6-1-2010

Case Study of Finite Resource Optimization in FPGA Using Genetic Algorithm

JingXia Wang
Shenzhen Polytechnic

Sin Ming Loo
Boise State University

Case Study of Finite Resource Optimization in FPGA using Genetic Algorithm[1]

JingXia Wang*
Shenzhen Polytechnic, ShenZhen, GuangDong, 518055, PRC

Sin Ming Loo†
Boise State University, Boise, ID 83725, USA

Abstract

Modern Field-Programmable Gate Arrays (FPGAs) are becoming very popular in embedded systems and high performance applications. FPGA has benefited from the shrinking of transistor feature size, which allows more on-chip reconfigurable (e.g., memories and look-up tables) and routing resources available. Unfortunately, the amount of reconfigurable resources in a FPGA is fixed and limited. This paper investigates the mapping scheme of the applications in a FPGA by utilizing sequential processing (e.g., Altera Nios II or Xilinx Microblaze, using C programming language) and task specific hardware (using hardware description language). Genetic Algorithm is used in this study. We found that placing sequential processor cores into FPGA can improve the resource utilization efficiency and achieve acceptable system performance. In this paper, three cases were studied to determine the trade-off between resource optimization and system performance.

Key Words: FPGA, resource utilization, genetic algorithm, scheduling.

1 Introduction

In recent years, Field-Programmable Gate Array (FPGA) has gained popularity in the digital integrated circuit market, specifically in high-performance embedded applications. One of the most significant features of FPGAs is that designers can configure them to implement complex hardware in the field.

With the improvement of integrated circuit technology, very large logical structures are allowed to reside in a single FPGA chip [5]. Not only the hardware function units (implemented using hardware description language) can be placed and routed into FPGAs, embedded processors can also be configured into FPGAs. There are two types of embedded Intellectual Property (IP) processor cores: hard and soft cores. Hard cores are physical manifestations (built into the chip by the designer and foundry) of the IP design. Soft cores, which are more portable and flexible than hard cores, are logical existence as integrated circuit netlist or hardware description language code. In general, an IP-based processor core, such as Xilinx Microblaze or Altera Nios II, is much more flexible than the general hardware logic. Once a processor is implemented in the FPGA, it can be reused many times in a time-shared manner. Unfortunately, the sequential operations limit the system performance [1].

It is important to note that FPGA resources are limited. When the hardware resources required by a task are more than the available resources in a single FPGA chip, generally, it is impossible to realize this system with the given FPGA. We can configure the processor IP cores into FPGAs with the application specific hardware. This allows us to implement some portions of the design using C programming and the rest are implemented using hardware description language. For example, we can implement a task system, which consumes 6674 Configurable Logic Blocks (CLBs), using a single FPGA chip with just 1920 CLBs by placing one soft processor IP core into the FPGA. This allows the finite resources in a FPGA to be used optimally and efficiently. More than one processor IP cores may be used in a single FPGA to implement the complex applications. However, it is noted that the processor IP cores consume both the hardware logic (LUTs) and block RAM resources of FPGAs. The number of processor IP cores integrated into FPGAs is limited by the finite hardware resources of FPGA. We found that, it was not good to put the maximum number processor IP cores into a FPGA because the sequential operations in the processor will restrict the system performance too much. In this paper, three different cases are studied with FPGAs Finite Resource Optimization Analysis Model in order to obtain the optimal FPGA finite resources utilization scheme.

This paper is organized as follows. An overview of FPGA architecture is described in Section 2. Section 3 presents FPGAs Finite Resource Optimization Analysis Model with Genetic Algorithm. Sections 4 and 5 provide a computational complexity analysis with an example. Simulations setup and results are presented in Section 6. The conclusion is presented in Section 7.

2 FPGA Architectures

The basic architecture of FPGAs consists of an array of logic

[1] A subset of this paper was published at 2009 Word Summit on Genetic and Evolutionary Computation Conference, June 12-14, 2009, Shanghai, China.

* Department of Electrical Engineering, XiLi Lake, NanShan District. Email: ljwjxlyr2005@yahoo.com.
† Department of Electrical and Computer Engineering. Email: smloo@boisestate.edu.

blocks, programmable interconnect, and I/O blocks. A logic block which includes a fixed number of LUTs and flip-flops is called a configurable logic block (CLB) or a logic array block (LAB). In this paper, CLB is used. Programmable interconnect joins these logic blocks to provide the required interconnections. I/O block is a pin level interface circuit, which provides the interface between package pins and the internal configurable logic.

