PERANCANGAN DAN PEMBUATAN *GENERATOR FRAMEWORK* SISTEM EMBEDDED BERBASIS FSM (*FINATE STATE MACHINE*) DENGAN MEMANFAATKAN *OPEN SOURCE* ArgoUML

Oleh
I Wayan Sutaya
Jurusan Pendidikan Teknik Elektro, FTK, UNDIKSHA

ABSTRAK

Tujuan dari penelitian ini adalah untuk membuat perangkat generator framework sistem embedded. Framework sistem embedded yang dihasilkan oleh perangkat generator ini berupa kode-kode C dengan menggunakan model FSM (*Finite State Machine*). Perangkat generator ini dibuat dengan menkustomisasi aplikasi *opersource* ArgoUML. Perangkat generator yang dibuat telah diuji dengan membuat studi kasus aplikasi sistem embedded kalkulator. Dari studi kasus ini disimpulkan bahwa penggunaan perangkat generator ini bisa menghemat waktu pengerjaan proyek sebesar 70%.

Kata-kata kunci: framework, sistem embedded, FSM, ArgoUML.

ABSTRACT

The aim of this research is to develop an embedded system framework generator tool. The embedded system framework that is resulted by the generator tool are C codes with FSM (*Finite State Machine*) model. This generator tool was developed by customizing an open source ArgoUML application. The generator tool that had been developed had been tested by making a case study of a calculator embedded system application. From the case study, it is drawn a conclusion that using the generator tool can reduce time of finishing project about 70 %.

Keywords: framework, embedded system, FSM, ArgoUML.

1. PENDAHULUAN

Sebuah sistem embedded (sistem komputasi embedded) bisa secara sederhana didefinisikan sebagai sebuah sistem yang didesain untuk melakukan satu atau beberapa tugas spesifik (Frank, 2002). Sistem embedded ini bukan produk akhir tetapi sebuah bagian embedded dari sebuah sistem yang besar, dimana dalam sistem ini juga sering terdapat bagian elektronik dan mekanik tambahan. Sebagai perbandingan, sebuah sistem *general-purpose* (PC) adalah sebuah *general computing platform* dan PC sendiri adalah sebuah produk akhir. PC didesain agar

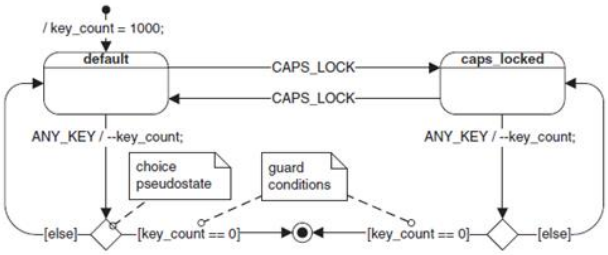
fleksibel dan mendukung berbagai jenis keperluan *end-user*. Program aplikasi dikembangkan berdasarkan *resource* yang ada pada PC. Sebuah desain sistem embedded yang bagus seharusnya hanya berisi *hardware resource* untuk memenuhi fungsi-fungsi yang diperlukan oleh aplikasi. Sedangkan disisi lain, sebuah sistem PC diharapkan untuk mendukung berbagai keperluan sehingga disediakan *hardware resource* yang berlebih. Dari pandangan ini sebuah sistem embedded bisa dipikirkan sebagai sebuah sistem komputasi dengan batasan *resource* yang ketat (Pong, 2011). Kemajuan teknologi dibidang sistem embedded saat ini berkembang sangat pesat. Industri-industri elektronik mengalihkan produk-produk mereka dari sebuah sistem yang dibangun dengan IC yang dibuat khusus ke sistem yang berbasis embedded seperti DSP, FPGA/PsoC, ASIC, Mikrokontroler dan lain sebagainya. Produk-produk elektronik ini mempunyai siklus hidup yang singkat sehingga proses pembuatan sistem embedded harus diotomatisasi sebanyak mungkin (Nam, 2004).

Tujuan dari penelitian ini adalah untuk membuat sebuah perangkat generator yang bisa digunakan untuk membuat framework sistem embedded secara otomatis. Beberapa keuntungan yang didapat dari penggunaan perangkat generator ini diantaranya: mempercepat proses pengerjaan suatu projek, memandu dalam membangun sebuah projek embedded, menghindarkan benang kusut yang mungkin terjadi dalam projek, dan projek tersebut akan mudah dikerjakan dalam sebuah kelompok.

