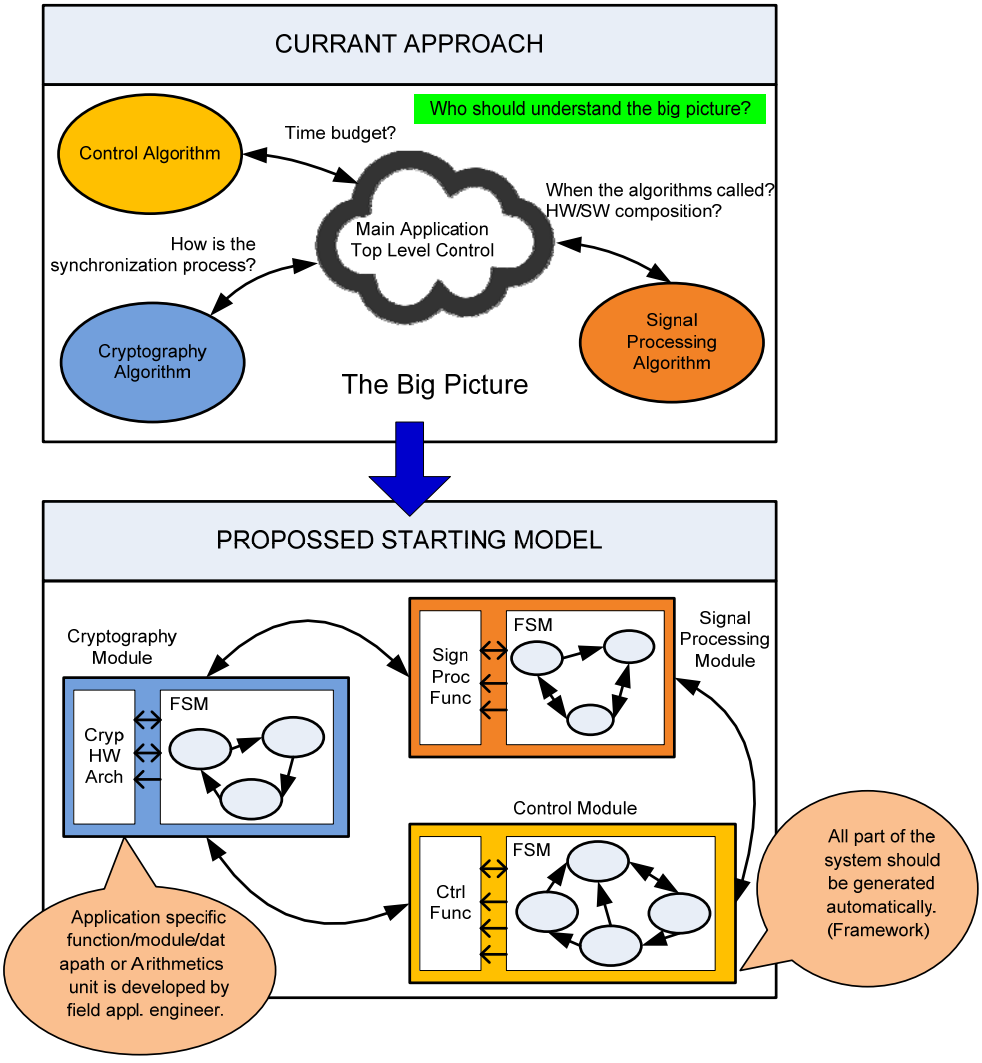

Figure 1. Framework Design Concept

The first model describes that system is an assemblage of Finite State Machine and events. Two uses of the first model are: 1) as the first input for generating framework code, and 2) as a model that is easy

to be understood by engineers/developers. Output that is produced from framework generator is: 1) FSM code for all blocks, 2) communication among FSM, 3) Prototype function (software), 4) hardware interface module that is used by FSM, and 5) Instantiation of function.

Output from framework generator is framework code for software and hardware that must still be completed by programmer by inserting function into each block system. A programmer can focus on implementing function as efficiently as possible. In the last step, the integration on the whole part of software and hardware was conducted.

Approach of event driven is used because the embedded system naturally works in this way. It is reactive to a signal and environment [11]. The signal that is received by this embedded system is as an event that influences a system to do something, as Figure 2. The strength of this design is that either software or hardware uses event driven approach so that we can do designing process with software and hardware.

