Design of a Microprocessor with Genetic Instructions

Jeong-Pil Choi, Kang-Ryong Han, Ho-Jeong Song, In-Jae Hwang*, Gi-Yong Song

Dept. of Computer Engineering, Dept. of Computer Education*

Chungbuk National University, Cheongju Chungbuk 361-763 Korea

Tel : +82-43-261-2452 Fax : +82-43-262-2449

E-mail : gysong@chungbuk.ac.kr

Abstract : A microprocessor with genetic instructions such as crossover, mutation and inversion is proposed. The processor is modeled using VHDL, synthesized to a schematic and implemented on a FPGA. The control path is implemented with a microprogram consisting of about 150 32-bit microwords, and the operation of each instruction is checked through simulation.

1. Introduction

The genetic algorithm (GA) is a powerful, domain-independent search technique that was inspired by Darwinian theory. It emulates the natural process of evolution to perform an efficient and systematic search for the solution space to progress toward the optimum[1]. It is based on the theory of natural selection that assumes that individuals with certain characteristics are more able to survive, and hence pass their characteristics to their offspring. All genetic algorithms work on a population. Each individual in the population is called a chromosome and these individuals are often coded as binary strings. In each iteration of the GA, a new generation is evolved from the existing population in an attempt to obtain better solutions[2]. Crossover is the main operator used for reproduction. It combines portions of two parents to create two new individuals, called offspring, which inherit a combination of the features of the parents. Mutation is an incremental change made to each member of the population with a very small probability. It enables new features to be introduced into a population. In this paper we design a processor which has genetic instructions such as crossover, mutation and inversion as well as ordinary instructions. The processor with genetic instructions can perform genetic algorithm-based codes effectively.

2. Microprocessor with Genetic Instructions

A interface is shown in Fig. 1 and a datapath for the processor is illustrated in Fig. 2.

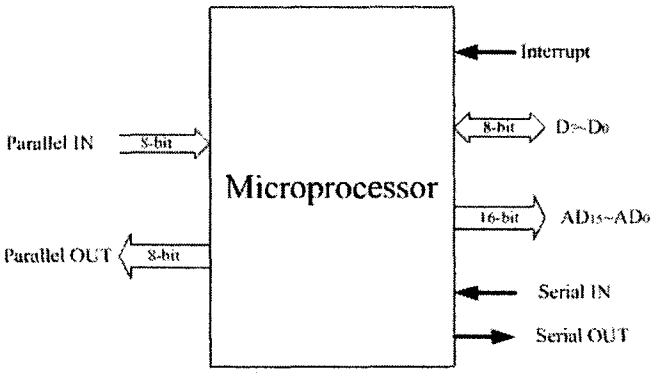

Fig. 1. Interface of the processor

The datapath contains six 16-bit internal registers; PC, SP, Y, BASE, MA, and MD, and six 8-bit internal registers; IR, AC, PIN, SIN, POUT, and SOUT. The microprogram for the processor can be up to 1Kbyte and 64Kbyte memory space can be referenced directly. The processor has four addressing modes; immediate, direct, register indirect and register based addressing mode. The instruction set, Table 1 and Table 2, covers 35 instructions including crossover, mutation and inversion[3~6].