Model framework yang dihasilkan oleh perangkat generator ini adalah kode-kode program yang menggunakan model FSM (*Finite State Machine*). Sebuah FSM adalah sebuah cara yang efisien untuk menspesifikasikan *constraints* dari keseluruhan *behavior* dari sebuah sistem (Miro, 2003). FSM sangat cocok digunakan untuk menentukan alur kendali dari sebuah sistem, tetapi kurang cocok sebagai alur data (Abdelaziz, 2007). Konsep dari sebuah FSM adalah penting dalam pemrograman karena ia membuat penanganan event secara *explicit* bergantung pada tipe *event* dan pada *state* dari sistem. Sebuah FSM apabila digunakan secara benar bisa memangkas jumlah alur pengeksekusian kode.

FSM

FSM dapat direpresentasikan dalam bentuk grafik berupa *state diagrams /statecharts*. Diagram ini adalah sebuah *graph* yang terhubung dimana *node* melambangkan *state* dan konektor melambangkan transisi *state*.



Gambar 1. Contoh FSM yang direpresentasikan dalam bentuk grafik (Miro, 2003).

Sebuah *state* menangkap aspek-aspek yang relevan dari *history* yang dimiliki oleh sistem secara efesien. Sebuah *state* bisa mengabstrakkan urutan *event* dan menangkap satu *event* yang relevan. Konsep dari *state* ini mengurangi masalah pengidentifikasian bagian yang dieksekusi didalam kode untuk memastikan hanya variabel *state* yang akan diproses dibandingkan banyak variabel, sehingga bisa mengeliminisasi banyak kondisional. Pada umumnya sebagain besar teknik pengimplementasian *state machine* menggunakan "*nested-switch*" (Miro, 2003).

Event

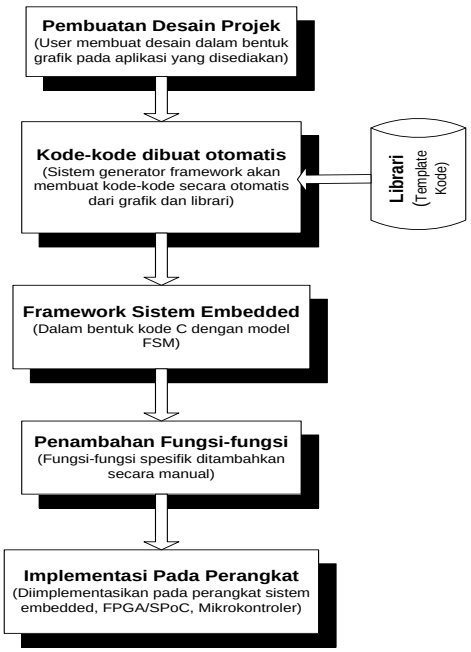
Event adalah sebuah kejadian dalam suatu waktu dan ruang yang mempunyai arti terhadap sebuah sistem. Dalam istilah yang dispesifikasikan dalam UML istilah *event* lebih merujuk ke tipe dari kejadian dibandingkan instansiasi konkret dari suatu kejadian. Saat sebuah *event* telah dibuat, instan sebuah *event* yang melalui sebuah proses siklus hidup bisa terdiri dari tiga tingkat, yaitu: pertama, instan *event* diterima ketika ia sedang menunggu pemrosesan (*event* berada pada *event queue*), kedua, instan *event* dikirimkan ke *state machine*, dimana saat ini ia menjadi *event* yang sedang terpakai, ketiga, instan *event* digunakan ketika *state machine* menyelesaikan pemrosesan.

2. METODOLOGI PENELITIAN

Perangkat generator yang dibuat akan menghasilkan kode-kode framework sistem embedded dimana kode-kode framework ini dihasilkan berdasarkan desain projek dalam bentuk grafik yang telah dibuat. Langkah awal yang dilakukan pada penelitian ini adalah merancang alur kerja sistem perangkat generator dan langkah kedua yang dilakukan merancang arsitektur kode framework yang akan dihasilkan.

