INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 9, Number 2, Pages 208–216

RESEARCH ON FPGA BASED EVOLVABLE HARDWARE CHIPS FOR SOLVING SUPER-HIGH DIMENSIONAL EQUATIONS GROUP

KANGSHUN LI, ZHAOLU GUO, ZHANGXIN CHEN, AND BAOSHAN GE

Abstract. Solving a super-high dimensional equations group is widely used in science and engineering, but the slow solution speed is the biggest problem researchers face. Research on FPGA based evolvable hardware chips for solving the super-high dimensional equations group (SHDESC) is proposed in this paper. These chips can be implemented on a milliongate scale FPGA chip. The core architecture of SHDESC is a systolic array which consists of thousands of special arithmetic units and can execute many super-high dimensional matrix operations parallelly in short time as well as really achieve the purpose of high speed solution in hardware/software codesign. The experiments show that these chips can achieve high precision results in a short period of time to solve a super-high dimensional equations group.

Key words. Evolvable Hardware, Super-High Dimensional Equations Group, FPGA, Hardware/Software Codesign, Systolic Array

1. Introduction

Solving a super-high dimensional equations group is widely used in science and engineering, for instance, in optimization structure of oil wells and mining, structural analysis of mineral resources, calculation of deformation of bridges and housing construction after the force, computation of the flow field around the aircraft in aerodynamics, and evaluation of the flow of the atmosphere in weather forecast. These problems are transformed into a super-high dimensional equations group to solve ultimately. The process of solving these problems is described by a number of differential equations. In general, solving these differential equations is acquiring a super-high dimensional equations group which has thousands or even millions of unknowns after discretizing differential equations, no matter whether the difference method or finite element method is used. Thus the advantages and disadvantages of a method to solve the super-high dimensional equations group are restricted to solving these problems largely. To find an efficient, fast algorithm to solve a super-high dimensional equations group, many scholars continue to carry out in-depth and long-term studies[1]-[5].

With the advances in modern microelectronic technology, The EDA technology has developed greatly in every aspect. One of this technology is evolvable hardware [6][7] which uses an intelligent, self-reproduction and self-healing method to design hardware. It refers to hardware that can change its architecture and behavior automatically and dynamically through interacting with its environment, which can be presented in the formula [7]: Evolutionary Algorithm + Programmable Logical Devices = Evolvable Hardware.

Research on FPGA based evolvable hardware chips for solving a super-high dimensional equations group (SHDESC) is proposed in this paper. This method is

Received by the editors February 12, 2011 and, in revised form, February 20, 2011.
2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.
This research was supported by the National Natural Science Foundation of China (No. 70971043).

researched on solving the super-high dimensional equations group with the evolvable hardware theory and hardware/software codesign technology, and implemented on a million-gate scale FPGA chip. The bottleneck of slow speed is mainly concentrated on the super-high dimensional matrix operations in the process of solving the super-high dimensional equations group using traditional algorithms. But the chips are implemented on a million-gate scale FPGA chip, and its core architecture is a systolic array which consists of thousands of special arithmetic units that execute matrix operations concurrently, break through the bottleneck of slow solution speed, make super-high dimensional matrix operations in a short period of time, and really achieve the purpose of hardware/software codesign to solve with high speed. On the one hand, this chip combines the advantages of evolvable hardware, which are intelligence, efficiency, and parallelism. On the other hand it makes full use of the advantages of modern VLSI, which are high integration, low cost, and easy to achieve the operation of parallel computing. Therefore, it improves the speed of solving the super-high dimensional equations group greatly and really achieve the purpose of hardware/software codesign to solve with high speed.

2. Design Process of FPGA

The designing process of FPGA (Field Programmable Processor Arrays) is the process of using EDA development software and program utilities to develop FPGA chips. The development process of FPGA includes design of function circuit, design entry, function simulation, synthesis optimization, simulation after synthesis optimization, system implement, circuits and wires distribution, time sequence simulation and verifying, circuit board level simulation and verify, and programming of Soc (System of chip) and verifying[8][9].