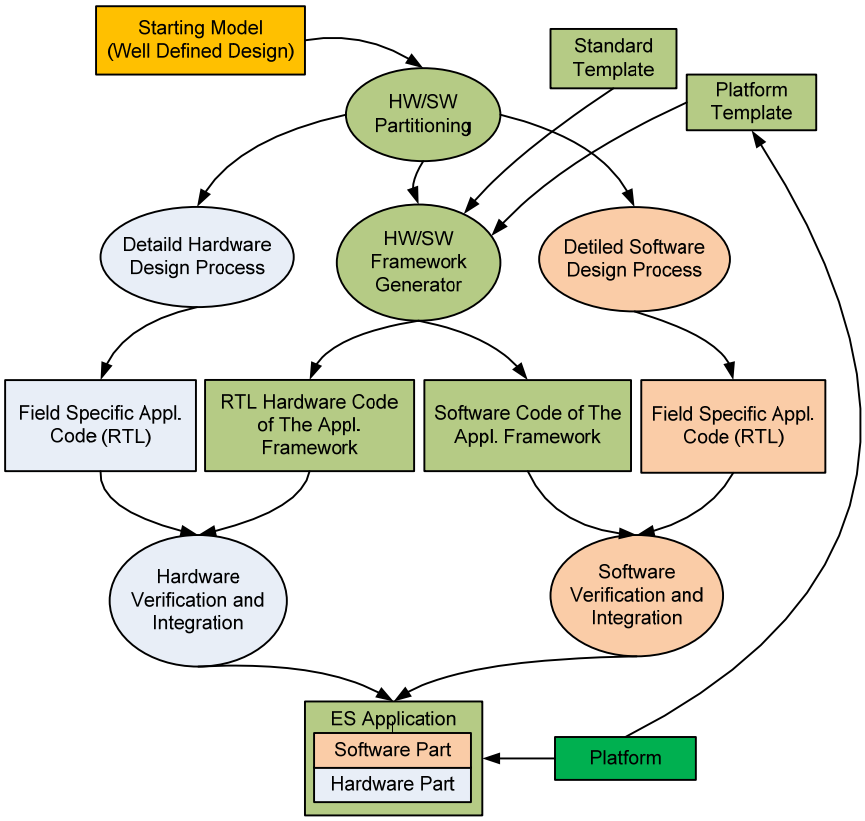

Figure 2. Flow of Embedded System Design

There are several assumptions that are used in this method:

1. Base assumption in this method is that all management stuffs need much less computing power compared to the application ofbehavior part. These management stuffs are anything that relates to synchronization, communication among tasks, and context switches.
2. Communication among FSM is implemented as sending and receiving signal. This communication signal has a role as event for a sub system.
3. Implementing system uses platform covering processor, OS, compiler, libraries and FPGA. It means that processor and RTOS that are not built specifically for this implementation.

The Structure of Generated Code

In the event-driven method views, embedded system design consists of three parts such as event generator, dispatcher and handler module. Automatically generated code in this paper is consists co three parts namely event generator, dispatcher and event handler. The behavior that will communicate with environment is done manually, as Figure 3. Information from environment or other systems are processed by event converter. The output of event converter will dispatch to suitable event handler for next action.

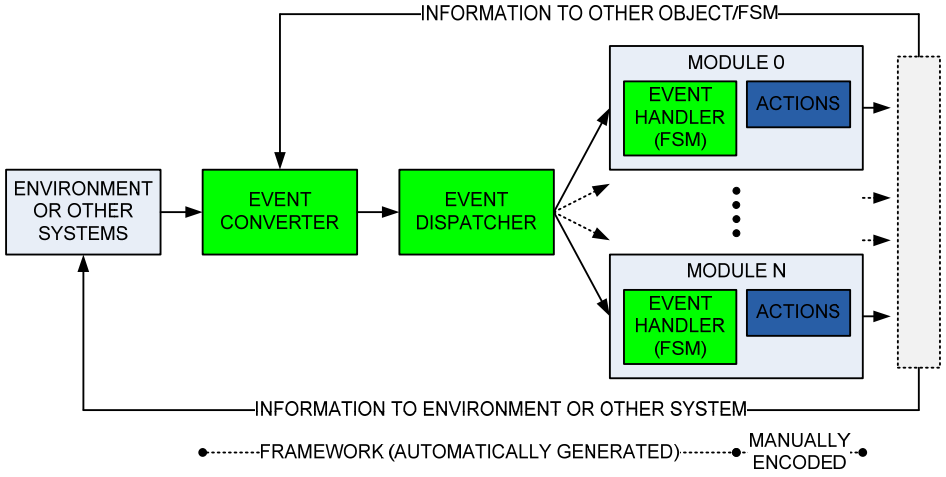

Figure 3. Generated Code Structure

The Starting Model

Starting model as Figure 4 is a description which the tool takes as input. This design model must have three criterias: 1. Easy to be understood because it will be discussed by many team members with difference background; 2. Precise since this model has to be used as the input of framework automatic generator; and 3. The model should support modularity.

At each module, additional information is needed to describe the module and inter-module communication. The featuress of the initial models are as follows: 1. The use of FSM models which can be understood either by software engineers or hardware engineers; 2. It is abstract enough and it doesn't need detail process contained in each module; 3. FSM provides information about control, dataflow, and communication between and within modules; 4. Functional and FSM diagrams are easily visualized using diagrams or pictures; and 5. Key points in the model are how the communication and interface are described (as event).