Desain alir generator framework

Pada alur kerja sistem ini, sistem akan menyediakan sebuah area kerja kepada user untuk membuat projek desain dalam bentuk grafik. Desain yang dibuat ini bisa disimpan sehingga bisa dilanjutkan pengerjaannya untuk waktu tertentu. Desain projek yang dibuat ini bisa di-*generate* secara otomatis menjadi kode-kode framework dimana pada proses ini sistem akan mengacu pada pada library-librari yang telah disediakan. Librari-librari ini adalah kumpulan kode template yang digunakan sebagai referensi. Ketersedian librari-librari memungkinkan untuk memilih jenis RTOS(*Real Time Operating System*) dan Mikroprosesor yang akan digunakan.

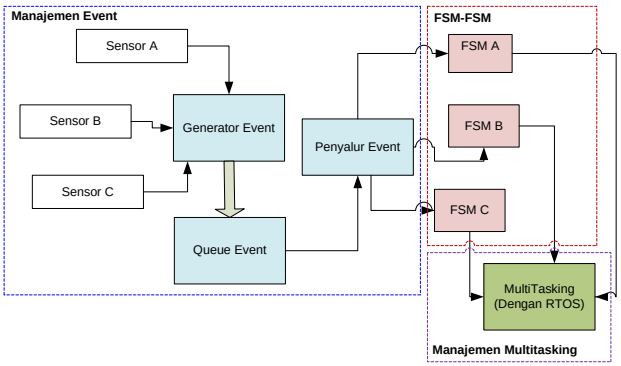


Gambar 2. Desain aliran kerja sistem generator framework

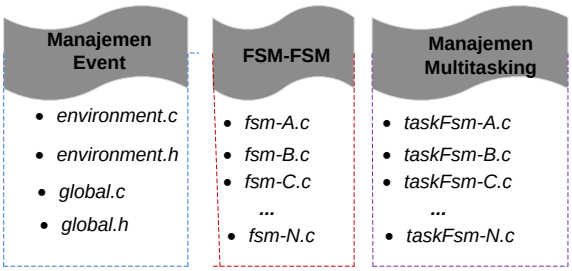
Framework yang dihasilkan dalam kode C dengan model FSM, dimana pada kode framework ini terdapat fungsi-fungsi spesifik yang telah dideklarasikan yang selanjutnya user harus menambahkan isi dari fungsi-fungsi tersebut. Apabila proses penambahan kode-kode secara manual sudah dilakukan maka dapat diimplementasi pada platform yang telah ditentukan.

Arsitektur kode-kode framework

Bagan yang ditunjukkan pada Gambar 3 menjelaskan arsitektur dari kode-kode framework yang akan dihasilkan. Inti dari model framework ini adalah pada blok FSM-FSM sedangkan blok event dan multitasking sebagai modul penunjang. Penambahan fungsi-fungsi spesifik yang akan dilakukan secara manual terletak pada blok FSM-FSM.



Gambar 3. Arsitektur kode-kode framework



Gambar 4. Susunan kode-kode framework

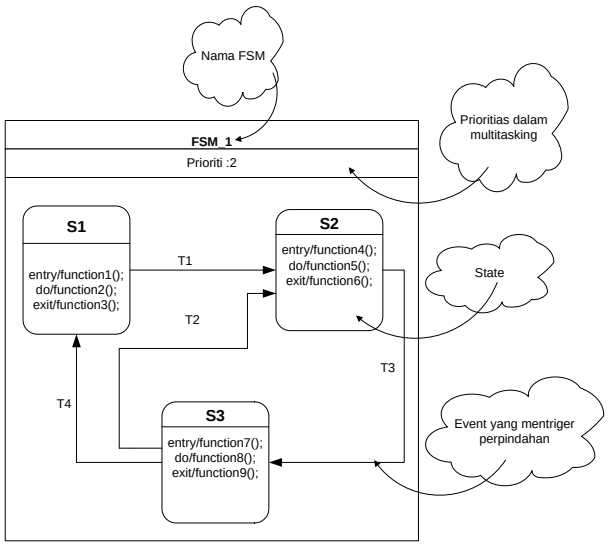
Arsitektur kode-kode yang dihasilkan oleh perangkat generator ini dikelompokkan menjadi tiga seperti yang ditunjukkan pada Gambar 4 yaitu :

1. Manajemen event

   Pada bagian ini terdapat sekumpulan file-file yang berfungsi untuk menangkap *event-event* dari dunia luar seperi sensor, penekanan keypad, dan sebagainya. Selain itu kelompok program file ini juga berfungsi menyalurkan *event-event* ke FSM yang sesuai. *Event-event* ini akan diperlukan dalam suatu FSM sebagai pemicu terjadinya perpindahan *state*. File-file program yang berfungsi sebagai manajemen event ini yaitu: *environment.h, environment.c, global.h, dan global.c*.

