IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 6, Issue 4, Ver. II (Jul. - Aug. 2016), PP 26-32
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Accurate area estimation model for FPGA based Implementation

Rachna Singh[1], Arvind Rajawat[2]
[1](Department of Electronics & Communication Engineering, SISTec-E, Bhopal(M.P),India)
[2](Department of Electronics & Communication Engineering, MANIT, Bhopal(M.P),India)

Abstract: *This paper presents parametric area estimation model for implementation using FPGA's from the Xilinx Spartan 3E family. Accurate estimates of the FPGA resources required provides the system designer important feedback on area which is valuable even during early design iterations. Detailed model for accurately estimating the number of LUT's, block RAMs and 18X18 multipliers for benchmark circuits like FFT8,DCT8 etc.. have been developed. In all cases model coefficients have been derived by using curve fitting analysis. Estimates are conservative, and accurate to within 12% of the post-mapping implementation report. In this paper ,we explain how block resource information is characterized in a MATLAB function. Area estimation in terms of LUT's is with an average error of 6.37% for Spartan 3E.*

Keyword: *Area Estimation, Field Programmable gate array (FPGA), HW/SW Partitioning, High-level synthesis.*

I. Introduction

Field-Programmable Gate Arrays (FPGAs) are becoming increasingly popular as recent trends indicate a faster growth of their transistor density than even general purpose processors. This high logic density plus the field programmability offers a very-lucrative, inexpensive, customized, VLSI implementation platform. One of the most important requirements of an FPGA-based automatic Hardware/Software partitioning tool is to estimate the number of Slices required by the design.

In this paper, a fast and accurate FPGA-based area estimation model is proposed. This tool is used in the context of a HW/SW partitioning tool to get a pre-synthesis estimate of the area of the Hardware part.. The hardware is viewed in this paper as set of modules. For estimating area consumed by a real system of higher complexity, the system is partitioned in terms of these modules.

The outline of the paper is as follows: Section 2 gives an overview of the related work.Section3 discusses the FPGA design methodology. Section 4 discusses the FPGA-based area estimation. Section 5 presents the experiments and results. Finally the paper is concluded in section6.

II. Related Work

The most important challenge in the embedded system design is HW/SW partitioning. Finding an optimal partition is hard because of the large number and different characteristics of the components that have to be considered. For the Hardware implementation, the most important characteristics to be considered are: hardware area, delay, latency, and power consumption. A quick and accurate estimation of such characteristics is of a paramount importance to guide the decision making process. An excellent survey of the hardware characteristics estimation techniques is presented in [1].

For the area estimation, some techniques are tailored for certain partitioning schemes [2-3]. Such schemes are suitable for iterative partitioning algorithms where the area of the hardware part is updated after adding (removing) any component to (from) the hardware side. Other techniques estimate the hardware area independently from the partitioning process.

Area estimation for different input description languages is widely studied (C [1, 4-5], SA-C [6], SystemC [7], MATLAB [8], Simulink [9], VHDL [10] …etc). Most of the published work performs a transformation step to express the input description into an Intermediate Representation (IR) such as Trimaran IR [4], Control Data Flow Graph (CDFG) [5], and VHDL AST [10], and then, apply the estimation process on the intermediate format.

Most of the techniques detect the resource sharing opportunities through scheduling [10-11], according to the complexity of the design [12], or by using a derived resource utilization formula [6].As sharing resources implies multiplexers instantiation and as multiplexers are always considered high cost on FPGAs, therefore,in FPGA-based high-level synthesis the number of inputs per multiplexers is restricted [13].

Regarding the target technology, current approaches target either ASIC-based designs [11-12] or FPGA-based designs [4-10, 14-20].FPGA-based area estimators either incorporate a physical model for the FPGA and estimate the area by performing actual mapping [14],by using modelling equations of the FPGA

functional resources [4-10, 15-17], or by building a large database for all possible resources configurations [18-19].As the routing of signals between resources can consume extra area, this area is difficult to determine prior to placement and routing. Fortunately, this routing area does not usually constitute a very large fraction of the overall area for the small and medium designs [6]. In Ref. [10] the routing effect is included as a constant scale factor. Most of the above research focus on the datapath area estimation and ignore control logic; others integrate control logic and datapath estimation in one tool flow [4, 19], and others concentrate completely on estimating area usage of control logic only [20].