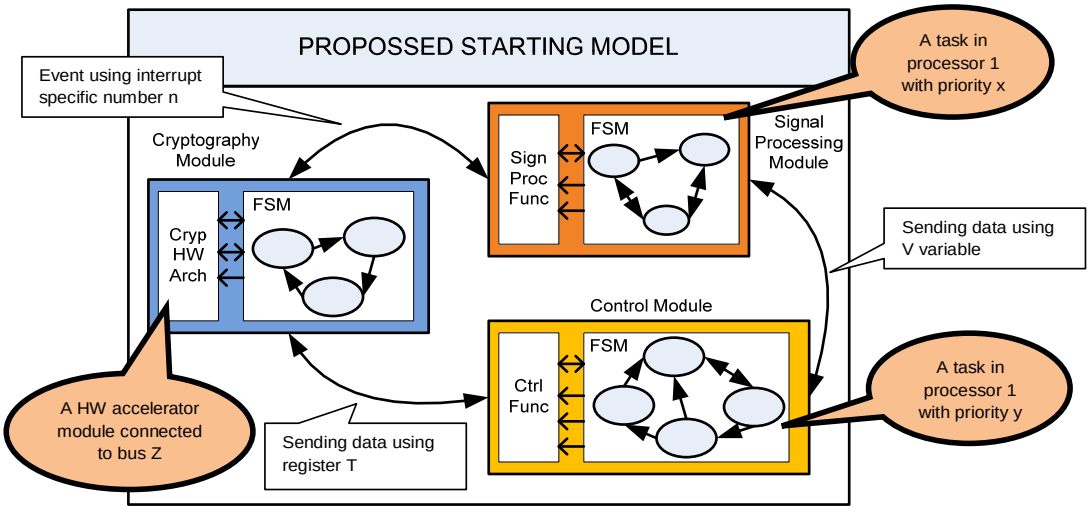

Figure 4. The Starting Model of the System

Platform

Platform is an artifact that is usedfor developing embedded systems. Hardware part of the platform can be a processor, DMA controller, peripherals and bus system. Part of the software in system can be an operating system, library or standard task. Figure 5 shows the platform used.

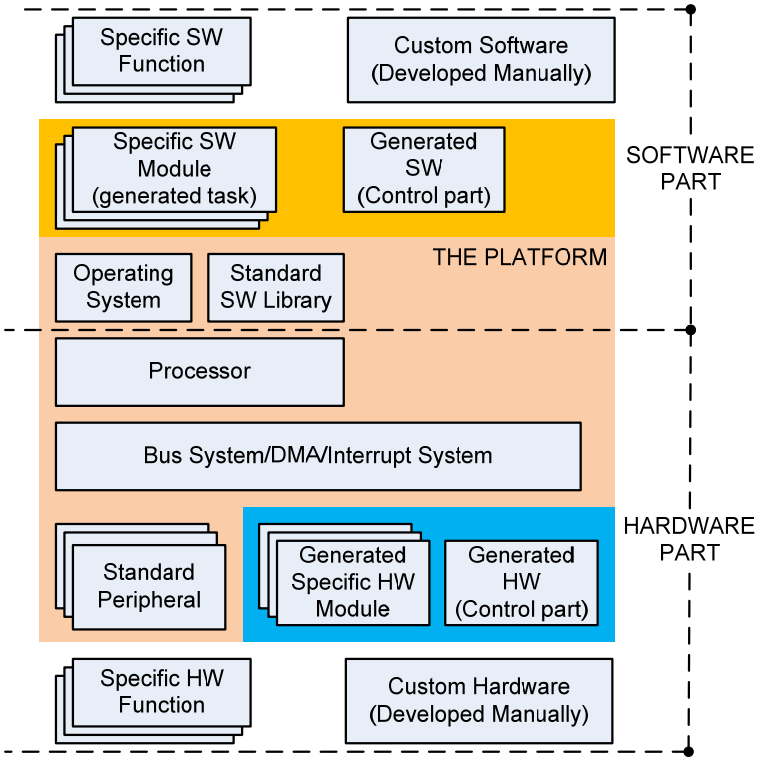

Figure 5. The Platform and the Framework of the System

FPGA board can be used as a prototyping platform for development of embedded applications system. The communication of hardware and software in the embedded system is shown in Figure 6. The processor used maybe a physical processor or soft processors in FPGA.