2. Kumpulan FSM

   Pada bagian ini terdapat sekumpulan file-file utama yang berfungsi menjalankan tugas utama. Setiap file FSM mempunyai satu tugas, misalkan file fsm pertama digunakan untuk proses pembacaan sensor suhu pertama, fsm kedua untuk pembacaan sensor suhu kedua, fsm ketiga untuk pembacaan tombol. Sebuah FSM ditunjukkan pada Gambar 5 yang terdiri dari tiga *state*, dimana perpindahan state dipicu oleh *event*.



Gambar 5. Contoh FSM

Dimana pada FSM terdapat tiga state yaitu: *S1*, *S2*, dan *S3*. Sedangkan *event* yang menyebakan terjadi perpindahan state yaitu: *T1*, *T2*, *T3*, *T4*. Proses yang bisa dilakukan pada setiap state adalah memanggil fungsi untuk melakukan proses tertentu. Desain dari FSM ini bisa diterjemahkan menjadi kode-kode program, dengan menggunakan konsep *Nested Switch Statement* dan setiap FSM akan menghasilkan satu file kode program (*fsm-x.c*). Berikut ini ditunjukkan contoh kode untuk menterjemahkan FSM.

```
Switch(state){
 case S1 :
  switch(condition){
     case ENTER:
         TRANCODITION(STEADY);
         function1();
     case STEADY:
         function2();
         switch(event){
            case T1:
              nextState = S2;
              TRANCONDITION(EXIT);
            default:
              TRANCONDITION(STEADY);
         }
     case EXIT:
         TRANCONDITION(ENTER);
         function3();
  }

 case S2 :
  switch(condition){
     case ENTER:
         TRANCODITION(STEADY);
         function4();
     case STEADY:
         function5();
         switch(event){
            case T3:
              nextState = S3;
              TRANCONDITION(EXIT);
            default :
              TRANCONDITION(STEADY);
         }
     case EXIT:
         TRANCONDITION(ENTER);
         function6();
  }
}
```

```
case S3 :
  switch(condition){
    case ENTER:
        TRANCODITION(STEADY);
        Function7();
    case STEADY:
        function8();
        switch(event){
           case T2:
             nextState = S2;
             TRANCONDITION(EXIT);
           case T2:
             nextState = S1;
             TRANCONDITION(EXIT);
           default:
             TRANCONDITION(STEADY);
        }
    case EXIT:
        TRANCONDITION(ENTER);
        Function9();
  }
```

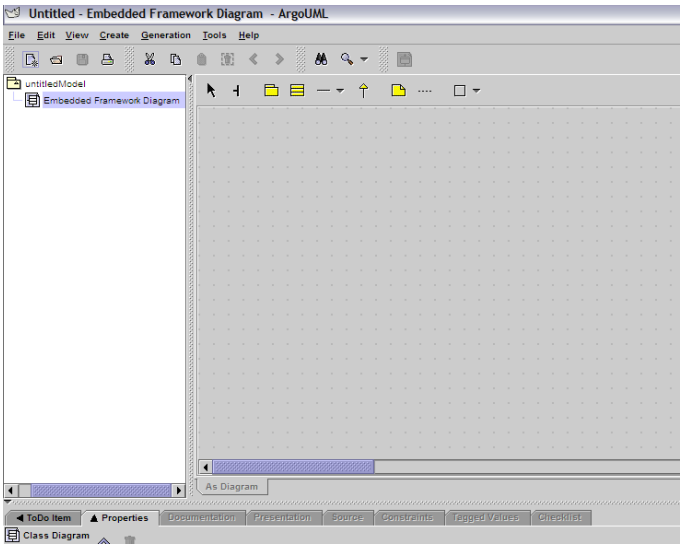
3. Manajemen *Multitasking*

   Bagian ini berfungsi untuk menangani proses multitasking yang menggunakan RTOS (Jean, 1999). Setiap FSM yang mempunyai tanda *priority* berbeda akan dibuat sebuah *task* yang akan memanggil file fsm. Penamaan untuk sebuah objek yang mempunyai *priority* berbeda adalah *taskFsm-A*.c, sedangkan apabila beberapa FSM mempunyai *priority* yang sama akan menggunakan salah satu nama FSM sebagai berikut *taskFsm-AandRelation.c*. Selain itu juga terdapat satu file tambahan *environment_os.h* untuk proses pendeklarasian *task-task.*