III. Research Methodology

3.1 Fpga Design Methodologies

FPGA design methodology as shown in fig. 1 is used as a guideline for the hardware realization of algorithms. The first step in FPGA design methodology is to capture the algorithm to be implemented on FPGA using hardware description languages (HDLs) or schematic depending on the complexity of the design.

After specifying the design using HDLs or Schematic, the designer needs to validate the logical correctness of the design. This is performed using functional or behavioralsimulation. Designers usually go through this step right after they finish the coding and logic synthesis. Logic synthesis converts HDL or schematic-based design into a netlistof actual gates/blocks specified in FPGA devices. After logic synthesis, technology mapping is done.

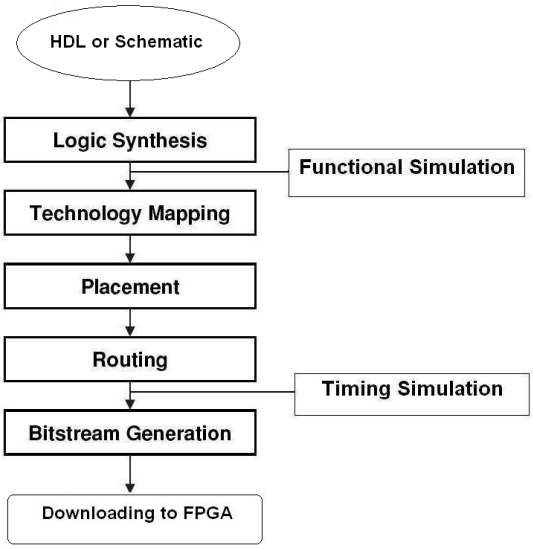

Figure 1. FPGA Design Methodology

In this step, the tool transforms a netlist of technology independent logic gates intoone consisting of logic cells and input/output blocks (IOBs) in the target FPGA architectures [12,13].

Placement which follows technology mapping selects the optimal position for each block in a circuit. The basic goal of an FPGA placement is to locate functional blocks such that then interconnect required to route the signals between them is minimized. A good placement is extremely important forFPGA designs. It directly affects the routability and the performance of a design on FPGA [14]. A poor placement will lead to lower maximum operating speed and increased power consumption. FPGA placement algorithms can be broadly classified as routability-driven and timing-driven [15].

The main objective of routability-driven algorithms is to create a placement that minimizes the total interconnect

required. In addition to optimizing for routability, timing algorithms use timing analysis to identify critical paths and/or connections and optimize the delay of those connections.

Routing is the last step in the design methodology prior to generating the bitstream to program the FPGA [16-18]. FPGA routing is a tedious process because it has to use only the prefabricated routing resources such as wire segments, programmable switches and multiplexers [17]. Hence it is a challenging task to achieve 100% routability in FPGAs.

After placement and routing, timing simulation is performed to validate the logical correctness of the design taking into account the delays of the FPGA device. Power consumed by a design is further estimated by doing the poweranalysis such as XPower and PowerPlay tools used in XilinxISE and Altera Quartus II tools respectively.

The final step in the FPGA design methodology is bitstream generation. It takes the mapped, placed and routed design as input and generates the necessary bitstream to program the logic and interconnects to implement the intended logic design and layout on the target device.

3.2 Resource Estimation Methodology:

This section presents our models to quickly and accurately estimate the area of implementation on Xillinx Spartan 3E FPGA. The hardware resources for a design are of course provided in the post technology mapping report. However this information becomes available only after compilation, synthesis and mapping stages.This entire process can take minutes or even hours, depending on the size of the system.

There are several straightforward approaches to pre-netlisting resource estimation.One is to build a database listing the resources given all the possible combination of a particular block.

Our model uses the methodology given below(Fig.2) as a guideline for the hardware implementation. The first step in FPGA design methodology is to capture the algorithm to be implemented on FPGA using behavioral VHDL description.Then the VHDL code is transformed to a CDFG intermediate format.The CDFG generated has been parsed and useful information for the operations such as; operation type, input bit width etc…has been stored.