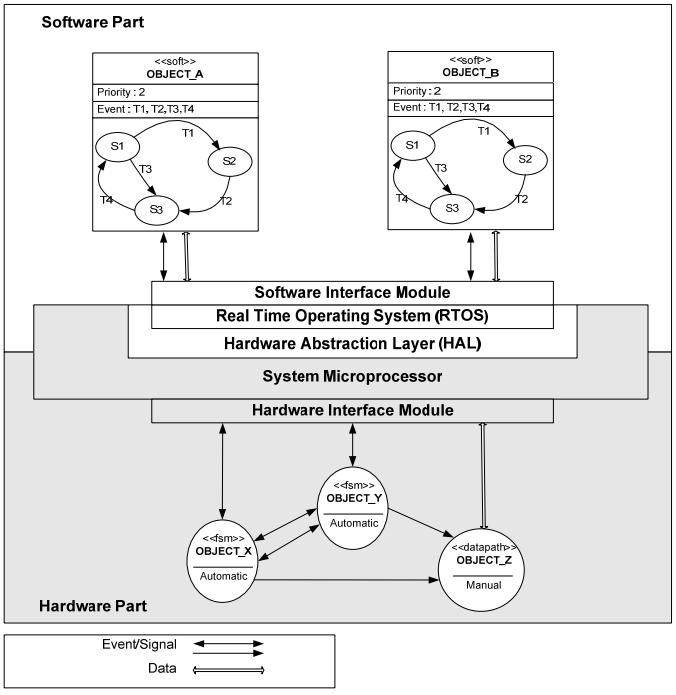

Figure 6. The communication of hardware and software in the embedded system

3. FRAMEWORK GENERATOR SYSTEM

3.1. Embedded System Platform

Embedded system consists of two parts: hardware and software. Generally, hardware consists of processor, coprocessor, and supported components that are located on data path like plus, mines, times, divides, registers etc. While, generally, software consists of operating system, function and custom template that support application.

The problem that must be answered in developing embedded system is the composition of optimum hardware or software for one need from the performance and budget perspective. This paper focuses on designing and fulfilling needs of user and application performance. In the hardware step, partitions that need fast computation like arithmetic components are implemented as a part of hardware. While, functions that are related to user interface like button for trigger are implemented as software. Target of implementation code for software is C file while hardware implementation is in VHDL file. System implementationis used using Field Programmable Grid Array (FPGA), and microprocessor used in this system is ARM and MicroC OS/II operation system.

3.2. Event Driven Method

Naturally, event-driven concept (Figure 7) is used in implementing embedded system. This approach is more efficient that another one that is proactive. The system is in idle situation as waiting event from outside, when the event happens so the system will react suitable with the kinds of event and its last system'sstatus. This event triggers the system to change from one status to another. Event from outside is handled by software and hardware. In the software is event generator that generates events that must be handled by its system. All events in the system are handled in line of global to be arranged by dispatcher, and they will be spread to appropriate event handler. In the hardware, events are received from their environment and handled by distributor. Then, each event is handled by appropriate event handler. Output from one event activates another event that will be transferred by distributor.

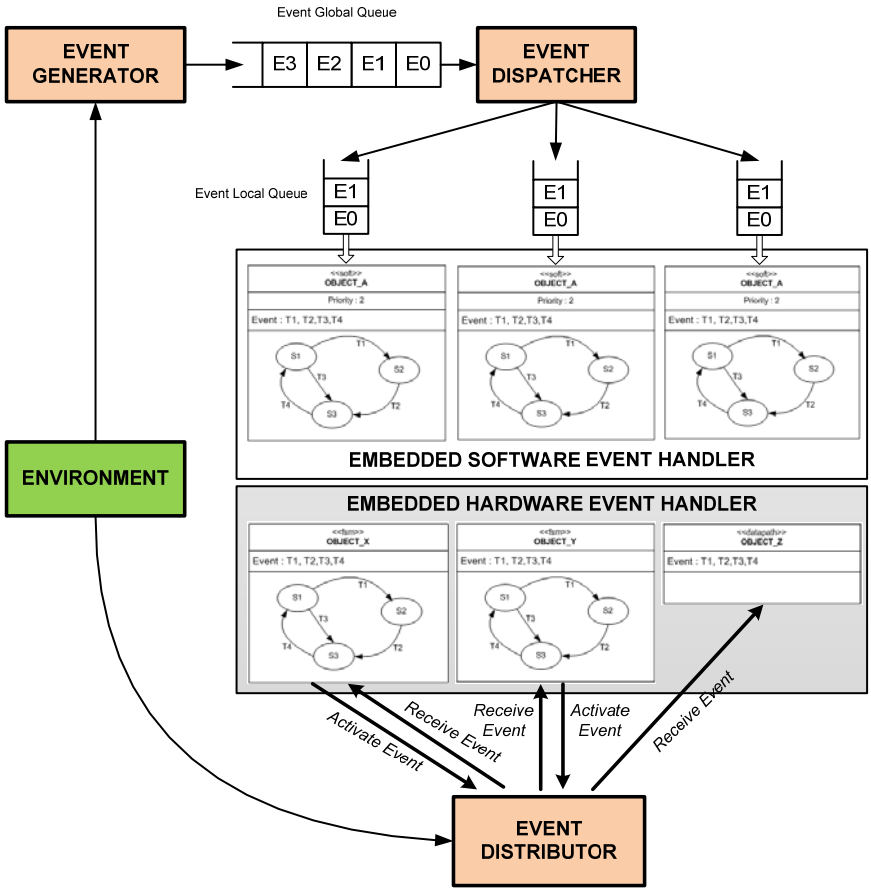

Figure 7. Event-driven concept in Embedded Hardware and Software

3.3. Framework Generator System Workflow

Starting from design that is defined correctly by designers, till this step, the system has been defined repeatedly by considering several functional needs and limitations. Design that is defined correctly is transformed automatically and becoming several frameworks of embedded software and hardware, as Figure 8.