2.1. Design of Function Circuit. Before system designing, some preparations such as schematic verifying and the selection of FPGA chip have to be done. Then the complexity of the indexes in system, running speed of FPGA, resources provided by chips and cost should be analyzed according to the requests of projects. After that, reasonable design schedule and suitable types of devices have to be selected; and at last the design method from top to down is used in general in the world. In this method we partition the circuit system to some base units as a top layer, and partition every unit to next base unit of next layer and so on, until we can find and load the EDA components from the components library to construct the complete circuit of the project.

2.2. Design Entry. The goal of design entry is to demonstrate the circuit system according to the requests of software development, and design entry is a process for input program to EDA utilities, and for input the components loaded from circuit components library to the circuit system using software or other design utilities. These software or design utilities are the main design utilities of EDA of FPGA, such as hard description language (HDL), Handle-C language and utility of schematic diagram description.

2.3. Function Simulation. Function simulation is called pre-simulation. Namely, function simulation is to verify the logic function of circuit system designed by using HDL programs before compiling the HDL programs, and this function simulation doesn't include the verification of delay circuits. In the first place, we use wave editor and HDL to build wave file and verified vector (namely, combining input signals to a execution sequence), and then the results will be saved in a report file and output signal waves which show the signals' change of all the circuit notes.

2.4. Synthesis Optimization. Synthesis is to transfer the abstract circuit layers in up level to the circuit layers in down level by using program description. Synthesis optimization is to optimize the circuits to be complanation circuits according to the goals and requests of the logic connection circuits, which are provided to implement distributed wires software of FPGA. In recent design circuit layers, synthesis optimization is to compile design entry to the logic connected network table composed with the based logic units such as 'and' gate, 'or' gate, 'not' gate, 'RAM' and trigger which are not the truly gate circuits. The truly gate circuits in FPGA have been produced through the function of circuits and wires distribution of FPGA producer and according to the standard network table of the gate level structures after synthesis optimization.

2.5. Simulation after Synthesis Optimization. The goal of simulation after synthesis optimization is to verify the consistency between synthesis results and original designed circuits. In the simulation process, we reversely label standard delay files generated through synthesis optimization to the synthesized simulation models and then we can estimate the effects brought by gates delay. Therefore, the running results in simulation after synthesis optimization are not accurate sometimes and are differences from the running results of FPGA after the wires are distributed. Because the recent synthesis utilities are relatively sophisticated, this step can be omitted in general FPGA designing.

2.6. System Implement, Circuits and Wires Distribution. System implement is to configure logic network generated through synthesis optimization on concrete FPGA chips, and the wires distribution is a most important process in system implement. Circuits distribution is to reasonably configure the hardware primitives and bottom units in the logic network tables to be the inherent hardware structure in chips of FPGA, and either speed optimal or area optimal must be selected. Wires distribution is also requested to reasonably connect all circuit units in FPGA correctly according the topology structure of circuit distribution and inner structure of chips.

2.7. Time Sequence Simulation and Verifying. Time sequence simulation and verifying is called post-simulation. It is to reversely label the delay information in circuits distribution and wires distribution on the designed network table to verify whether the phenomenon of violating time sequence (dissatisfying constrained condition of time sequence or violating the inherent rules of the time sequence) exist in the design process or not. Time sequence simulation includes all the delay information of circuits, and it will report the accurate truly state of the FPGA chips.

2.8. Circuit Board Level Simulation and Verification. Circuit board level simulation and verify is mainly to apply to the high-speed circuit design, and analyse the integrality of the high-speed system signals and the character of the magnetic disturbance. In general, we use the third part's utilities to conduct the circuit board level simulation and verification.

2.9. Programming of Soc and Verifying. Programming of Soc and debugging is the last step in FPGA circuits design. Programming of Soc is to produce the usable data files (Bitstream Generation) derived from HDL program, and then download these data bitstram files into the chips of FPGA (we call these types of program as chips-level program). Chipslevel programming must satisfy some

constrained conditions such as voltage programming, time sequence programming and algorithm programming etc.