With the development of micro-electrical technology, the architecture of modern FPGAs is more complicated than before. It is composed of more resource elements, such as embedded memory blocks (bRAMs), multipliers, and even processor IP cores. Figure 1 shows the generic architecture overview of Xilinx Virtex II-Pro FPGA [5]. Xilinx Virtex-II Pro FPGA includes an array of CLBs, IOB, Multipliers, Block RAM, Embedded RocketIO, and two processors (IBM PowerPCTM405) [5].

The embedded IBM PowerPC 405 is a 32-bit high performance and low power RISC hard processor core. It is fully compliant with the 32-bit implementations of the PowerPC User Instruction Set Architecture (ISA) and can be reprogrammed with the other resources together during the FPGA configuration. Xilinx Virtex-4 is the newest generation with more high-speed I/O technology and digital signal processing features. With these modern FPGAs, it is possible to configure multiple soft processors into a single FPGA device by designers.

The processor in FPGAs provides incomparable vast flexibility for the trade-off between software and hardware. In general, the number of soft processors that can fit into a FPGA is only limited by the resource of the FPGAs.

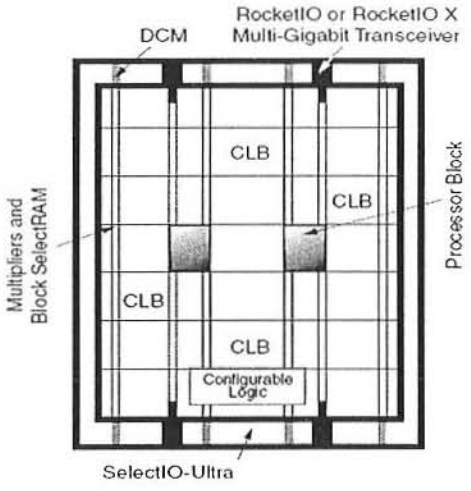


Figure 1: The generic architecture overview of Xilinx Virtex-II Pro FPGA

3 FPGA Finite Resource Optimization Analysis Model with Genetic Algorithm

The FPGA Finite Resource Optimization Analysis Model is shown in Figure 2.

It is used to describe the major elements of the FPGA

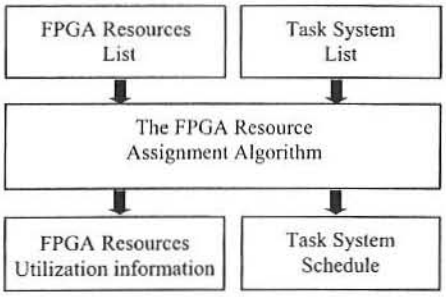


Figure 2: FPGA finite resource optimization analysis model

resources assignment algorithm. The nucleus of this model is the FPGA resources assignment algorithm based on Genetic Algorithm. One of the inputs to the algorithm is the FPGA resources list, which includes the number of CLBs for a given FPGA device, the number of soft processors to be integrated into the given FPGA, and the number of CLBs and the size of the bRAMs that each soft processor employs. Another input is the system information list, which gives the detailed information of the application (number of tasks and how these tasks are related). The FPGA resources assignment algorithm generates the FPGAs resources utilization information and the task system schedule. The former presents the percentage of the specific FPGAs hardware utilization (CLBs) for the given task system. The latter shows the execution time (unit time) of the task system. The shorter the schedule length is, the more desirable the solutions are.

3.1 FPGA Resources

There is a number of hardware resources in a single FPGA chip, including CLBs, IOBs, bRAMs, special logics (e.g., multiplier and digital signal processing block), and routing resources. A soft processor placed and routed in a FPGA will use a fixed amount of CLBs and bRAM. For example, it has been found through experiments that Xilinx Microblaze will use about 400-500 CLBs. In this paper, the number of CLBs in a FPGA is considered as the hardware resources restrictions. In addition, the number of bRAMs, which are used for internal program and data storage of the configured soft processors, present the limitation of the tasks that are assigned to the soft processors. Thus, FPGA resources list includes the number of the CLBs where the hardware applications can be assigned to and the size of memories which store the program and data for the soft processor.