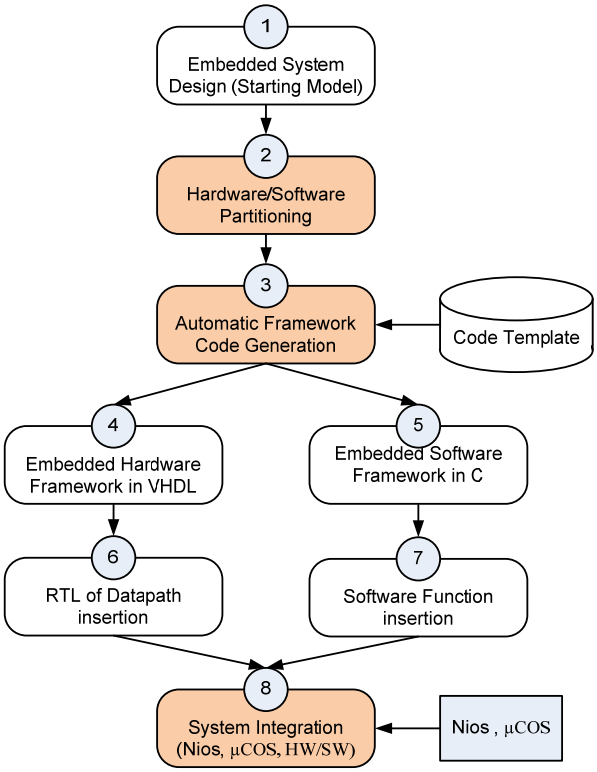

Figure 8. Framework generator system process

Framework is entity for software function or hardware components. All frameworks are empty, and must be completed by programmer with detail code that is appropriate to each function. Detail implementation of software modules use program C and Hardware's uses VHDL language. It needs to be noticed that the strength of this transformation concept is generating hardware and software framework automatically. After detail implementation using C language for embedded software using VHDL for embedded hardware, the next step is testing in FPGA board. System implementation uses NIOS microprocessor and MicroC OS II operating system.

3.4. Hardware/Software Framework Code Generator

During the implementation of framework code generator, there were two steps conducted. They were behavior determination by defining FSM objects for embedded hardware/software and code structure writing in C (for embedded software) and in VHDL (for embedded hardware), as Figure 9.

Object behavior of embedded hardware and software functions delineated in event handler was defined using Finite State Machine (FSM). Each object was defined in FSM, both hardware and software. In defining FSM, there was a set of states and transitions. Transition is an event that becomes a trigger to move from one state to another. State is filled with function calls to be carried out. FSM can be translated into program codes (C for software and VHDL for hardware) using nested switch statement concept. The transformation from FSM into program codes used nested switch principles. There is a main switch to determine state position. The transformation/action performed by the system is determined by the occurring event. Both embedded software and hardware were implemented using similar method. The next step was filling out function details in each state.

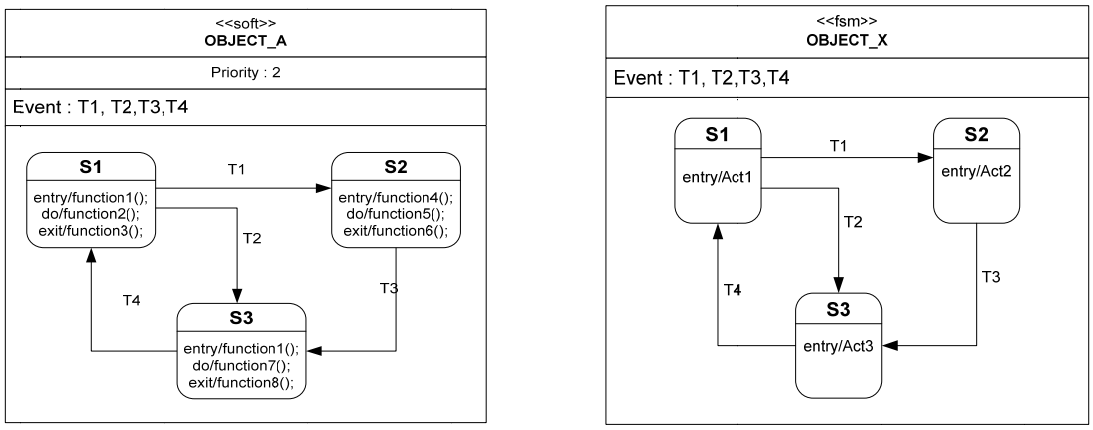

Figure 9. FSM from an object on (a) Embedded Software and (b) Embedded Hardware

Software Impelementation

```
switch (state) {
    case S1 :  switch (condition){ case entry : fucntion1 ();
                    case steady : fucntion1 ();
                        switch (event) {   case T1 : TRAN_TO(S2)
                                           case T2 : TRAN_TO(S3)
                        }
                    case exit : function3 ();   }
    case S2 : switch (condition) { case entry : function4();
                    case steady : function5(); switch(event) {     case T3: TRAN_TO(S2)    }
                    case exit: function6();                 }
    case S3:
            switch(condition) {
                    case entry: function1();
                    case steady: function7();
            switch(event) {    case T4:          TRAN_TO(S2)          }
                    case exit: function8();      }
}
```