3. Solving Super-High Dimensional Equations Group Based on Evolvable Hardware

The essence of solving super-high dimensional equations group is to find groups of vector X which satisfy: $AX = b$, for a given matrix A, vector b. An elite-subspace evolutionary algorithm which has very high efficiency has been presented in [10]. Its idea will be used in solving super-high dimensional equations group in the SHDESC chip. In the design of SHDESC, real coding is adopted, and the individuals are represented by vector X'.The core question of design SHDESC are the storage of matrix A, the design of fitness function, selection and cross-replace operator.

3.1. The Storage of Matrix A. The storage of matrix A is considerable cost determined by the characteristics of super-high dimensional equations group. For example, a 20000-dimensional equations group, the storage of matrix A costs $4 * 20000 * 20000$ bytes, nearly 2G bytes, if stored with traditional method. Apparently, it is not realistic, as well as not feasible. Actually, the matrix A is always sparse matrix in science and engineering and it exists a large number of zero elements. Thus, it's necessary to adopt triad (row, col, value) to store the matrix A compressingly and it is effective proved by experiments.

3.2. The Design of Fitness Function. In the EHW, The design of fitness function is essential, for it directly affects the convergence speed and accuracy of the solution. In this paper, the fitness function adopts 2-norm. For every individual X', the formula of fitness value calculation is:

$$Fitness(X') = \left\|AX' + b\right\|_2 \text{ ,where } \left\|X\right\|_2 \text{ is 2-norm of the vector } X \tag{1}$$

This fitness function can reflect the sum-gap between individual X' and the aim solution. While the less the fitness value is, the closer between the individual X' and the target solution is, the individual X' is better.

3.3. The Design of Selection Operator. In the EHW, the selection operator is based on the idea of the elite-subspace selection and is depicted as follows:
Step 1: Using formula (1), calculate the fitness value of each individual in the population as follows:

$$P(t) = \{X'_1, X'_2, X'_3, ..., X'_{Popsize}\}$$

Step 2: Sort the individuals in population $P(t)$ according to the fitness value in ascending order. After sorting still recorded as follows:

$$P(t) = \{X'_1, X'_2, X'_3, ..., X'_{Popsize}\}. \text{ Where } X'_1 \text{ is the best individual,}$$

and $X'_{Popsize}$ is the worst individual.
Step 3: Select the best $Elite\_K(Elite\_K \leq Space\_M)$ individuals:

$$X''_1, X''_2, X''_3, ..., X''_{Elite\_K}$$

Step 4: Select $Space\_M - Elite\_K$ individuals from the remaining $PopSize - Elite\_K$ individuals in the population $P(t)$ randomly, and recorded as follows:

$$X''_{Elite\_K+1}, X''_{Elite\_K+2}, X''_{Elite\_K+3}, ..., X''_{Space\_M}$$

Step 5: End of selection operator.

In the selection operator above, there are $Space\_M$ individuals are selected, in which the best $Elite\_K$ individuals are selected in the step 3, which sufficiently exert the oriented effect of the better individuals to enhance the precision and convergent speed. While in the step 4, $Space\_M - Elite\_K$ individuals are selected from the remaining $PopSize - Elite\_K$ individuals in the population $P(t)$ randomly, which ensure the diversity of the population, avoid falling into local optima and leading to precocity.

3.4. The Design of Cross-Replace Operator. In the EHW, the cross-replace operator adopts the idea of Multi-Parent-Crossover operator which was proposed in [11].When using the Multi-Parent-Crossover operator to solve the question in this paper, the detail design is depicted as follows:
Step 1: $I = 1$;
Step 2: If $(I > Replace\_L)$ then goto Step 8;
Step 3: Generate $a_1, a_2, ..., a_{Space\_M}$ randomly, which satisfy :

$$\sum_{j=1}^{Space\_M} a_j = 1, -0.5 \leq a_j \leq 1.5.$$

Step 4: Generate a new individual $\overline{X}$:

$$\overline{X} = \sum_{j=1}^{Space\_M} a_j X_j''.$$

In the formula above, the $Space\_M$ individuals:

$$X_1'', X_2'', X_3'', ..., X_{Space\_M}'' \text{ are produced by the selection operator.}$$

Step 5: Assume X_{worst} is the worst individual.
If individual $\overline{X}$ is better than X_{worst} then $\overline{X}$ replaces X_{worst}.
Step 6: $I = I + 1$;
Step 7: Goto Step 2;
Step 8: End of cross-replace operator.
In the cross-replace operator above, $Replace\_L$ is the number of elimination individuals. it determines the pressure of elimination in the algorithm. Step 4 is Multi-Parent-Crossover operation.