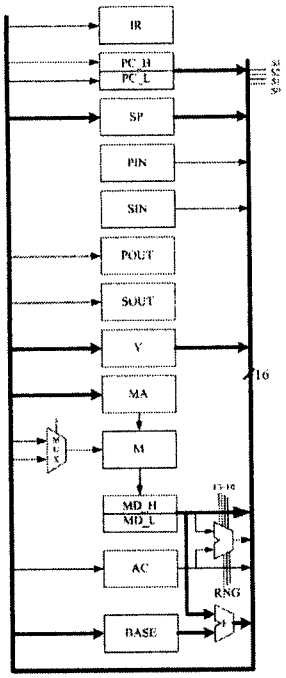

Fig. 2. Datapath for the processor

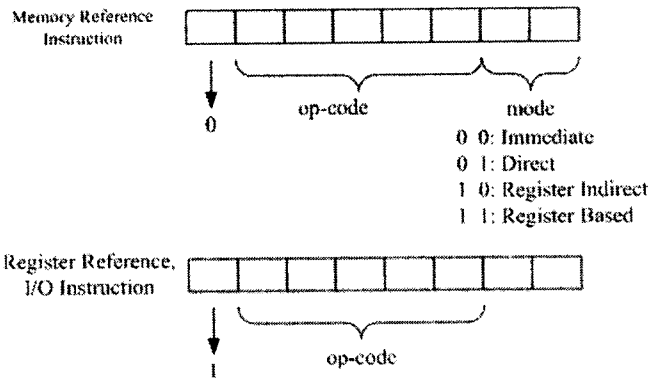

Fig. 3. Instruction formats

Table 1. Memory reference instructions

Instruction	Code	Description
ADD	00000	AC ← AC + [M]/imm; CF, ZF
AND	00001	AC ← AC ∧ [M]/imm; ZF
OR	00010	AC ← AC ∨ [M]/imm; ZF
CMP	00011	CF ← AC - [M]/imm; CF, ZF
LDA	00100	AC ← [M]/imm;
LDSP	00101	SP ← [M]/imm;
LDY	00110	Y ← [M]/imm;
ST	00111	[M] ← AC
BR	01000	PC ← imm
BC	01001	If CF='1' then PC ← imm
BZ	01010	If ZF='1' then PC ← imm
BY	01011	If YZF='1' then PC ← imm
CALL	01100	[SP] ← PC, PC ← imm, SP ← SP+1
RET	01101	PC ← [SP-1]
XOR	01110	AC ← AC ⊕ [M]/imm; ZF
LDB	01111	BASE ← [M]/imm
XOVRML	10000	AC ← AC(7..*i*) & [M](*i*-1..0)
XOVRLM	10001	AC ← [M](7..*i*) & AC(*i*-1..0)
XOVR2	10010	AC ← AC(7..*j*) & [M](*j*-1..*i*)&AC(*i*-1..0)

Table 2. Register and I/O instructions

Instruction	Code	Description
INC	00000	Y ← Y + 1
COM	00001	AC ← $\overline{AC}$
SHL	00010	AC ← SHL(AC)
SHR	00011	AC ← SHR(AC)
ROTL	00100	AC ← ROTL(AC)
ROTR	00101	AC ← ROTR(AC)
PIN	00110	AC ← PIN
SIN	00111	AC ← SIN
POUT	01000	POUT ← AC
SOUT	01001	SOUT ← AC
CC	01010	CF ← 0
HALT	01011	CC ← 0
INV	01100	AC(*j*..*i*) ← AC(*i*..*j*)
MUT1	01101	If AC(*i*)='1' then AC(*i*)='0' else AC(*i*)='1'
MUT2	01110	If AC(*i*)='1' then AC(*i*)='0' else AC(*i*)='1', If AC(*j*)='1' then AC(*j*)='0' else AC(*j*)='1'

3. Simulation and Synthesis

The simulation of the proposed processor showed that each instruction in the instruction set executes correctly as shown in Fig. 4, Fig. 5. The processor was synthesized to a schematic with Synopsys design compiler and implemented on a FPGA using Xilinx Foundation. The schematic and the

implementation on a FPGA of the processor are shown in Fig. 6, Fig. 7 respectively[7].