Implementasi dengan kustomisasi ArgoUML

Proses pengimputan data pada perangkat generator yang dibuat pada penelitian ini menggunakan grafik, dimana user bisa menggambar desain dari projek yang dibuat dalam bentuk grafik. Sebuah bahasa pemodelan dalam bentuk grafik seperti UML(*Unified Modeling Language*) akan sangat membantu dalam pengembangan, menjelaskan dan mengkomunikasikan sebuah sistem dalam pandangan yang berbeda (Hassan, 2008). Penulis memaanfaatkan opensource ArgoUML sebagai penyedia grafik dan melakukan kustomisasi pada opensource ini. ArgoUML adalah sebuah aplikasi pembuatan diagram UML yang ditulis dalam

bahasa Java dibawah lisensi Open *Source Enclipse License*. ArgoUML merupakan sebuah projek *open source* yang mana kode-kode *open source* nya ditaruh di Tigris.org (Linus, 2007). Dalam bahasa UML sebuah FSM direpresentasikan dengan statechart. Karena pada ArgoUML menyediakan seluruh varian grafik UML, sedangkan dari penelitian ini hanya dibutuhkan *statechart* untuk merepresentasikan FSM, maka dilakukan kustomisasi pada ArgoUML sehingga grafik yang tersedia hanya grafik *statechart*. Kemudian pada penelitian ini melakukan penambahan modul mesin *generator* yang berfungsi untuk membuat kode-kode framework. Karena ArgoUML adalah open source yang menggunakan bahasa Java, maka modul yang ditambahkan berupa kode-kode java. Hasil kustomisasi ArgoUML ditunjukkan pada Gambar 6.



Gambar 6. Hasil kustomisasi ArgoUML

3. HASIL DAN PEMBAHASAN

Untuk menguji perangkat generator yang telah dihasilkan maka dilakukan dua jenis pengujian yaitu: pengujian berdasarkan kegunaan (*fungsional*) dan pengujian dengan sebuah studi kasus. Pada pengujian berdasarkan kegunaan diujikan apakah sistem ini sudah berjalan sesuai dengan yang telah dirancang. Proses ini dilakukan dengan membuat sebuah framework projek sistem embedded

dengan menggunakan perangkat generator kemudian mengecek framework yang dihasilkan apakah sudah sesuai dengan prosedur.

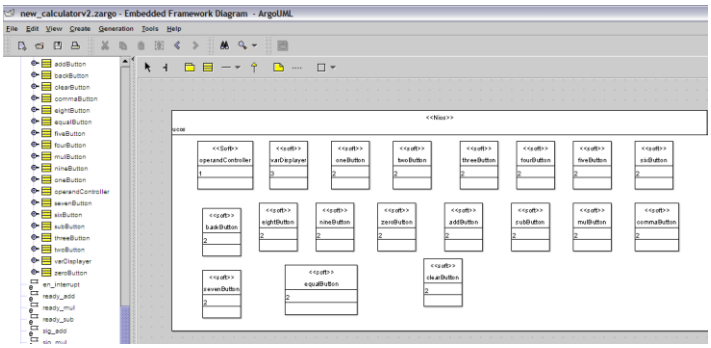
Pada pengujian berdasarkan studi kasus sistem ini diuji apakah mampu digunakan untuk membantu dalam menyelesaikan suatu projek sistem embedded yang bisa diimplementasikan secara nyata. Dimana kasus yang digunakan adalah pembuatan kalkulator yang berbasis prosesor Nios.

Studi kasus pembuatan kalkulator embedded berbasis FPGA

Untuk melakukan uji coba sistem dengan kasus projek yang sebenarnya sehingga dapat diketahui seberapa besar kebermanfaatan perangkat generator ini dalam membantu pembuatan projek sistem embedded maka dibuat sebuah projek kalkulator embedded. Aplikasi kalkulator embedded menggunakan perangkat FPGA Altera ,sebuah monitor dan mouse seperti yang ditunjukkan pada Gambar. Prosesor Nios terpasang pada FPGA dan RTOS yang digunakan adalah ucos. Adapun langkah-langkah dalam membuat projek ini yaitu:

1. Pendeklarasian FSM

   FSM-FSM yang akan dibuatkan diagram nya terlebih dahulu dideklarasikan pada bagian area kerja seperti pada Gambar.