A library is created for each basic operations like logic gates, adder, multiplier, comparator etc…A formula for each operation from the basic idea of the hardware circuit design is derived.The bit width of the operations is varied and the corresponding VHDL instances is created.Synthesize the VHDL files and record the real values reported by the synthesis tool A curve fitting analysis on the real values is done to generate a closed formulae.A library of area estimation models for these low granularity modules has been built.For estimating area consumed by a real system of higher complexity, the system is partitioned in terms of these functional units.

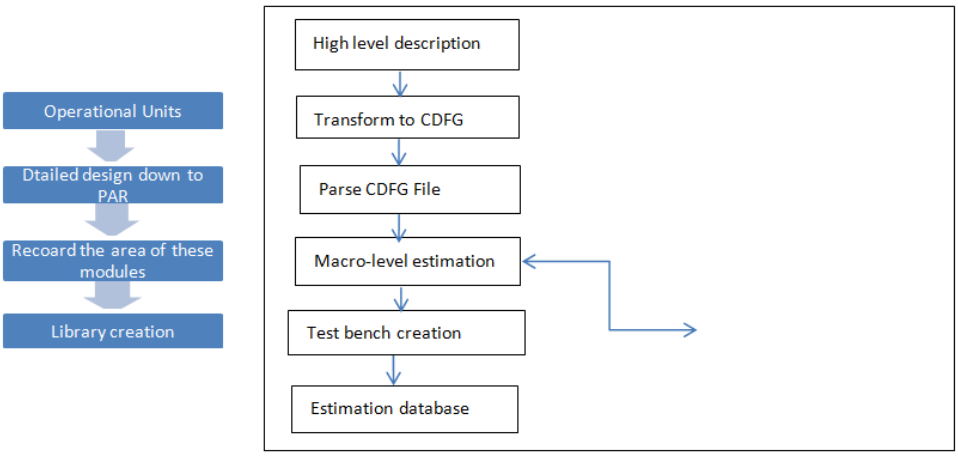

Figure.2 Reasearch Methodology Flow Chart

IV. FPGA Based Area Estimation

In the last few years, FPGA vendors started to include a microprocessor core inside their FPGA's, the processors are included as a hardwired cores(Power PC 405 processor in Xilinx Virtex-II pro and Virtex-4 devices(27) as well as ARM 922T processor in Altera Excalibur devices(28) or soft cores(micro-blaze core in Xilinx(27) as well as NIOS-II core in Altera(28).Such inclusion has made FPGA's a perfect platform for embedded systems since it facilities the design process. The interface synthesis ,co-debugging and co-verification of the design to be partitioned.For the above reason, we have selected FPGA's as our platform. Even though our tool is specific for the Xilinx 4 input LUT- based FPGA, the approach could be easily adapted for use with a variety of other FPGA's.

Most of the Xilinx FPGA devices include arrays of slices that represent the finest resource unit. Each slices contain a 4-input logic functions and has its own programmable register.Note that due to high regularity of the FPGA device, the final resource usage of most operations are very predictable and their estimation function can be expressed in closed form equations. For example the resource usage ie slices of an adder/subtracter is equivalent to the output's bit-width.

In this paper to estimate the area we apply a methodology similar to that of[13].In general , given the target FPGA architecture , the final area of a functional unit and/or a multiplexer are largely determined by the total number of input operands and the precision ie, bit-width of the calculation performed by the functional unit.

The difference between our method and that of [13] is that we target Spartan 3E FPGA rather than cyclone FPGA's. Our area estimation methodology goes as follows: We consider small to medium granularity modules (Adders, multipliers, registers, counters etc..).These modules are named as operational units. We carry out detailed design down to the place & route (PAR) phase for these modules. We then record the areas of these modules as function of their global parameters (number of inputs.....etc).We then derive phenomenological behavioural area estimation models of these units as a function of their global parameters using curve fitting approach (MATLAB7). In effect, we build a library of area estimation models for these low granularity modules. To estimate the area consumed of a real system with a higher complexity, the system is partitioned in terms of these functional units, and the area consumed is estimated in terms of LUTs.

V. Experiments And Results

A set of behavioral VHDL benchmarks obtained from [22] are used to test our tool. The set is selected according to the granularity suitable for our Hw/Sw partitioning approaches in terms of LUT's.

