EE213

WIMP51

Adding
Bit Addressable Internal Registers
to the Wimp51

Cooperative Engineering Program:
Missouri University of Science & Technology &
Missouri State University

under the direction of

Professor Rohit Dua PhD

Abstract: Starting with a simple FPGA based processor of the Von Neumann Architecture design called Wimp51 (Weekend Instructional Micro Processor), the group implemented the added capability of a Bit Addressable Register.

Table of Contents

Introduction

This project was designed with the intent of having participants become intimately familiar with the internal functions of a simple processor. To accomplish this the group was provided with a simple WIMP51 processor designed on an Altera DE2 FPGA board with the assignment of designing and implementing additional functionality to the processor.

Chosen design

The project approved was to add bit addressability to the internal registers. The original Wimp51 was built such that the entire value of each register was set at once. With this design the value of bit can be set or cleared as desired.

Description of work

In order t e used. This required
instructio , logic had to be created to
recognize ypical input and would be routed
differentl ed in order to implement this
design. T

Registe

In the ori ive a full byte from the
accumula these registers could not be
selected. ster can be set or cleared as
desired. I g the bits in and in figure_2, the
2nd_regi

Bit_Ad

The Bit A input of an instruction pair and
decodes i d set of an instruction pair is
available

REG_i

When the the output will go low. This tells
the 8-2-1 he Standard Register

PC_ALU

Inside the PC_A ve the program counter
increment on the of a two byte instruction
set.

AUX_WE

In this block, fig de requirements. When this
device detects th h the data at the inputs

BIT_SEL_C

The BIT_SEL_C de of each Register, of
which 8 are loca lect the individual bits of
each register

REG_we

Decoding of ou so that we would be able to store th n the AUX, but it wouldn't save a

Register

The Register, f on of 8 BIT_SEL_CO n in each register.

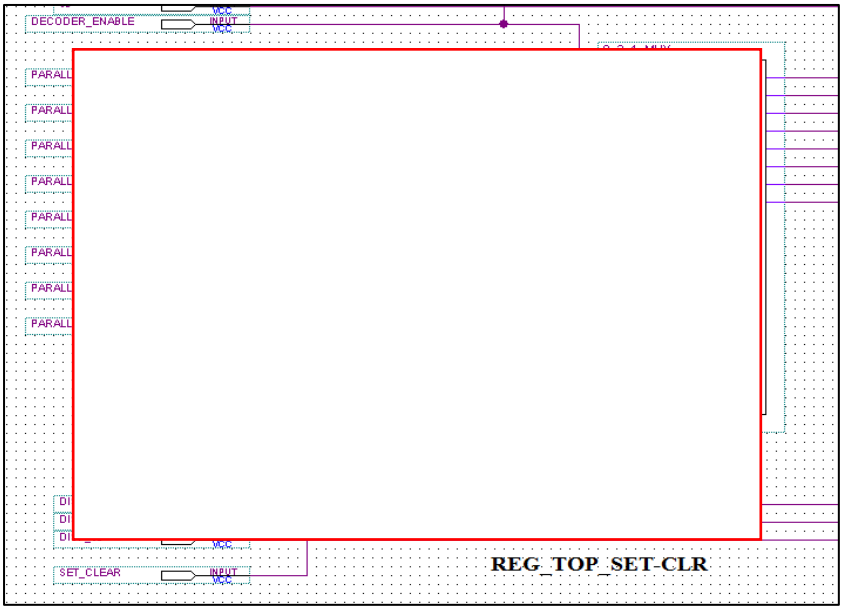

Figure 1: Register_Top (8-2-1 MUX controls each bit of the selected register)

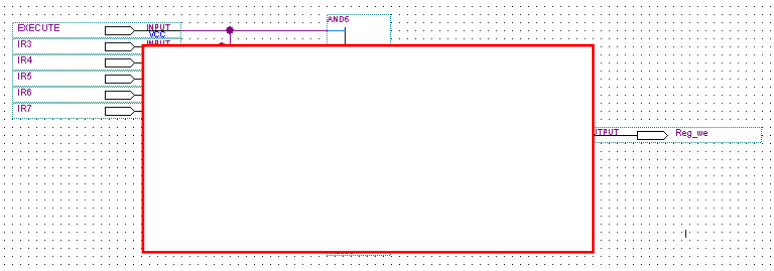

Figure 2: REG_we (Decoding for storing the bit in REG_TOP)