3.2 Task System

Task system is a model of the application. It is decomposed into a set of tasks which can be executed as software processes on a soft processor core, or as hardware functions (implemented using hardware description language) within a FPGA. When the task system is created, we define the execution time for the software and hardware respectively. A sample task system data is shown as Table 1 [2]. We use a directed acyclic graph (DAG) to present a task system [1]. A sample task system DAG is shown as Figure 3 [2].

Table 1: A sample task system resources

Task name	Soft Processor		FPGA Hardware Logic	
	Execution time (unit time)	Program Memory required (Bytes)	Execution time (unit time)	FPGA Resources Required (CLBs)
Task 1	74,640	184	2,560	32,220
Task 2	5,836,560	19,240	208,400	30,509
Task 3	34,200	22,348	400	30,509

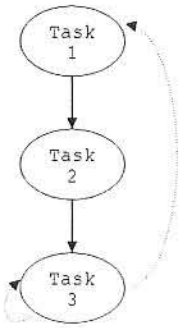


Figure 3: A sample task system DAG

3.3 FPGA Resource Utilization Information

One of the outputs of FPGA resources assignment algorithm is FPGA resource utilization report. It describes the allocation scheme for every task, whether it is in the processor or task specific logic, and the percentage of the utilization of CLBs for a FPGA device.

3.4 Task System Schedule

Another output of FPGA resources assignment algorithm is task system schedule. This report gives the execution time of the task system and the order of execution of the tasks within each soft processor in a FPGA chip. The execution time of the task system determines the system performance.

3.5 The FPGA Resources Assignment Algorithm—Genetic Algorithm

Genetic Algorithm (GA) based on Darwinian natural evolution and selection is a search technique for approximate solutions to optimization problems. The basic operations of GA include initialization, evaluation, selection, reproduction, and termination [4]. Starting from an initial population (list scheduling is used), a population is randomly initialized with tasks assignment using the available hardware resources or soft processors. The individual member of the population is expressed by a separate data structure as candidate. For each candidate, the schedule length is the fitness function. The selection of candidates for the subsequent generation is based on their fitness function. A set of parents with the best genetic information (better schedule length) are selected to breed the offspring. A roulette-wheel-style selection is used in this process. The reproduction process creates the next generation population through two genetic operations: crossover and mutation. A single-point crossover technique is used for the crossover process. We randomly select a location on the chromosome structure as the crossover point. The new candidate is generated by replicating and combining two parent candidates at the chosen crossover point. One is from the first parent chromosome above the crossover point, and the other is from the second parent chromosome below the crossover point. The mutation operation will take place according to a given probability of mutation parameter. The task assignments are selected at random for mutation process when the mutation occurs. There is an equal probability that the genes are chosen from either parent when the mutation does not occur [1]. The evaluation, selection, and reproduction processes are repeated until a termination condition has been reached. Figure 4 depicts this Genetic Algorithm process.

4 Computational Complexity of the FPGA Resources Assignment Algorithm

In this section, we discuss the time complexity and efficiency of the FPGA resources assignment algorithm. There are many factors that will influence the GA efficiency, including the population encoding, fitness function, population size, the probability of mutation, and the termination condition. In this paper, the following parameters are considered:

p: the size of population in GA
i: the number of evolution generation
pm: the probability of mutation
s: the number of soft processors to be configured into FPGA
k: the number of tasks in the task system to be assigned

To determine the complexity of this GA and measure the efficiency, tests were carried out using a computer system with Pentium 4 3GHz processor 1GByte memory running CentOS 4.3 Kernel 2.6.9. Table 2 and Figure 5 show the execution time of GA with the different parameters.

The curve in Figure 5(a) shows the time complexity of GA for parameter p is O(p2). From Figure 5(b), the approximate straight lines show the linear characteristic and the time complexity of GA for parameter i is O(i). Both Figure 5(c) and Figure 5(d) show the characteristic of the log2n (n is a positive integer). So, the time complexity of GA for parameter pm and s is O(log2pm) and O(log2s) respectively.