5.1 Library Creation: For a design small to medium granularity modules have been considered. Then the detailed design down to place and Route phase for these modules is done and areas of these modules as function of their global parameters is recorded. Mathematical model of these units has been derived using curve fitting approach(MATLAB).A library of area estimation models for these modules is build.Table 1 shows the area of modules in term of LUT's for FPGA(Spartan-3E).
Similarly we build the library for other modules also like Adder, comparator, shift register , divider etc.

5.2 Area Estimation for Basic Modules

The models for estimating areas of modules like adder, multiplier, counter etc..in terms of number of LUT's ,number of block RAMs and number of 18X18 multipliers has been recorded as function of their global parameters(bit width, no. of inputs etc..).Area estimation model of these units has been derived using curve fitting approach (MATLAB). We vary the precision for each operational unit(upto 64 bits).The maximum no of inputs is limited to two inputs except for the multiplexer's where we vary the number of inputs for upto 32 inputs.Eachoperation has two inputs with bit-widths N and M. In general the operational units could be divided into 2 main operation categories:

Table1.Area in LUT's of Spartan-3E (XC3S500E)for operational units

S.NO	OPERATIONAL UNIT	I/P BIT WIDTH	O/P BIT WIDTH	NO OF 4 I/P LUT'S	NO OF SLICES	NO OF IOB'S	NO OF MULTIPIERS
1	AND	2	2	2	1	6	-
2	AND1	4	4	4	2	12	-
3	AND2	8	8	8	4	24	-
4	AND3	16	16	16	9	48	-
5	AND4	32	32	32	18	96	-
6	AND5	64	64	64	37	192	
7	AND6	128	128	128	74	384	
8	MULTIPLIER1	2	4	3	2	8	0
9	MULTIPLIER2	4	8	-	0	16	1
10	MULTIPLIER3	8	16	-	0	32	1
11	MULTIPLIER4	16	32	0	0	64	1
12	MULTIPLIER5	32	64	108	55	128	4
13	MULTIPLIER6	64	128	608	307	256	16
14	MULTIPLIER7	128	256	16916	8471	512	Area used more than 100%

5.2.1Simple Operations:

In this category, a closed form equation could represent the area usage of the operations. The area estimation functions for the simple logic and arithmetic units are listed below. Note that each operation has a maximum of two inputs with bit width N & M.

i) Bit wise logic operations: This contains all bit wise logic operations like AND, OR,NAND NOR etc….
Area= Max(N,M)

ii) Adder/Subtracter: The area in terms of LUT's is given below
Area= Max(N,M)+1

iii) Signal assignment, Register:No of LUT's consumed for the signal assignment or a register depends on its size ie, for a N bit register the no of LUT's consumed will be N.

5.2.2Optimizable Operations:

This category contains the operations that have optimization and/or transformation chances where the area could have different formulation in accordance.The main operations that fall in this category are:the shifter , the divider, the comparator and the multiplier.

i)Shifter: For the sifter, if the shifting distance is variable, ie the input data could be shifted in one direction with a maximum distance equal's the input bit width,the area usage in LUT's:

Area=$[0.9826*N^2+0.8657*N]$

On the other hand, if the shifting distance is constant, the area usage is N LUT's.

ii)Divider: For the divider, if the denominator value (M) is a power of two, the division is then converted into shift right operation and the area usage equals the nominators width (N),otherwise ,equations are used to get area utilization for $N \geq M$ & $N<M$ respectively.

Area=$[M^2 + 6.588*M-7+ 2*(N-M)*(M+1)]$

Area=$[N^2 + 6.588*N-7+ (M+N)*(M-N)-8]$

iii)Comparator:For the comparator area usage depends on packing each four inputs in one LUT .

Area=[2.524*N+0.431]

iv)Multiplier:For the multiplier as some devices include a dedicated hardwired multiplier,the area usage in this is different . For example for Spartan 3E devices, up to 18X18 bit multiplication could be configured inside the dedicated multiplier.When the target multiplier is less than 18X18 size, it can fit into one embedded multiplier.Otherwise multiple embedded multipliers are needed , each of which generates a partial product followed by adder logic to sum the partial products into the final output.

Finally for the general multiplier,the area usage when the input width are same is given below:

Area=$[N^2+3.8657*N-1]$

Number of 18X18 multipliers:The multiplier18 block supports two data input ports :18 bit signed or 17 bit unsigned. The number of multiplier18 blocks to implement the product of M and N is given by

$M_{18}(M,N)=[(M/17)*(N/17)]$

v)v) No of Block RAMs:The Spartan 3E has a total of 36 18-Kb block select RAM.In order to store a table with N entries & M bits per entry a BRAM configuration with greater than N entries is selected and then multiple BRAMs of that type are utilized to accommodate the M bits per entry.