3.5. The General Description of the Algorithm. All the steps of the algorithm to solve super-high dimensional Equations Group are depicted as follows:
Step 1: $t = 1$;
Step 2: Initialize the population $P(t)$, generate $Popsize$ individuals randomly:

$$X_1', X_2', ..., X_{Popsize}'.$$

Step 3: Execute the selection operator.
Step 4: Execute the cross-replace operator.
Step 5: Assume X_{worst} is the worst individual, X_{best} is the best individual in the population $P(t)$.
If $Fitnes(Xworst) == Fitnes(X_{best})$ then goto Step 8;else goto Step 6.
Stpe 6: $t = t + 1$;
Step 7: Goto Step 3;
Step 8: The end.

4. The Design and Implementation of SHDESC

4.1. Speed Bottleneck Analysis and Breakthrough. The algorithm to solve super-high dimensional equations group above is simulated with software. The analysis of the simulation results shows convergent speed of the algorithm is very fast when the dimension of the equations group is less than 1000, but the convergent speed of the algorithm is rapidly declining when the dimension of the equations group is more than 1000.Through the analysis, it is known that the Bottleneck of slow speed is mainly concentrated in the operations of calculating the fitness value of each individual using formula (1). For an equations group which has N dimensions, calculating the fitness value of a single individual should operate: $N*(N+1)$ floating-point multiplications + $N*(N+1)$ floating-point additions. If N is more than 1000, the operations of floating-point are so large that is led to the Bottleneck of the speed of the whole algorithm.

Thus, the key to break through the Bottleneck is to accelerate the speed of calculating the fitness value of individuals using formula (1). The method of SHDESC is that the core architecture of SHDESC is a systolic array which consists of thousands of special arithmetic units, which execute matrix operations concurrently, break through the Bottleneck of slow solving speed, and make super-high dimensional matrix operations in a short period of time.

4.2. The Core Structure of Parallel Processing of SHDESC. From the algorithm design above, it is known that the structure of SHDESC has four parts: Initializing Population Module, Calculating Fitness Value Module, Selection Operator Module, and Cross-Replace Operator Module. And Their algorithms design above are implemented by VHDL, and the whole structure and data flow diagram is shown in figure 1. Where the Control Module generates the entire control signal and synchronizes all the other Module as well as generates random number to the Initializing Population Module, Selection Operator Module and Cross-Replace Operator Module. The Storage Module, consists of memory bank and read/write controller, in which the matrix A, vector b, and all the individuals of population $P(t)$ are stored. The read/write controller generate read/write control signal to the memory bank for every address which any other module provider.

In the structure of SHDESC, the core Module is Calculating Fitness Value Module which is a systolic array and consists of N special ALUs shown in figure 2. In the figure 2, CLK is clock signal, and $b_N b_{N-1}, ..., b_1$ are elements of vector b, which come into the systolic array one by one for every clock according to the order from 1 to N. The structure of each special ALU is shown in figure 3, which has four input terminals: CLK,X, Y, Z and two output terminals: Z_1,X_1.Where $Z_1 = XY + Z, X_1 = X$ and CLK is clock signal.

5. Experiments and Analysis

The whole design of SHDESC is depicted by VHDL and implemented on the XC2V1000 (100 million gates) FPGA. The following experiments have been done, and compared with the software implementation.

5.1. Experiment 1. Generate an equations group which has 100 dimensions. The algorithm is implemented by software and hardware. The results of experiment 1 are shown in table 1.