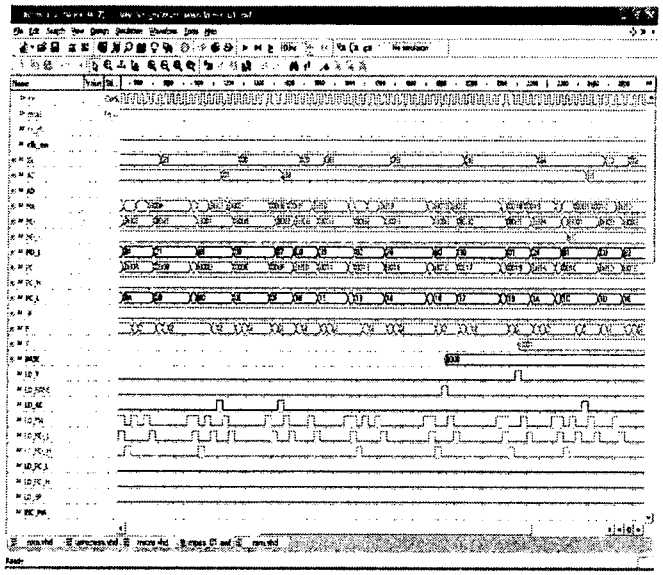

Fig. 4. Simulation of the instruction set I

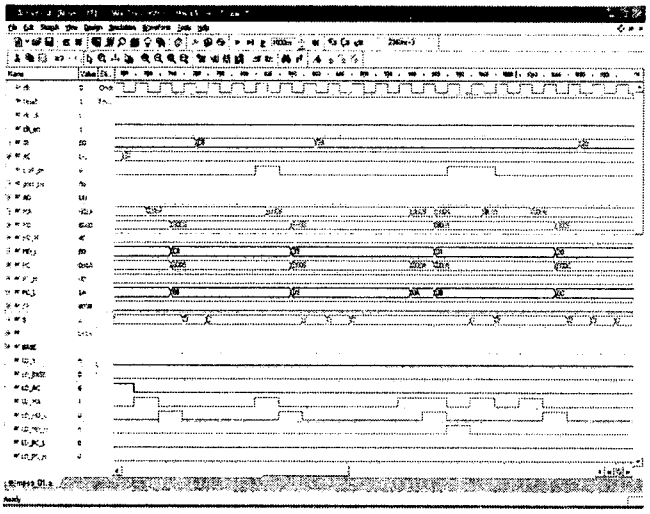

Fig. 5. Simulation of the instruction set II

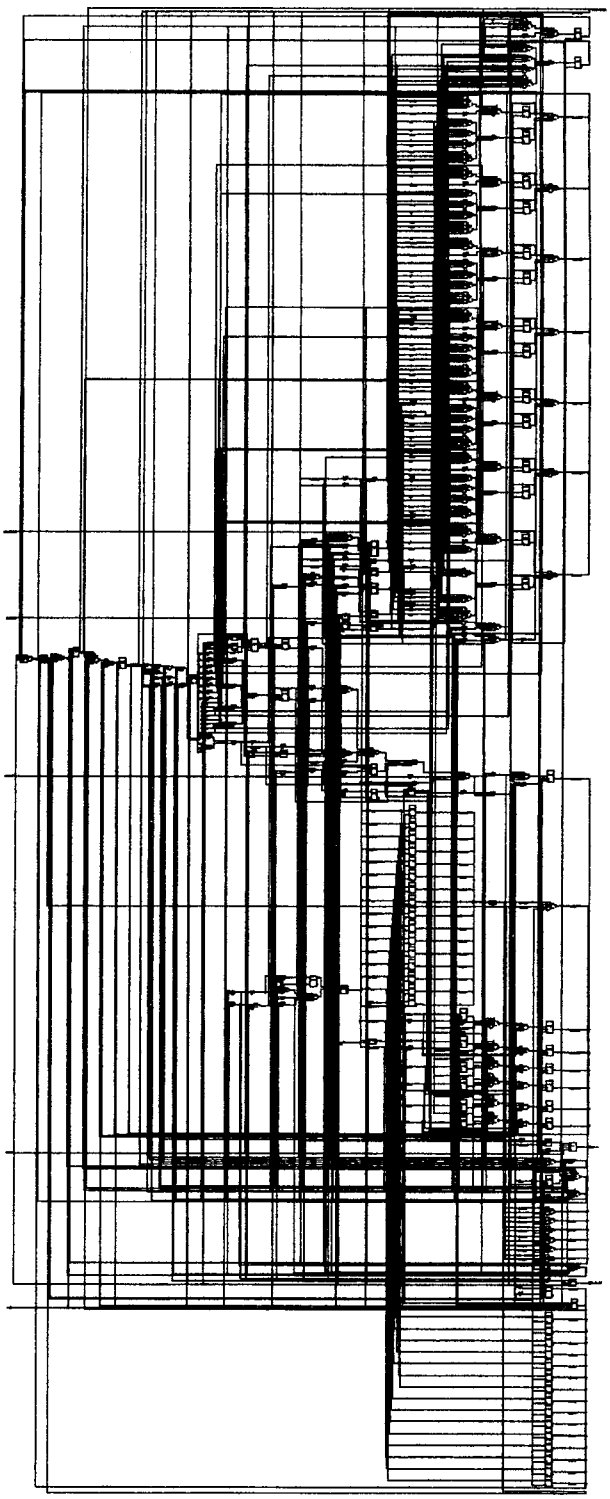

Fig. 6. Synthesized schematic for the processor

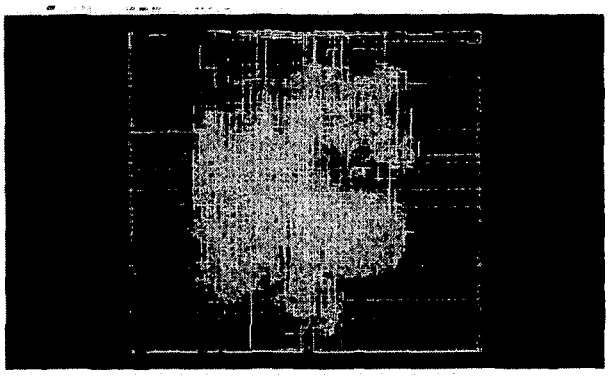

Fig. 7. Implementation of the processor on a FPGA

The layout of the processor is shown in Fig. 8. It is implemented automatically to a layout using Apollo tool provided by AVANT!

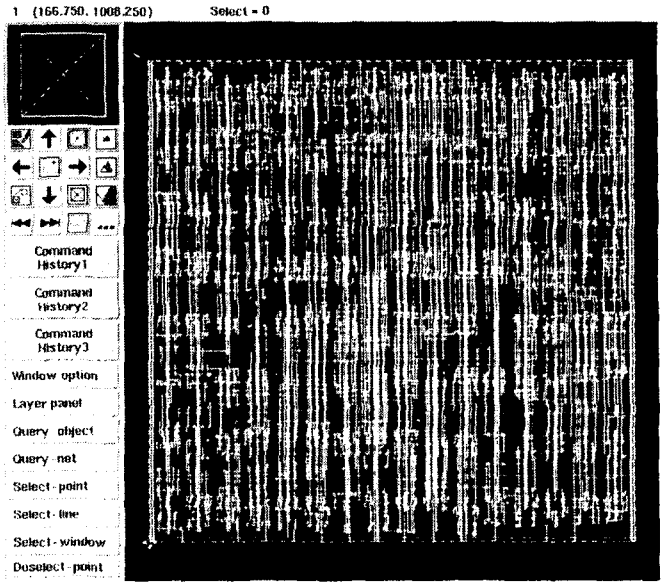

Fig. 8. Layout of the processor

4. Conclusions

This paper proposed a microprocessor which is capable of 35 instructions including genetic instructions such as crossover, mutation and inversion. The proposed processor has four addressing modes and facility for interrupt as well. The processor was designed with VHDL and the functionality of the processor was verified through simulation. The implementation on Xilinx FPGA VL-XC4062XL executed test programs in ROM correctly.

References

[1] M. Gen, *Genetic Algorithms and Engineering Optimization*, John Wiley & Sons, INC., 2000.

[2] Pinaki Mazumder, *Genetic Algorithms for VLSI Design, Layout & Test Automation*, Prentice-Hall, 1999.

[3] Douglas J. Smith, *HDL Chip Design*, Doone Publications, 2000.

[4] K. C. Chang, *Digital Systems Design with VHDL and Synthesis*, IEEE Computer Society, 1999.

[5] Charles H. Roth, Jr., *Digital Systems Design using VHDL*, PWS Publications, 1998.

[6] John P. Hayes, *Computer Architecture and Organizations*, McGraw-Hill, 1998.

[7] Xilinx, *Xilinx Data Book*, 1999.