According to the computational complexity theory, we can know that the time complexity of GA is O(p2). In the algorithm, there is a sorting operation that sorts all candidates in a population with a bubble sorting algorithm for crossover and mutation operations. The time complexity of candidates sorting operation is O(p2) [6]. The sorting operation is the most complex process in this GA implementation. So, we can use the time complexity of this sorting operation to present the time complexity of GA. On the other hand, the population size

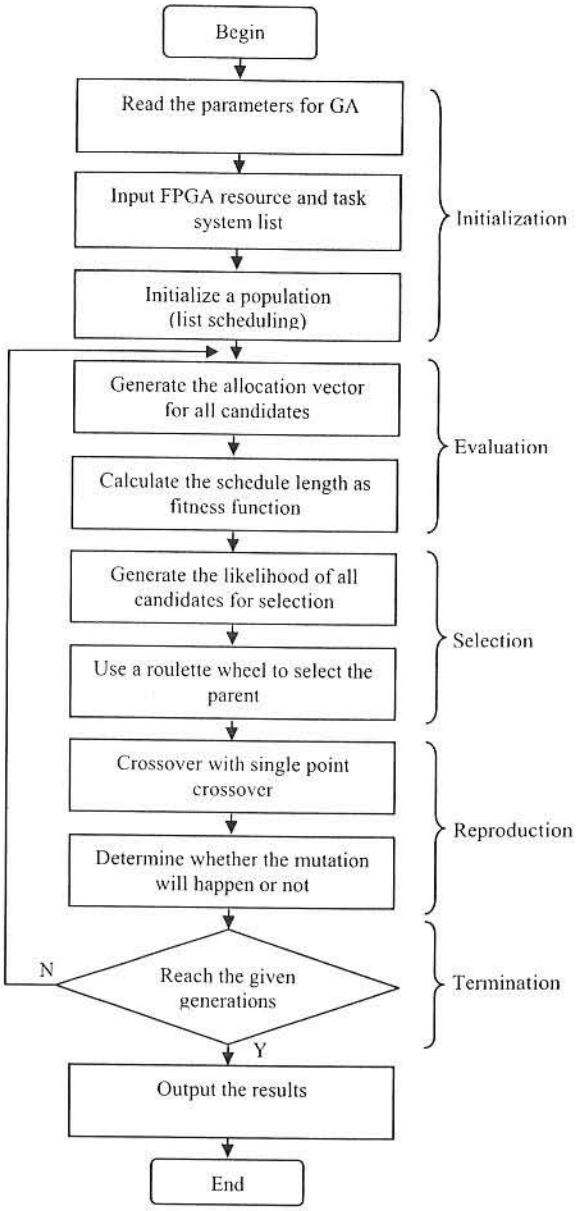


Figure 4: Genetic algorithm process

p is an important parameter for GA and can determine the search space complexity of GA directly. If we enlarge the search space, the computational complexity will be increased greatly. From this analysis, the population size must be selected carefully for GA. If it's too low, the search space is too small to provide enough samplings and GA gets the poor results. If it's too large, time complexity of the algorithm will be raised and GA gets the lower algorithm efficiency [3]. Also, finding a good termination condition and decreasing the number of the evolution generations can also improve the algorithm efficiency.

5 FPGA Resource Utilization Analysis Case Studies

In this example, the Power Quality Monitor System (PQMS) [4] is placed and routed into Xilinx XC3S1000 FPGA in different assignment schemes to test the utilization of the FPGA resources. The PQMS is designed to measure the quality and reliability of the power system. It can be decomposed into 10 tasks. Figure 6 is a DAG for the PQMS [2]. Table 3 shows the performance data of hardware and software of the PQMS.

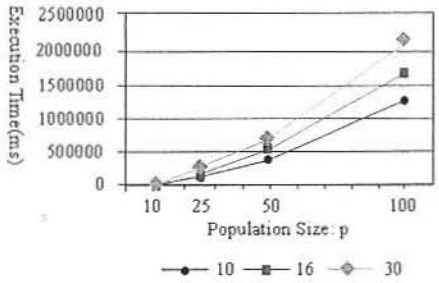
Xilinx XC3S1000 FPGA consists of 1,920 CLBs and 55,296 bytes bRAM. We assume that each Xilinx Microblaze soft processor will consume 500 CLBs. When all the tasks are implemented using the FPGA hardware logic (using HDL), it has been determined that the task system will require 12,020 CLBs (See Table 3). This value is more than the hardware logic resources in the given FPGA. So, it is impossible for this task system to be implemented using pure HDL.