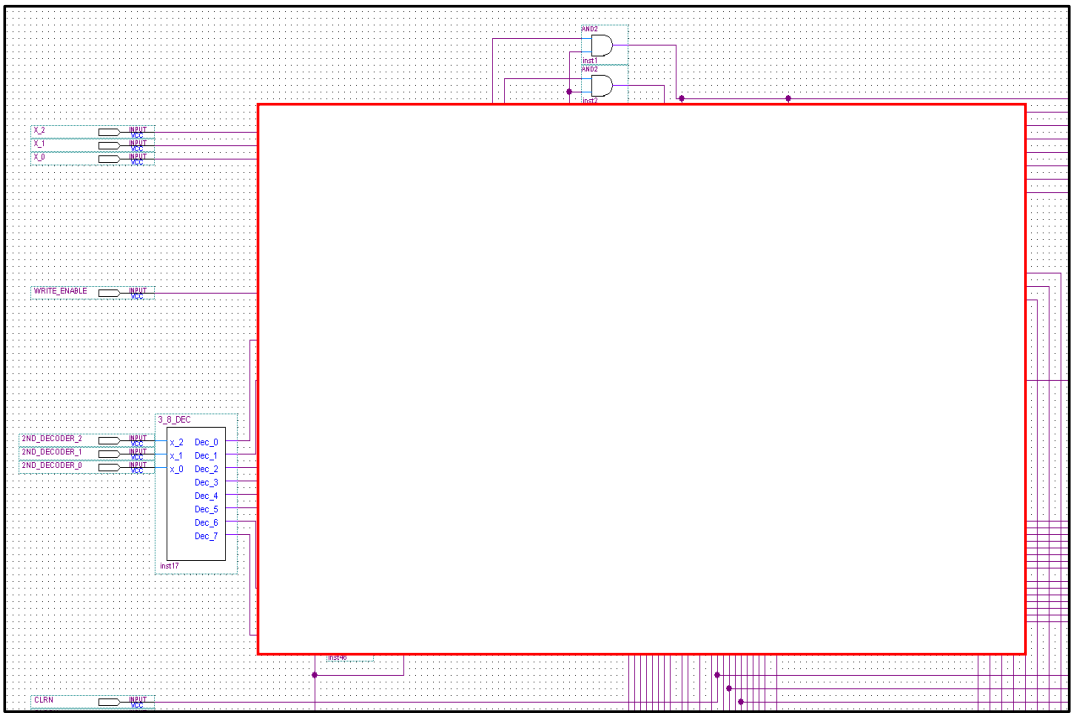

Figure 3: Register_Top (2nd_Decoder provides each bit)