S1, S2 and S3 are states

entry, steady and exit are conditions in each state

T1, T2, T3 and T4 are the events as a system triger.

Hardware Implementation

```
process(current_state, T1, T2, T3, T4)
begin
case current_state is
        when S1 =>  --Act1
                if(T1='1') then    next_state<=S2;
                elseif(T2='1')     next_state<=S3;
                endif;
        when S2=>  --Act2
                if(T3='1') then    next_state<=S3
        when S3=> --Act3
                If(T4='1') then    next_state<=S1;
end case;
end process;
process(iRST_N, iCLK)
begin
    if(iRST_N = '0') then current_state<= S1;
    elseif (iCLK'eventandiCLK='1') then    current_state<= next_state;
    endif;
end process;
```

S1, S2 and S3 are states

T1, T2, T3 and T4 are the events as a system triger.

4. CASE STUDY ON TRON GAME AND SIMPLE CALCULATOR

4.1. Tron Game

As a proof that the generator has been running well, Trongame application framework has been generated as shown in Figure 10. By adding specific functions to the framework, Tron game can run well on Nios Platform. There are some steps in developing Trongame. The first step is defining objects required in Trongame. There are four objects defined as follows: Arena, Tron 1, Tron 2 and Laser. Arena functions as background, Tron 1 and Tron 2 as actors who perform position shift and Laser as obstacle that appears at certain position and time. The second step is defining Finite State Machine for each object: Arena has two states which are "Start" and "Play". Tron1 and Tron2 have "move_Hor" and "move_Ver" states. Meanwhile, Laser has "OFF" and "ON" states. The third step is making diagram objects in software generator. Niosprocessor and ucosoperation system were used as application platform. The fourth step is making state chart diagram for each object. The fifth step is generator framework that automatically generated files .c and .h. There are files for each object in which there are 4 file objects here: laser.c, tron1.c, tron2.c, arena.c. There are four files handling communication among objects: global.h, global.c, environment.h, environment.c.

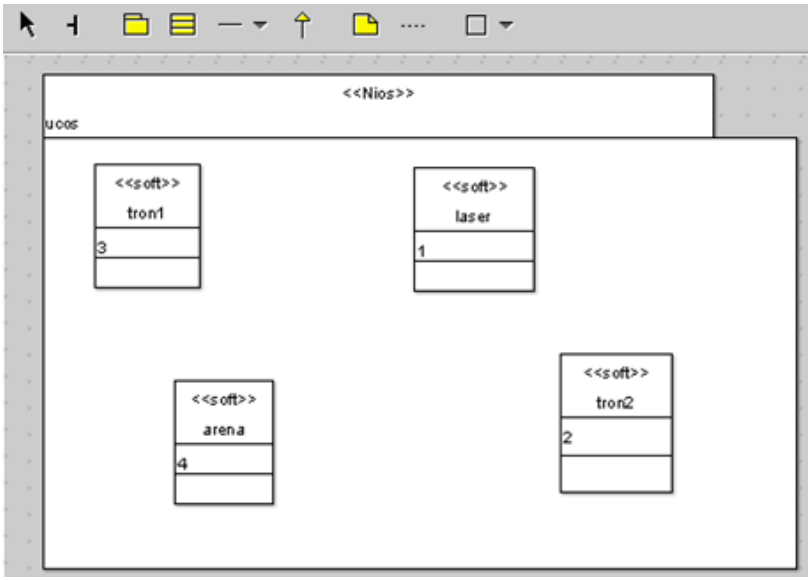

Figure 10. Process of object diagram development of tron game

4.2. Simple Calculator

In simple calculator, arithmetic functions are defined as hardware while button functions related to users are implemented as part of software. Before making diagram of embedded system used a set of generator, at first, we determine objects in the embedded system. Then, we determine whether the object is imported into embedded software or hardware. After that, the object will be imported into embedded software if its orders need high flexibility, and the object will be imported into embedded hardware if its orders need a high performance. So, in this calculator, extrapolation processes will be imported into embedded hardware whereas process of graphic screen and data management will be imported into embedded software. The objects were divided into two parts: the embedded software and embedded hardware. The functions that require quick process were defined as hardware such as adder, substractor, multiplier and main components, while the rest acts as software.

Each ofthese objects will have state machine described in form of state chart diagram, as Figure 11. The followings are the process of creating state chart from three objects.