As shown in Table 4, three cases with 1, 2, and 3 Microblaze soft processors in the FPGA are simulated using the Finite Resource Optimization Analysis Model. When one Xilinx Microblaze is used, the whole task system can fit into the FPGA because some of the tasks are assigned to the soft processor. The task system consumes 95.408 percent of FPGA hardware resources and the design gets an acceptable system performance. When two Xilinx Microblazes are employed, the percentage of the hardware resources utilization of FPGA is 90.668 percent and the better system performance is obtained. When three Xilinx Microblazes are used, the smallest percentage of the hardware resource utilization of FPGA is achieved, but the worst system performance is found. The reason is that the more tasks that are assigned into the soft processors, the performance suffers from the highly sequential operation in the processors. Thus, the best area-time trade-off in this example is to integrate two soft processors into a single Xilinx XC3S1000 FPGA. With this configuration, some tasks execute in software processors and the rest of the tasks are implemented using HDL. All are in one FPGA.

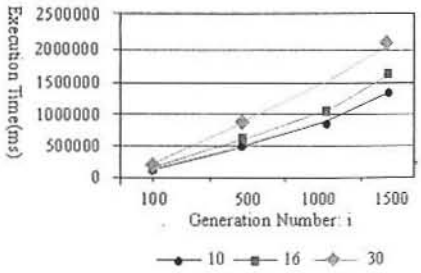
Table 2: The execution time of the GA with the different parameters

Test parameters		The execution time (milisecond) k: Task numbers		
		k=10	k=16	k=30
Population size: p (i=1000,pm=0.08, s=1)	10	17750	21100	26670
	25	92650	108560	141330
	50	334360	405150	532990
	100	1277990	1541400	2036900
Generation number: i (p=25,pm=0.08, s=1)	100	9890	10510	13570
	500	47600	53550	70900
	1000	92650	108560	141330
	1500	137940	164250	211690
Probability of mutation: pm (p=25,i=1000, s=1)	0.08	92650	108560	141330
	0.16	123670	141570	162660
	0.24	143730	157350	171860
	0.32	153570	164620	173100
Soft processor number: s (p=25,i=1000, pm=0.08)	1	92650	108560	141330
	2	116210	146350	156040
	3	115960	149410	158570
	4	132370	150050	157760

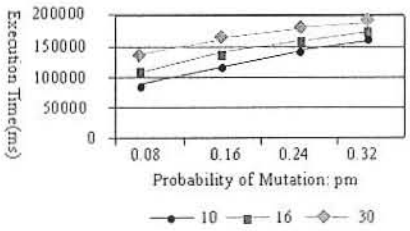
*Five tests are carried out for every item and the average value is shown in the table.



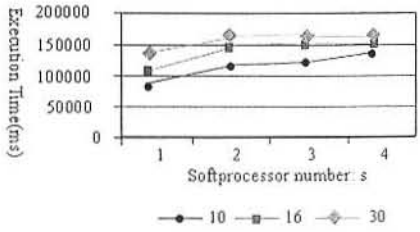
(a) Execution time characteristics with different population size