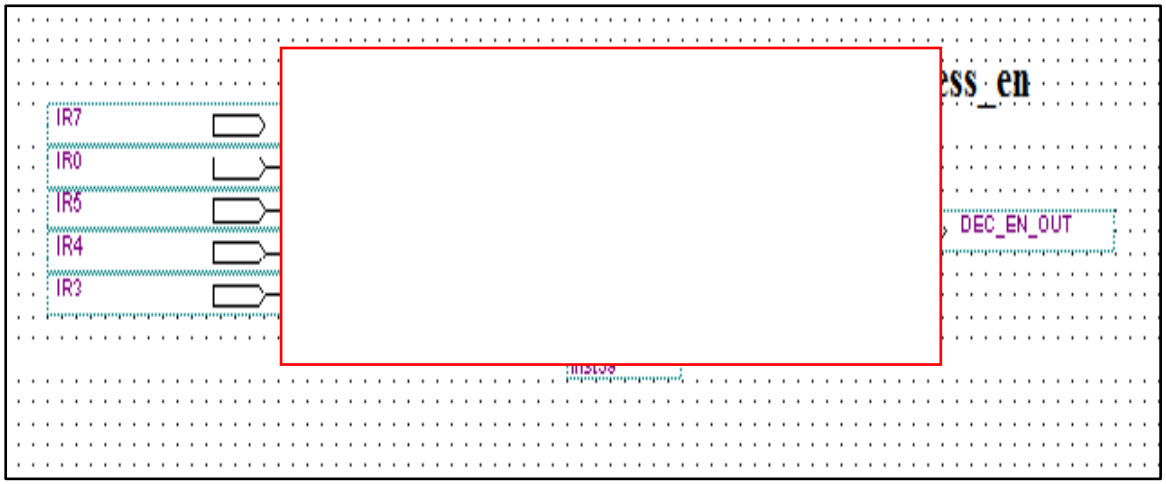

Figure 4: Bit Address Enable

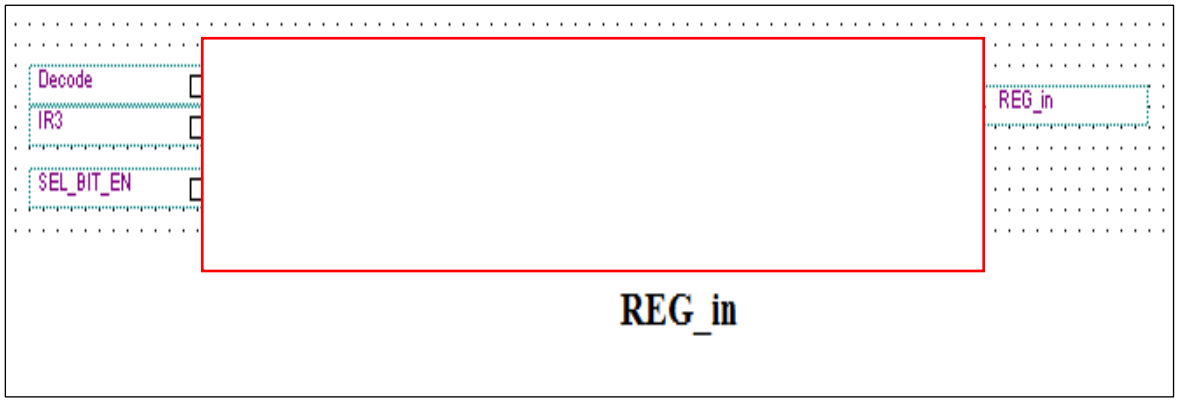

Figure 5: Register In

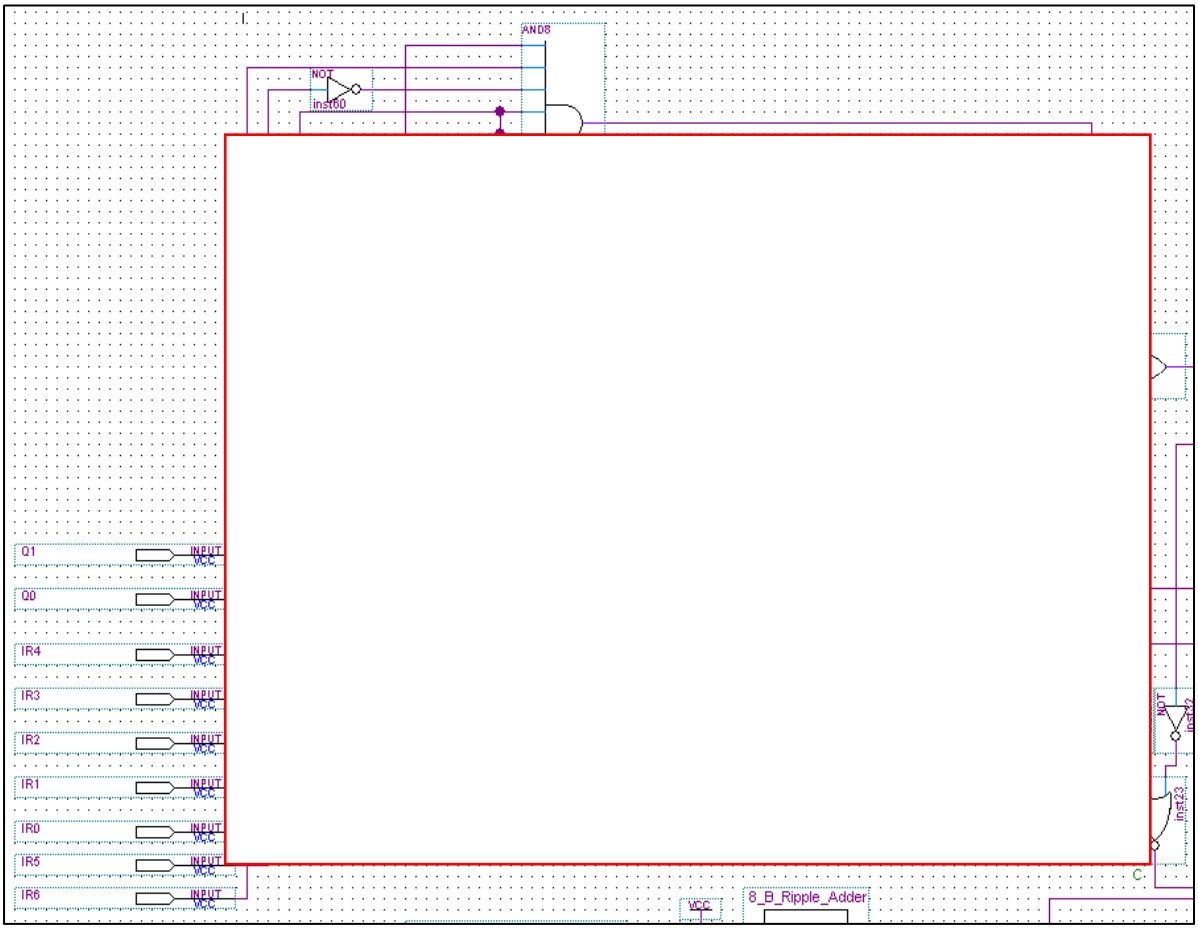

Figure 6: Inside PC ALU

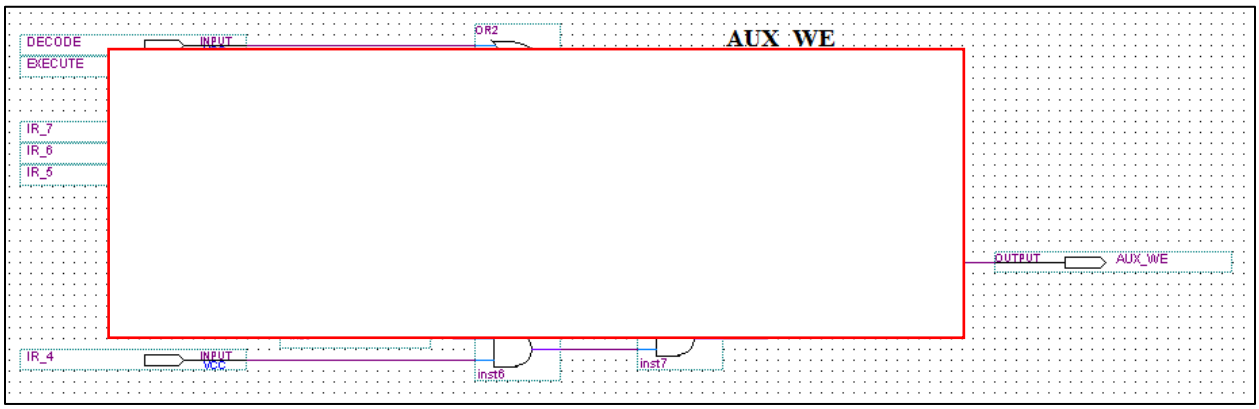

Figure 7: AUX_WE

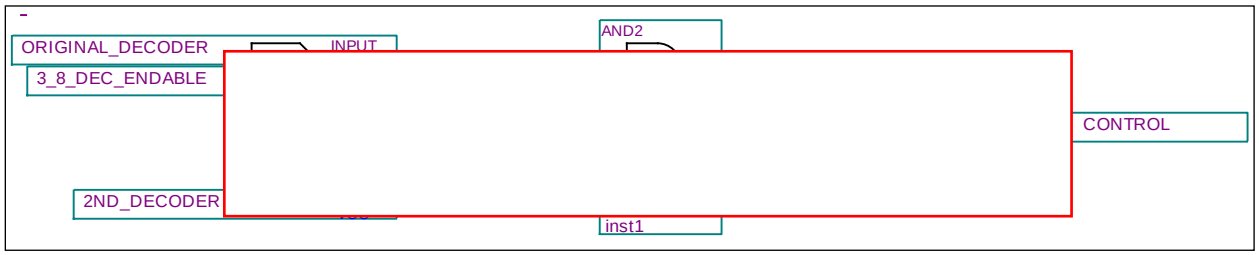

Figure 8: BIT_SEL_CONTROL

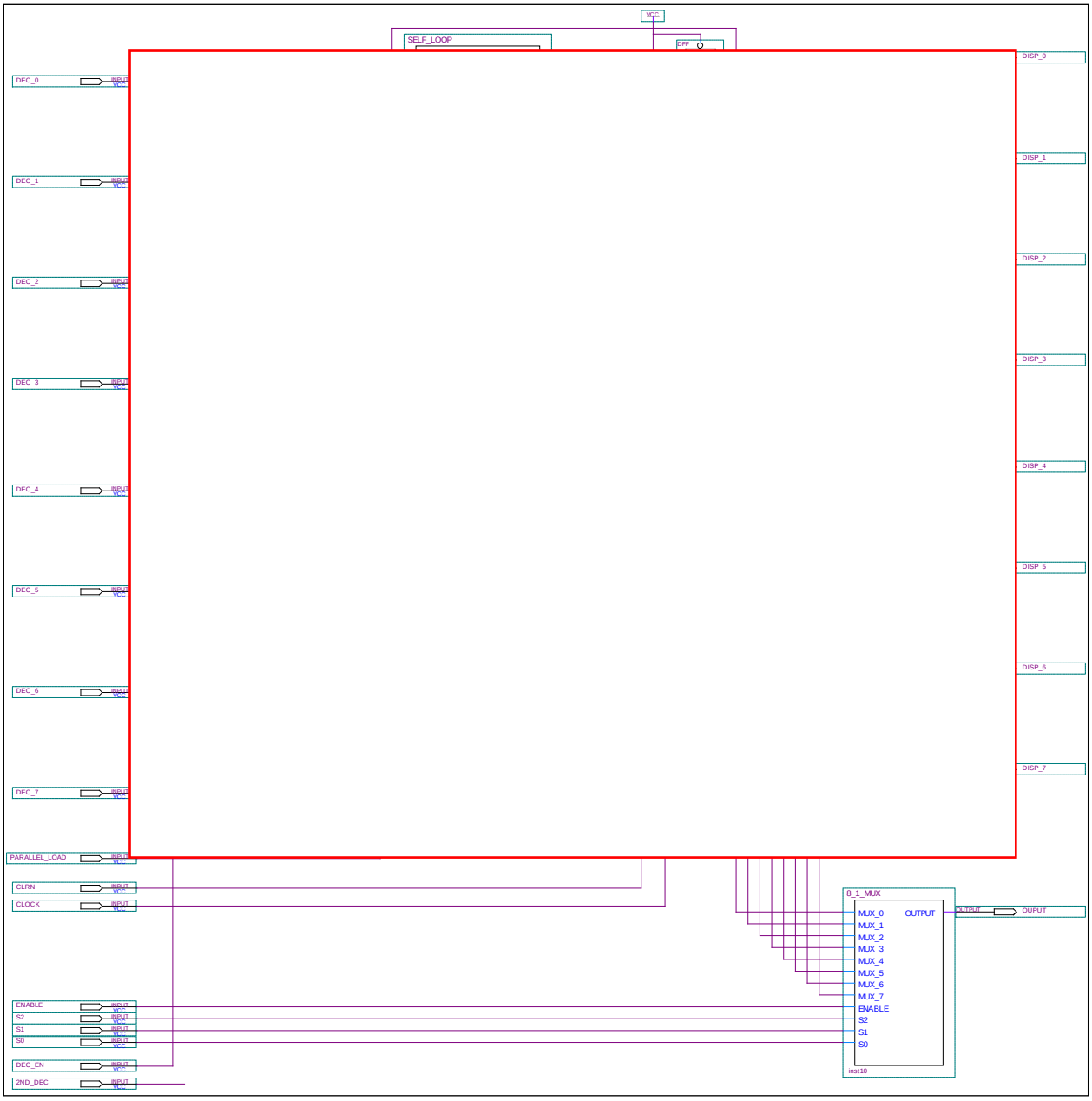

Figure 9: Register

```
; ================================================================================
;                        Test Code for Setting and Clearing Internal Registers
; ================================================================================
00H             DFH     SETB R7.7               ; R7=80,                1000 0000
01H             0FH
02H             DEH     SETB R6.6               ; R6=40,                0100 0000
03H             0EH
04H             DDH     SETB R5.5               ; R5=20,                0010 0000