From the results of experiment 1, it is known: when solving the equations group of 100-dimensions, both the Solution Error is 0 when the algorithm is implemented by software to solve this question and when the algorithm is implemented by hardware.

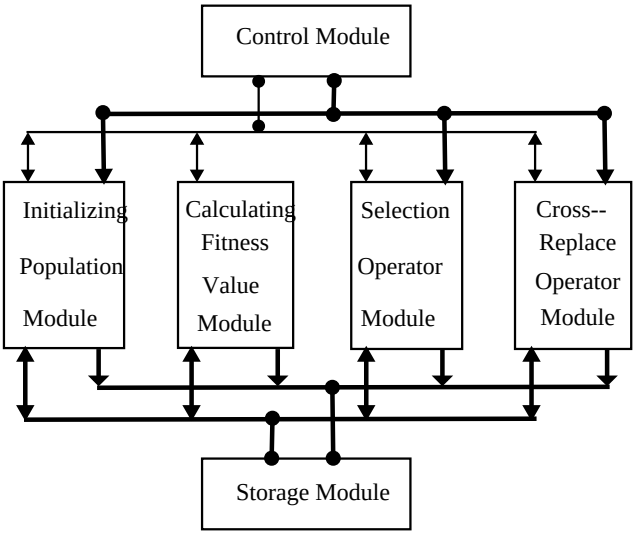

FIGURE 1. The whole Structure and Data Flow Diagram of SHDESC

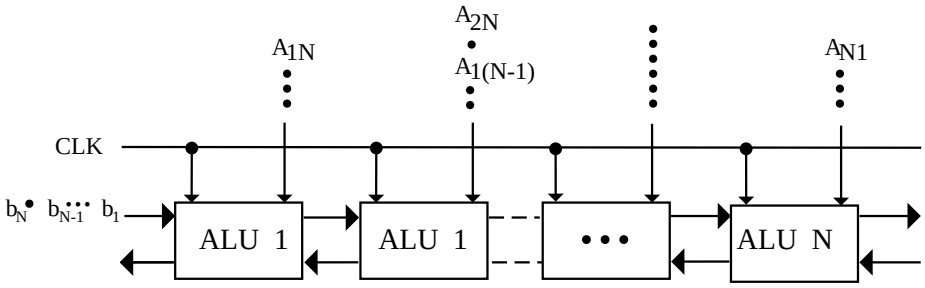

FIGURE 2. The Systolic Array of Calculating Fitness Value

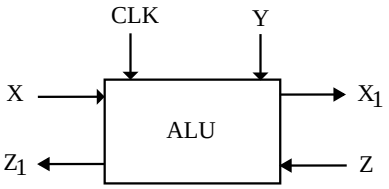

FIGURE 3. The Structure of Special ALU

TABLE 1. The results of Experiment 1

Method	Running Time (Unit:s)	Solution Error
Software	21.56	0
Hardware	10.18	0

TABLE 2. The results of Experiment 2

Method	Running Time (Unit:s)	Solution Error
Software	218.38	2.4703179247e-27
Hardware	20.27	0

Thus, the precision is very high. And also the speed of the algorithm implemented by hardware is about 1 times faster than implemented by software. As a result, the matrix operations executed by hardware concurrently could increase the speed of the algorithm.

5.2. Experiment 2. Generate an equations group which has 1000 dimensions. The algorithm is implemented by software and hardware. The results of experiment 2 are shown in table 2.

From the results of experiment 2, it is known: when solving the equations group of 1000-dimensions, the Solution Error is 2.4703179247e-27 when the algorithm is implemented by software to solve this question and the Solution Error is 0 when the algorithm is implemented by hardware. And also the speed of the algorithm implemented by hardware is obviously faster than implemented by software, which is faster more than about 10 times. As a result, when the dimension of equations group is high, the super-high dimensional matrix operations executed by hardware concurrently could increase the speed of the whole algorithm and could acquire high precision results much faster than software implement.