(b) Execution time characteristics with generation number I



(c) Execution time characteristics with probability of Mutation pm



(d) Execution time characteristics with soft processor number s

Figure 5: The execution time with the different parameters

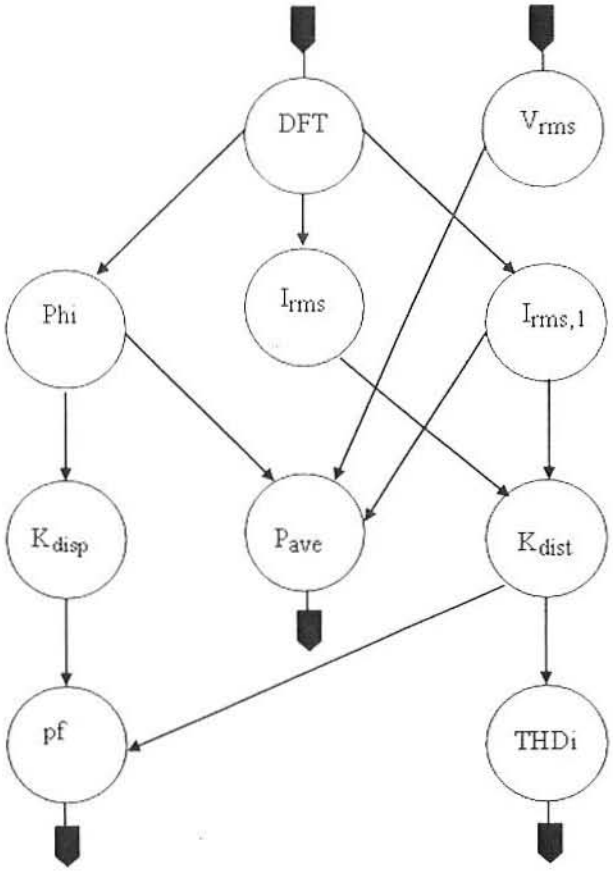


Figure 6: The DAG of PQMS

Table 3: Power quality monitor system task system information

Task name	Soft processor		FPGA hardware logic	
	Execution time (unit time)	Program memory required (bytes)	Execution time (unit time)	FPGA resources required (CLB)
DFT	83,124,480	7,688	61,540	697
Vrms	60,520	1,004	92.25	275
Phi	313,460	1,464	200	564
Irms	15,115,930	156	10,240	779
Irms,1	60,520	1,004	92.25	275
Kdisp	1,161,310	1,092	150	722
Pave	1,201,540	1,284	150	1,339
Kdist	26,680	28	132.02	619
Pf	8,600	28	29.72	74
THDi	73,030	72	272.42	1,330

6 More Comprehension Simulations and Results

In this section, we performed more simulations and expanded the results with a 30-tasks system, Space Shuttle Turbo Pump Task System [2]. The given FPGA is Xilinx

Table 4: The resource utilization of different FPGA resource assignment for PQMS

Simulation	1 Soft Processor		2 Soft Processors		3 Soft Processors	
	RU (%)	SL (ut)	RU (%)	SL (ut)	RU (%)	SL (ut)
1	95.94	18254300	88.65	15917000	82.34	98340100
2	95.94	18254300	88.70	15844000	82.40	98340100
3	95.94	18254300	88.70	15844000	82.34	98340100
4	95.26	18507300	92.66	15831000	96.72	98340100
5	95.26	18507300	92.71	16965600	82.40	98340100
6	91.98	18340300	92.60	15831000	96.72	98340100
7	95.94	18254300	92.60	15831000	96.82	98340100
8	95.94	18254300	88.70	15844000	82.40	98340100
9	95.94	18254300	92.66	15831000	82.34	98340100
10	95.94	18254300	88.70	15844000	96.93	98340100
Max	95.94	18507300	92.71	16965600	96.93	98340100
Min	91.98	18254300	88.65	15831000	82.34	98340100
Ave	95.408	18313500	90.668	15958260	88.141	98340100

*RU(Resources Utilization), SL(Schedule length), ut(unit time)
*The schedule length is an approximate value.
*It is assumed that each Xilinx Microblaze soft processor consumes 500 CLBs in a single FPGA.
*The number of CLBs in Xilinx XC3S1000 is 1920.
*Total resource usage if all tasks are to assign to: soft processor: 12,020, CLB: 6,674.
*The different random seed of GA is used in the simulation for each row in above table.

XC3S5000 with 8,320 CLBs and 234K bytes bRAMs. We also assumed that each soft processor consumes 500 CLBs. The FPGAs Finite Resource Optimization Analysis Model was used to simulate and test the utilization resources for the different number of soft processor to obtain the best possible performance within the available resources. Table 5 contains the results of the simulations.

For each category of a different number configuration (1-8) in Table 5, 10 simulations with different random seeds of Genetic Algorithm were completed. The total of eighty simulations is completed in this test. Generally, when we put eight soft processors into FPGA, the better resource utilization and the best system performance were achieved. But this result doesn't mean that the more soft processors we use, the better solution will be obtained.

Global System for Mobile Communication (GSM) [2] task system, with 16 tasks, is used to carry the same simulations. In this simulation, we use Xilinx XC3S1500 with 3,328 CLBs and 72K bytes bRAMs. Table 6 shows the results of the simulations for the 16 tasks. When one soft processor was integrated into FPGA, the better resource utilization and the best system performance were found in these cases.

Table 5: Feasible utilization resource found by GA for different FPGA resource assignment

Number of soft processor	Resource utilization (%)			Schedule length (unit time)		
	Min	Max	Ave	Min	Max	Ave
0	-	-	-	-	-	-
1	95.16	99.87	97.55	325	560	452.1
2	55.89	96.96	73.63	202	436	303.2
3	38.70	99.71	81.31	139	225	179.6
4	65.44	99.31	90.42	113	151	124.9
5	54.54	93.51	76.91	86	139	115.0
6	63.17	95.50	86.66	84	96	90.7
7	63.15	98.50	91.14	85	99	91.8
8	75.12	99.27	85.56	84	98	88.8

*Ten simulations with different random seeds of GA are made for each row in above table.
*It is assumed that each Xilinx Microblaze soft processor consumes 500 CLBs in a single FPGA.
*The number of CLBs in Xilinx XC3S5000 is 8320.
*Total resource usage if all tasks are to assign to: soft processor: 24580, CLB: 281575.