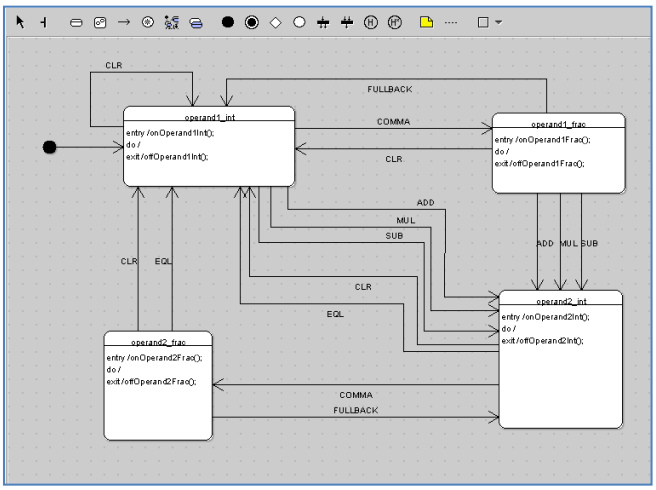
Gambar 7. Pendeklarasian FSM-FSM aplikasi kalkulator embedded

2. Pembuatan diagram FSM

Proses selanjutnya adalah pembuatan diagram untuk setiap FSM yang telah dideklarasikan. Jumlah diagram yang harus dibuat adalah sebanyak 19 FSM, tetapi

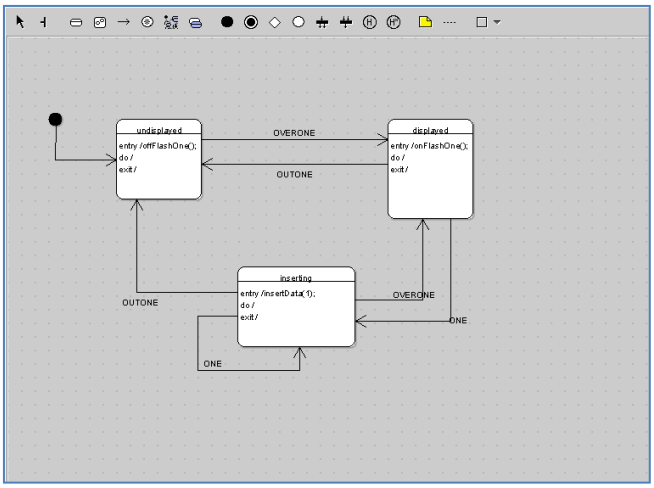
pada tulisan akan dibahas sebanyak dua FSM, yaitu: FSM *OperandController* dan FSM *OneButton.*

FSM *OperandController* ini melakukan kendali dalam menentukan operand-operand yang semestinya harus aktif. Operand-operand yang terdapat pada projek kalkulator ini yaitu: *operand1Int, operand1Frac, operand2Int, operand2Frac*.



Gambar 8. FSM *OperandController*

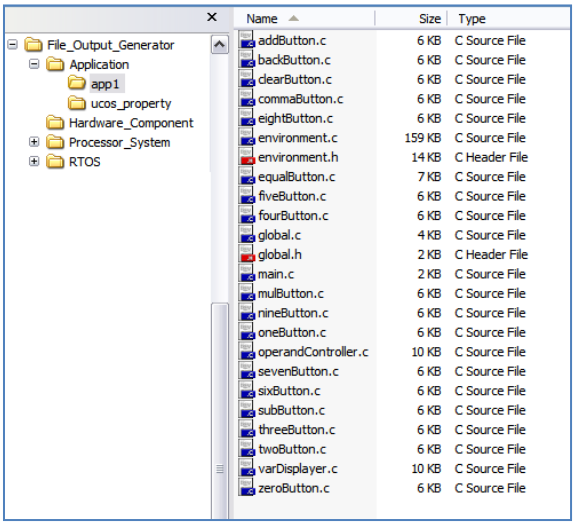
FSM *OneButton* akan melakukan proses yang berhubungan dengan tombol angka satu.