05H             0DH
06H             DCH     SETB R4.4               ; R4=10,                0001 0000
07H             0CH
08H             DBH     SETB R3.3               ; R3=08,                0000 1000
09H             0BH
0AH             DAH     SETB R2.2               ; R2=04,                0000 0100
0BH             0AH
0CH             D9H     SETB R1.1               ; R1=02,                0000 0010
0DH             09H
0EH             D8H     SETB R0.0               ; R0=01,                0000 0001
0FH             08H
10H             EFH     MOV A,R7                ; A = R7 = 80,          1000 0000
11H             4DH     ORL A,R5                ; A = A0 =              1010 0000
12H             C4H     SWAP A                  ; A = 0A =              0000 1010
13H             34H     ADDC A, #96             ; A = A0 =              1010 0000
14H             96H
15H             C3H     CLR C                   ; Clear Carry
16H             3BH     ADDC A, R3              ; A = A8 =              1010 1000
17H             49H     ORL A, R1               ; A = AA =              1010 1010
18H             F8H     MOV R0, A               ; R0 = AA
19H             74H     MOV A, #55              ; A = 55 =              0101 0101
1AH             55H
1BH             68H     XRL A, R0               ; A = FF =              1111 1111
1CH             F8H     MOV R0, A               ; R0 = FF =             1111 1111
1DH             D8H     SETB_C R0.7             ; R0 = 7F =             0111 1111
1EH             07H
1FH             D8H     SETB_C R0.6             ; R0 = 3F =             0011 1111
20H             06H
21H             D8H     SETB_C R0.5             ; R0 = 1F =             0001 1111
22H             05H
23H             D8H     SETB_C R0.4             ; R0 = 0F =             0000 1111
24H             04H
25H             D8H     SETB_C R0.2             ; R0 = 0B               0000 1011
26H             02H
27H             D8H     SETB_C R0.0             ; R0 = 0A               0000 1010
28H             00H
29H             58H     ANL A, R0               ; A = 0A =              0000 1111
2AH             F8H     MOV R0,A                ; R0 = A = 0A           0000 1010
2BH             74H     MOV A, #00              ; A = 0
2CH             00H
2DH             60H     JZ 03                   ; Jump 3
2EH             03H
32H             80H     SJMP HALT
33H             FEH
```

Conclusion:

After drafting and programming the bit addressability to the WIMP51, a deep insight to the inner processes of the 8051 architecture was gained. Due to our initial lack of knowledge, many timing and gate errors were made; causing a better understanding of the internal operations and general flow of the microprocessor.