Table 6: Feasible utilization resource found by GA for different FPGA resource assignment for task system with 16 tasks

Number of soft processor	Resource utilization (%)			Schedule length (unit time)		
	Min	Max	Ave	Min	Max	Ave
0	-	-	-	-	-	-
1	83.44	99.43	94.88	156615	162516	160075.2
2	89.24	99.19	94.46	167131	173384	170734.1
3	83.02	99.90	97.05	172455	179460	176498.2
4	94.32	98.02	97.02	177675	188379	181404.6
5	93.72	99.76	96.61	199260	204091	201716.5
6	97.45	98.44	98.31	230581	234769	231999.8

*Ten simulations with different random seeds of GA are made for each row in above table.
*It is assumed that each Xilinx Microblaze soft processor consumes 500 CLBs in a single FPGA.
*The number of CLBs in Xilinx XC3S1500 is 3,328.
*Total resource usage if all tasks are to assign to: soft processor: 11,858, CLB: 12,014.

7 Conclusions

This paper presents a basic overview of the genetic algorithm with resource utilization analysis for FPGAs. How to utilize the finite FPGA resources optimally and efficiently is significant in the FPGA design. Integrating soft processor into FPGAs can greatly improve FPGA's resource utilization efficiency, but it doesn't mean that the more soft processors are better for the performance. In fact, it is the reverse case. For the different applications and given FPGA devices, we show that the trade-off between the resource utilization and the system performance can be found using FPGAs Finite Resource Optimization Analysis Model.

References

[1] Sin Ming Loo and Earl Wells, "Task Scheduling in a Finite-Resource, Reconfigurable Hardware/Software Codesign Environment," *INFORMS Journal of Computing*, 18(2):151-172, Spring 2006.

[2] David Lee McCarver, *Finite Resource Reconfigurable Hardware/Software Codesign: Case Studies*, M.S. Dissertation in Computer Engineering, Boise State University, June 2005.

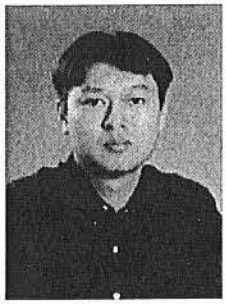
[3] Ling Wang, *Intelligent Optimization Algorithms with Applications*, Peking, TsingHua Univ Press, in Chinese, 2001.

[4] Theerayod WiangTong, Peter Y. K. Cheung, and Wayne Luk, "Comparing Three Heuristic Search Methods for Functional Partitioning in Hardware-Software Codesign," *Design Automation for Embedded Systems*, 6:425-449, 2002.

[5] Xilinx Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data sheet, http://direct.xilinx.com/bvdocs/publications/ds083.pdf, 2007.

[6] ZhaoKun Yang and ChongJun Yang, The Challenge of the Mathematicians in the Future, http://episte.math.ntu.edu.tw/articles/mm/mm_10_2_04/index.html, in Chinese, 6:28-36, 2002.

JingXia Wang received her M.S. in Electrical Engineering from the University of WuHan Surveying and Mapping Technology in 1994. From 1994 to 2003, she was employed as an Assistant Professor and currently is an Associate Professor in the Department of Electrical Engineering in Shenzhen Polytechnic. From 2005 to 2006, she worked in the Department of Electrical and Computer Engineering at Boise State University in the USA as a visiting scholar. Her research interests include embedded system, reconfigurable computing, computer architecture, hardware/software codesign.



Sin Ming Loo received his Ph.D. in Computer Engineering from the University of Alabama at Birmingham and the University of Alabama in Huntsville in 2003. From 1998 to 2003, he was involved in research projects which included development of space plasma simulation model utilizing parallel processing on 256-processor HP Exemplar X2000 and 128-processor SGI Origin, built and maintained 80-node Pentium-4 Beowulf cluster, and development of digital system rapid prototyping course materials. Dr. Loo is presently an Associate Professor of Electrical and Computer Engineering at Boise State University. His research interests include scheduling, parallel processing, distributed sensor networks, embedded system, hardware/software codesign, and reconfigurable computing.