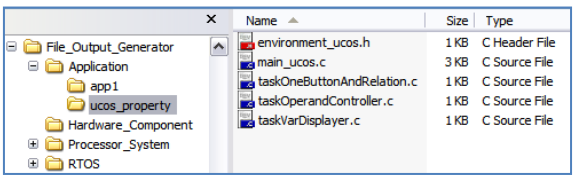
Gambar 9. FSM *OneButton*

3. Kode-kode yang dihasilkan

   Kode-kode framework yang dihasilkan disimpan dalam dua folder yang berbeda. Folder pertama untuk menyimpan file-file yang berkaitan dengan FSM dan manajemen event. Sedangkan folder kedua menyimpan proses multitasking.



Gambar 10. File-file framework FSM



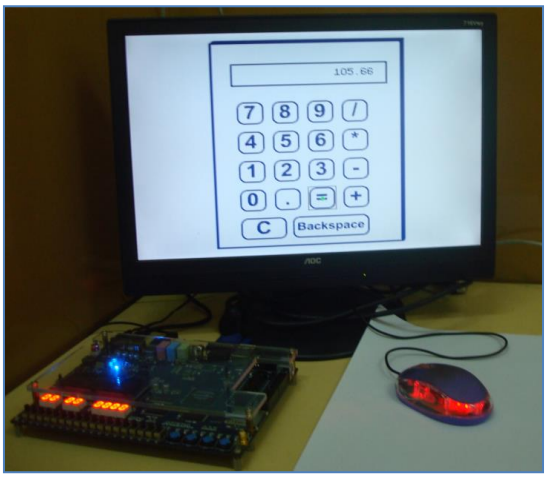
Gambar 11. File-file framework manajemen multitasking

Fungsi-fungsi spesifik kemudian ditambahakan secara manual pada file-file program yang memerlukan.

4. Implementasi dengan FPGA

   Sebelum diimplementasikan ke sbuah alat, terjadi proses penambahan kode-kode pada fungsi-fungsi yang telah dideklarasikan secara manual. Selanjutnya kode program ini dikompilasi dan ditanamkan ke hardware. Pada implementasi ini hardware yang digunakan adalah FPGA. Sebelum FPGA ini dgunakan dilakukan

suatu konfigurasi menjadi SoC (*System On Chip*) dengan prosesor Nios terdapat didalamnya.



Gambar 12. Hasil implementasi kode-kode framework

4. PENUTUP

Perangkat generator framework sistem embedded sangat berguna dalam pembuatan aplikasi sistem embedded. Hal ini dibuktikan dari pengujian yang telah dilakukan dengan sebuah studi kasus menunjukkan terjadi penghematan waktu sebesar 70% dalam pengerjaan sebuah projek aplikasi sistem embedded. Selain itu kode-kode aplikasi sistem embedded yang dihasilkan mudah dibaca karena sudah menggunakan model standar. Pengembangan sistem akan mudah dilakukan karena desain projek dalam bentuk grafik bisa disimpan dan selanjutnya bisa ditambahkan modul-modul FSM yang baru atau menghapusan modul FSM.

Hal yang perlu dikembangkan pada perangkat generator ini adalah mengotomatisasi kode-kode spesifik yang saat ini masih dilakukan secara manual.

DAFTAR PUSTAKA

Abdelaziz G. dan Harald R. 2007. *Adaptation of State/Transition-Based Methods for Embedded System Testing*. World Academy of Science, Engineering and Technology Vol:10

Frank V., Tony G. 2002. *Embedded System Design : A Unified Hardware-Software Introduction*. Wiley, New Jersey.

Hassan G. 2008. *Model-based Software Design of Real-Time Embedded Systems*. IJSE Vol.1, No.1

Jean J. L. 1999. *MicroC/OS-II : The Real-Time Kernel.* R&D Books Lawrence.

Linus T., Markus K., and Michiel V.W. 2007. *Cookbook for Developers of ArgoUML : An introduction to Developing ArgoUML*. University of California.

Miro S. 2003. *Practical UML Statecharts in C/C++ : Event-Driven Programming for Embedded Systems*. Elsevier.

Nam H. L. dan Sung D. C. 2004. *Generating Reduced Finite State Machine from Concurrent Scenarios Using Static Partial Order Method*, Journal of Research and Practice in Information Technology, Vol. 36, No. 3.

Pong P. C. 2011. *Embedded SOPC Design With Nios II Processor and VHDL Examples*. Wiley.