5.3Experimental Result:

In this section we compare the area using our model and those obtained by the actual synthesis for a set of benchmark circuits. Two groups of experiments are performed ,one is for implementations that require BRAMs such as look up table based function evaluations and the other is for implementation that require no BRAM such as ID Discrete Fourier Transform, ID Discrete Cosine Transform and FIR filter.

5.3.1)Lookup Table based Implementation:

For this set as shown in table no we write detailed VHDL RTL codes of the three implementation. This set includes a simple mathematical operation given by, y=[(a+b)*(c+d)] (2ADD) , a similar design where y=[(a*b)+(c*d)] (2MUL) ,a differential equation solver(DiffEq) and finally an iterative Square Root calculator(Sqrt) to test the estimation of designs with loops.

The error term is defined as:

$$Error = \frac{Estimated\ Area - Synthesis\ Area}{Synthesis\ Area}$$

Table .2 Comparision of the proposed resource estimation tool(new tool) with map-report (Xillinx(Spartan 3E)) for LUT based Implementation

Design	Our tool	Xillinx (Spartan 3E)	Error
2MUL	141	155	-9.03%
2ADD	160	151	5.96%
DiffEq	573	509	12.57%
Sqrt	715	675	5.92%

The comparison results are shown in Table . For the examples shown here , the average error is 8.37% for the no of LUT's. The result of block RAM and 18X18-bit multiplier are not shown because there is no mismatch between the estimated and synthesized results , as expected.

5.3.2Transform Computations:

In this section, we choose two algorithms which are widely used in digital signal processing: the 8-point FFT(FFT8) , the 8-point DCT(DCT8),FIR & IIR.

Table .3 Comparision of the proposed resource estimation tool(new tool) with map-report (Xillinx(Spartan 3E)) for Transform computations

Design	Our tool	Xillinx (Spartan 3E)	Error
FFT(8)	2913	2856	1.99%
DCT(8)	801	780	2.69%
FIR	470	448	4.91%
IIR	478	443	7.9%

For the FIR,IIR ,FFT8 and DCT8 configuration ,the average error is 4.3725%for the no of LUT's. Note that the error here is lower than the LUT based implementations presented in table . This is because the estimate for the number of LUT's is a lot more accurate when the design is larger and a significant portion of the FPGA slices is utilized

VI. Conclusion

In this paper a fast area estimation model for FPGA based implementation is presented. The model was developed to speed up the algorithm-architecture co-exploration for systems that have to meet area requirements. The model consists of parameterized functions that estimate the resources in terms of LUT's. The models have been derived using curve fitting tool(MATLAB).Testing this estimation tool on designs showed that this tool is also accurate Area estimation in terms of LUT's are with an average error of 6.37 % .Further more, the time required to generate these estimates is of the order of microseconds,as compared to minutes or even hours for designs which undergo actual synthesis followed by Placement &Routing.While all the results presented in this paper are for Xillinx Spartan 3E FPGA, the method can be applied to many other FPGA platforms as well.

Reference

[1]. Shi C., Hwang, J., McMillan, S., Root, A., and Singh, V.,"A System Level Resource Estimation Tool for FPGAs",International Conference on Field Programmable Logic andApplications (FPL),LHCS 3203,pp.423-433,2004.

[2]. RoelMeeuws, "A Quantitative model for Hardware/Software partitioning," MSc thesis, Delft University of Technology, Delft,Netherland,Tech.Rep.RCOSY DES.6392,pp 735-739,2007.

[3]. V. Srinivasan, S. Govindarajan, and R. Vemuri, "Fine-grained and coarse-grained behavioral partitioning with effective utilization of memory and design space exploration for multi-FPGA architectures," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 9, no. 1, pp. 140–158, 2001.

[4]. F. Vahid and D. D. Gajski, "Incremental hardware estimation during hardware/software functional partitioning," in Readings in hardware/software co-design, G. De Micheli, R. Ernest, and W.Wolf (eds.), Morgan Kaufmann, pp. 516–521, 2002.