1. State chart object-oneButton (embedded software)

OneButton object will perform a big number of 1if mouse pointer is on the button of the number 1, and it will import the number 1 to an active operand if the mouse is pressed. This object has the following states:

a) Undisplayed state

This state will perform a normal button of number 1 when an object leaves this state without doing any process. The events that cause the object turns out from this state are OVERONE (when the pointer is on the button of the number 1).

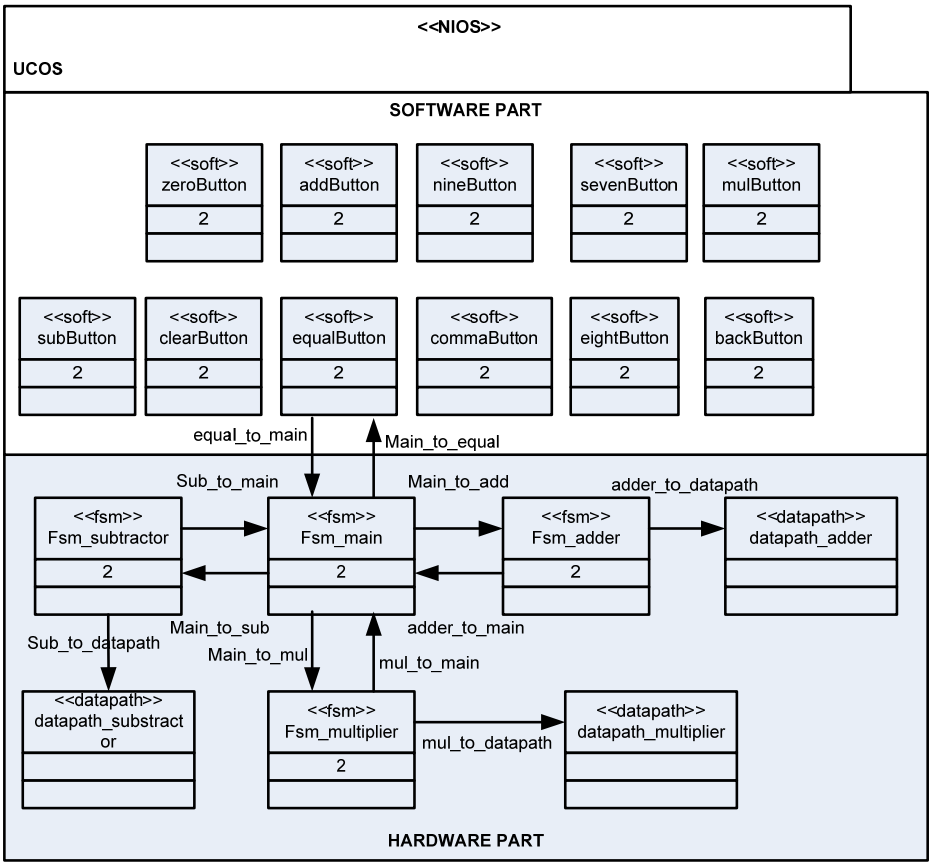

Figure 11. Object diagram of Embedded Hardware/Software

b) Displayed state
This state will perform a big button of number 1 when an object leaves this state without doing any process. The events that cause the object turn out from this state are ONE (by pressing the button of number 1), OUTONE (when the pointer goes off from area of the button of number 1).

c) Inserting State
This state will conduct the process of inserting number 1 into an active operand when an object enters to the state even though the object turns out from the state without doing any process. The events that cause the object turns out from this state are ONE (by pressing the button of number 1), OUTONE (when the pointer goes off from area of the button of number 1), OVERONE (when the pointer is on the button of the number 1).

2. Object state chart-Operand Controller (embedded software)

This object will control and determine operands that must be active. The operands of this calculator project are *operand 1 Int, operand 1 Frac, operand 2 Int, operand 2 Frac*. Consider these states:

a) Operand 1 int.
When an object enters into the state, it will activate *operand1int* and deactivate it when going off from this state. The events that cause the object turns out from this state are COMMA (pressing button of comma), ADD (pressing button +), MUL (pressing button *), SUB (pressing button -), CLR (pressing button C).

b) Operand 1 frac.
When an object enters into the state, it will activate *operand1frac.* And deactivate it when going off from this state. The events that cause the object turns out from this state are ADD (pressing button +), MUL (pressing button *), SUB (pressing button -), CLR (pressing button C) FULLBACK (If data in *operand 1 frac.* is used up).

c) Operand 2 int.
When an object enters into the state, it will activate *operand 2 int.* and deactivate it when going off from this state. The events that cause the object turns out from this state are COMMA (pressing button of comma), CLR (pressing button C), EQL (pressing button =).

d) Operand 2 frac.
When an object enters into the state, it will activate *operand 2 frac.* and deactivate it when going off from this state. The events that cause the object turns out from this state are CLR (pressing button C), EQL (pressing button =).