6. Conclusions

A research on FPGA based evolvable hardware chips for solving super-high dimensional equations group (SHDESC) was proposed in this paper. This method was researched on solving super-high dimensional equations group with evolvable hardware theory and Hardware/Software Codesign technology, and implemented on a million-gate scale FPGA chip. The Bottleneck of slow speed are mainly concentrated in the super-high dimensional matrix operations in the process of solving super-high dimensional equations group using traditional algorithm. But this chip is implemented on a million-gate scale FPGA chip, and its core architecture was a systolic array which consists of thousands of special arithmetic units, which execute matrix operations concurrently, break through the Bottleneck of slow solving speed, make super-high dimensional matrix operations in a short period of time, and really achieve the purpose of Hardware/Software Codesign to solve with high speed. On the one hand, this chip combines the advantages of evolvable hardware, which are intelligence, efficiency, and parallelism; On the other hand it makes full use of the advantages of modern VLSI, which are high integration, low cost and easy to achieve the operation of parallel computing. Therefore, it improves the speed of solving super-high dimensional equations group greatly, really achieve the purpose of Hardware/Software Codesign to solve with high speed and solve the problems which are slow speed of solving super-high dimensional equations group in science and engineering.

References

[1] CHI Li-Hua, LIU Jie, LI Xiao-Mei. An Effective Parallel Algorithm for Tridagonal Linear Equation. CHINESE JOURNAL OF COMPUTERS, 22(2):218-221, 1999.

[2] LUO Zhi-Gang, LI Xiao-Mei. A Parallel Algorithm for Block-Tridagonal Linear Systems on Distributed-Memory Multicomputers. CHINESE JOURNAL OF COMPUTERS, 23(10):1028-1034, 2000.
[3] LUO Zhi-Gang, LI Xiao-Mei. A Parallel Solver for Certain Toeplitz Tridiagonal Systems on Distributed-Memory Multicomputers. CHINESE JOURNAL OF COMPUTERS, 24(2):173-178, 2001.
[4] SHENG Yue-bin, SONG Xiao-qiu, LIU De-gui. Parallel algorithm for banded linear equations on distributed-memory multicomputers. Systems Engineering and Electronics, 26(7):967-969, 2004.
[5] Cui Xi-ningLv Quan-yi. A parallel algorithm for band linear systems. Applied Mathematics and Computation, 181(1):40-47, 2006.
[6] H. de Garis, Evolvable Hardware: Genetic Programming of a Darwin Machine, In: Artificial Neural Nets and Genetic Algorithms, Springer Verlag, 1993.
[7] Zhengjun Pan, Lishan Kang, Yuping Chen, Evolutionary Computation. Beijing:Tsinghua University Press, 1998.
[8] Hofmann Andreas, Waldschmidt Klaus. SDVMR: A scalable firmware for FPGA-based multicore Systems-on-Chip. Proceedings-IEEE Computer Society Annual Symposium on VLSI: Trends in VLSI Technology and Design, ISVLSI. Montpellier, France, 2008.
[9] Noseworthy Joshua, Lecser Miriam. Efficient communication between the embedded processor and the reconfigurable logic on an FPGA. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 16(8):1083-1090, 2008.
[10] WU Zhi-jian, KANG Li-shah, ZOU Xiu-fen, An Elite-subspace Evolutionary Algorithm for Solving Function Optimization Problems, Computer Applicafions, 23(2):13-15, 2003.
[11] Guo Tao, Lishan Kang, Yan Li, A New Algorithm for Solving Inequality Constrained Function Optimization Problems, Wuhan University Journal of Natural Sciences, 45(5B):771-775, 1999.

School of Information, South China Agricultural University, Guangzhou 510642, China. And School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou 341000, China
E-mail: likangshun@sina.com

State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China
E-mail: gzl990137@gmail.com

Center for Computational Geoscienes and Mathematics, Faculty of Science, Xi'an Jiaotong University, Xi'an,710049, P.R. China and Department of Chemical and Petroleum Engineering, Schulich School of Engineering, University of Calgary, 2500 University Drive N.W., Calgary, Alberta, Canada
E-mail: zhachen@ucalgary.ca

Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
E-mail: gbsh@263.net