[5]. L. Yan, T. Srikanthan, and N. Gang, "Area and Delay Estimation for FPGA Implementation of Coarse-Grained Reconfigurable Architectures," LCTES, Ottawa, Ontario, Canada,pp.182–188, 2006.

[6]. R. Enzler, T. Jeger, D. Cottet, and G. Troster, "High level area and performance estimation of hardware building blocks on FPGAs," FPL 2000, Villach, Austria, pp. 525–534, 2000.

[7]. D. Kulkarni, Walid A. Najjar, R. Rinker and F. J. Kurdahi,"Compile-Time Area Estimation for LUT-Based FPGAs," ACMTODAES, Vol. 11, No. 1, pp. 104–122, 2006.

[8]. P Bjureus, M. Millberg, and A. Jantsch, "FPGA resource and timing estimation from MATLAB execution traces," CODES 2002,Estes Park, Colorado, USA, pp. 31–36, 2002.

[9]. L. M. Reyneri, F. Cucinotta, A. Serra, and L. Lavagno, "A hardware/software co-design flow and IP library based on Simulink," DAC '01, Las Vegas, Nevada, USA, pp. 593 - 598,2001.

[10]. A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee,"Accurate area and delay estimators for FPGAs,"DATE'02, Paris,France, pp. 862-869, 2002.

[11]. Peter A. Milder, Mohammad Ahmad, James C. Hoe, and Markus P¨uschel, "Fast and Accurate Resource Estimation of Automatically Generated Custom DFT IP Cores"Proc. ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Monterey, California, USA.pp. 211-220, 2006.

[12]. Paul Schumacher and PradipJha, "Fast and Accurate Resource Estimation of RTL-based designs targeting FPGA's",International conference on field programmable logic &applications,SanJose,CA,pp 59-63,2008.

[13]. M .B. Abdelhalim and S. E. -D. Habib, "Fast FPGA-based area and latency estimation for a novel hardware/software partitioning scheme"the 2nd intl.IEEE Design and Test Workshop,IDT07,Cairo,Egypt,pp 775-779,2008

[14]. F. Vahid, T. Dm Le, and Yu-Chin Hsu, "Functional Partitioning Improvements over Structural Partitioning for Packaging Constraints and Synthesis: Tool Performance," ACM TODAES, Vol. 3, No. 2, pp. 181–208, 1998.

[15]. C. Menn, O. Bringmann, and W. Rosenstiel, "Controller Estimation for FPGA Target Architectures during High-Level Synthesis," ISSS'02, Kyoto, Japan, pp. 56-61, 2002.

[16]. R. L. Ernst and J. Henkel, "High-level estimation techniques for usage in hardware/software co-design," ASPDAC '98, Yokohama,Japan, pp. 353–360, 1998

[17]. M. Nemani and F. N. Najm, "High-level area and power estimation for VLSI circuits," ICCAD'97, San Jose, California,USA, pp. 114–119, 1997

[18]. Pablo González de Aledo Marugán, Javier González-Bayón and Pablo Sánchez Espeso,"Hardware performance estimation by Dynamic Scheduling" ;in Proc. FDL, 2011, pp.1-6.

[19]. Michael Kunz, Martin Kumm, Martin Heide, Peter Zipf, "Area Estimation of Look-Up Table Based Fixed-Point Computations on the example of a Real-Time High Dynamic Range Imaging System"In 22nd International conference on Field Programmable Logic and Applications(FPL), Oslo,Norway,pp. 591-594,Aug 29-31 2012.

[20]. XiaoxiaNiu,YanxiaWu,Bowei Zhang, "Rapid FPGA-based Delay Estimation for the Hardware/Software Partitioning" Journal of networks,vol.8,pp 1183-1190,5 May 2013.

[21]. XiaoxiaNiu,YanxiaWu,BoweiZhang,GuochangGu,Guoyin ZHANG " Auto Estimation Model of FPGA based Delay for the Hardware/Software Partitioning"Journals of Computational Information system,vol.9,pp 6767-6774,1 September 2013.
[22]. RajendraPatel,A.Rajawat"Recent trends in embedded system software performance estimation"IN Design Automation for Embedded System(Springer)Volume 17, Issue 1,pp 193-213, March 2013.
[23]. CDFG Toolset [Online @ http://poppy.snu.ac.kr/CDFG]