3. Object state chart Fsm_main (embedded hardware)
This object controls the process of computation that is being active. Its states are *S-idle* (as a hint of no action), *S-adding* (activating summation signal), *S-multiplying* (activating multiplication signal), *S-subtracting* (activating subtraction signal), *S-ready* (activating signal that signifies that computation has finished, and *S-finish* (resetting all active signals).

5. CONCLUSION

In this paper, application for automatic generator framework of embedded system has been explained. The proof was carried out by implementing Trongame application and simple calculator. The stages of application development used new concept in designing embedded system. By using new workflow, the process of application development is more efficient and flexible as some processes that require time has been managed by automatic generator framework software. For discussion, a tool that can perform detail transformation needs to be designed so the designing process can be carried out more quickly.

ACKNOWLEDGEMENTS

Unified Communication Lab of School of Computing - Telkom University, and ELKA Lab STEI – ITB those have supported financially and given sources in this study and also thanks to Sidik Prabowo, I Wayan Sutaya, Nina Asaad, and Ricky Henry Rawung who have helped us in implementing the system.

REFERENCES

[1] Frank Vahid, Tony Givargis (2002). Embedded System Design. A Unified Hardware-Software Introduction. Wiley.
[2] Hendra, A., Liang D., Pramono, S. Software Development of Automatic Data Collector for Bus Route Planning System. International Journal of Electrical and Computer Engineering (IJECE). Vol. 5. No. 1.
[3] Basjaruddin, N., Kuspriyanto, Saefudin, D., Nugraha, I. Developing Adaptive Cruise Control Based on Fuzzy Logic Using Hardware Simulation. International Journal of Electrical and Computer Engineering (IJECE). Vol. 4. No. 6.
[4] Jahromi, M., Faez, K. An Adaptive Steganography Scheme Based on Visual Quality and Embedding Capacity Improvement. International Journal of Electrical and Computer Engineering (IJECE). Vol. 4. No. 6.
[5] Permana GT, Abdurohman M, Khairudin M, Lutfi M. Automated Navigation System based on Weapon-Target Assignment. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2011; 9(3): 445-452.
[6] Abdurohman, M., Sasongko, A., & Rawung, R. (2013). Mobile Tracking System Based on Event Driven Method. *Applied Mechanics and Materials*, *321*, 536-540.
[7] Abdurohman, M., Herutomo, A., Suryani, V., Elmangoush, A., & Magedanz, T. (2013, October). Mobile tracking system using OpenMTC platform based on event driven method. In *Local Computer Networks Workshops (LCN Workshops), 2013 IEEE 38th Conference on* (pp. 856-860).
[8] Abdurohman, M., Ariyanto, E., & Anggis, N. (2015). System on Chip Design Methodology as Systematic Steps for Handling System on Chip Design Complexity Based on Hardware/Software Codesign. *Advanced Science Letters*, *21*(1), 66-69.
[9] Abdurohman, M., Kuspriyanto, Sutikno, S., Sasongko, A., "The New Embedded System Design Methodology For Improving Design Process Performance", *International Journal of Computer Science and Information Security*, Vol. 8, No. 1, 2010.
[10] Pong P. Chu (2011). Embedded SOPC Design With Nios II Processor and VHDL Examples. Wiley.
[11] Linus Tolke, Markus Klink (1996-2006). Cookbook for Developers of ArgoUML : An introduction to Developing ArgoUML. University of California.
[12] Nrusingh Prasad Dash, RanjanDasguptay, JayakarChepadaz and ArindamHalderx (2011). Event Driven Programming for Embedded Systems: A Finite State Machine Based Approach. *The Sixth International Conference on Systems.*
[13] MiroSamek (2009). Practical UML Statecharts in C/C++ : Event-Driven Programming for Embedded Systems. Second Edition. Elsevier.
[14] Jean J. Labrosse (1999). MicroC/OS-II : The Real-Time Kernel. R&D Books Lawrence.

BIOGRAPHIES OF AUTHORS

Maman Abdurohman received the B.S. degree in Informatics from the Informatics Faculty, Telkom University in 1988. He received the M.S. degree in Information Technology in 2014, and the Doctorate in Electrical and Informatics Engineering 2010 from the School of Eletrical and Informatics Engineering, Bandung Institute of Technology. He was involved in smart building research collaboration with IDEC division of Telkom corp., and Fraunhofer FOKUS Jermany from 2013 until now. He was involved in Bandung Smart City Council in the Technopolis project. Since 2000, he worked for the School of Computing in Telkom University in the capacity as a lecturer.

Arif Sasongko received the B.S. degree in Electrical Engineering in ITB. He received the M.S. and Doctoral degree in Grenoble University, France. He was involved in many projects on Hardware and Chip Design such as Indonesian WiMax Project, 4G chip designs and BTS monitoring system. His major research areas are Embedded System, SoC, VLSI design and Elliptic Curve Cryptography (ECC). He worked for Electrical Department in ITB since 2007.