CpE 213
Project 1
3/4/2014

Project 1: Modified WIMP51

The purpose of this project was to create a version of the WIMP51 processor in Quartus II using Block Diagram Files, and to become more familiar with how a microprocessor works in the process.

To do this project, two instructions were chosen to add to the original instruction set of the WIMP51 (Table 1): XCH A, Rn and CLR A. The instruction XCH A, Rn has an op code of 11001nnn and should be able to exchange the data from the accumulator and a chosen register in the register bank. The instruction CLR A has an op code of 11100100 and when activated, should set all bits of the accumulator to 0.

MOV A, #D	01110100	dddddddd	A<=D
ADDC A, #D	00110100	dddddddd	C, A<=A+D+C
MOV Rn, A	11111nnn		Rn<=A
MOV A, Rn	11101nnn		A<=Rn
ADDC A, Rn	00111nnn		C, A<=A+Rn+C
ORL A,Rn	01001nnn		A<=A OR Rn
ANL A, Rn	01011nnn		A<=A AND Rn
XRL A, Rn	01101nnn		A<=A XOR Rn
SWAP A	11000100		A<=A(3-0) SWAP A(7-4)
CLR C	11000011		C<=0
SETB C	11010011		C<=1
SJUMP rel	10000000	aaaaaaaa	PC<=PC+rel+1
JZ rel	01100000	aaaaaaaa	PC<=PC+rel+1 if Z
XCH A, Rn	11001nnn		A<=Rn Rn<=A
CLR A	11100100		A<=0

Table 1: Modified WIMP51 Instruction Set

In order to ir ted into the
accumulator nter this, an
additional au
accumulator om the
accumulator le and the
REG write en o that if
the XCH instr . The
AUX_WE was MP51 that
needed to be e data from
the AUX nee ng (Figures

4-6). This was implemented using and AND gate into the MUX_2 branch, which selects the bypass function in the ALU so that when XCH instruction appears, the ALU will allow the data to pass without any changes.

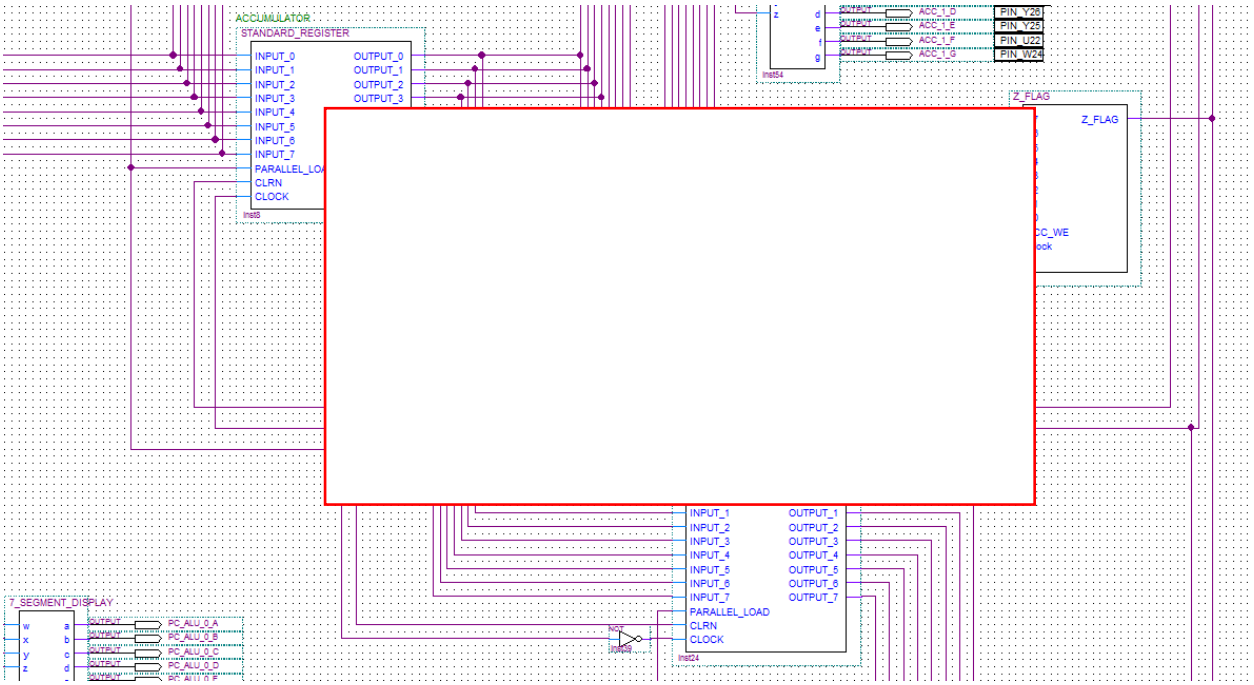

Figure 1: Additional AUX register

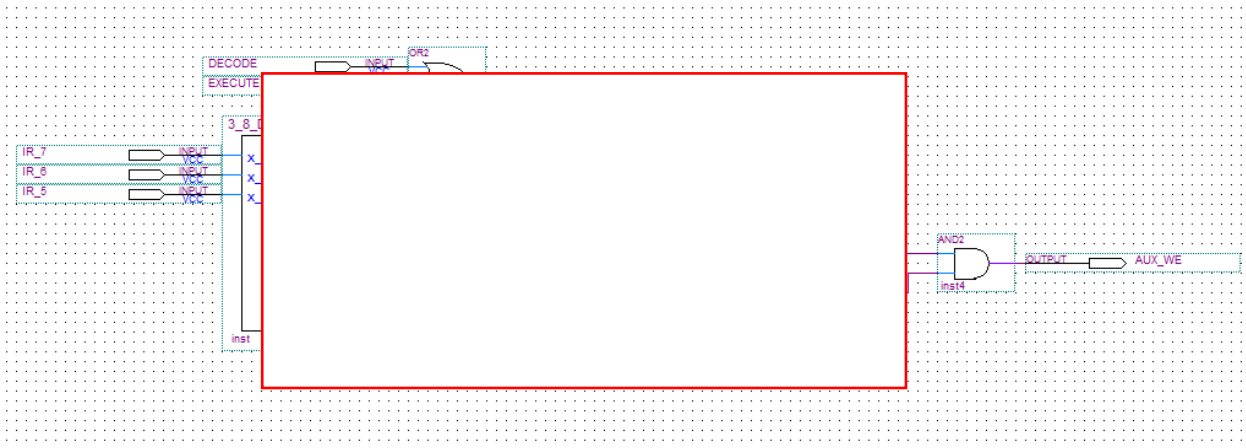

Figure 2: AUX_WE

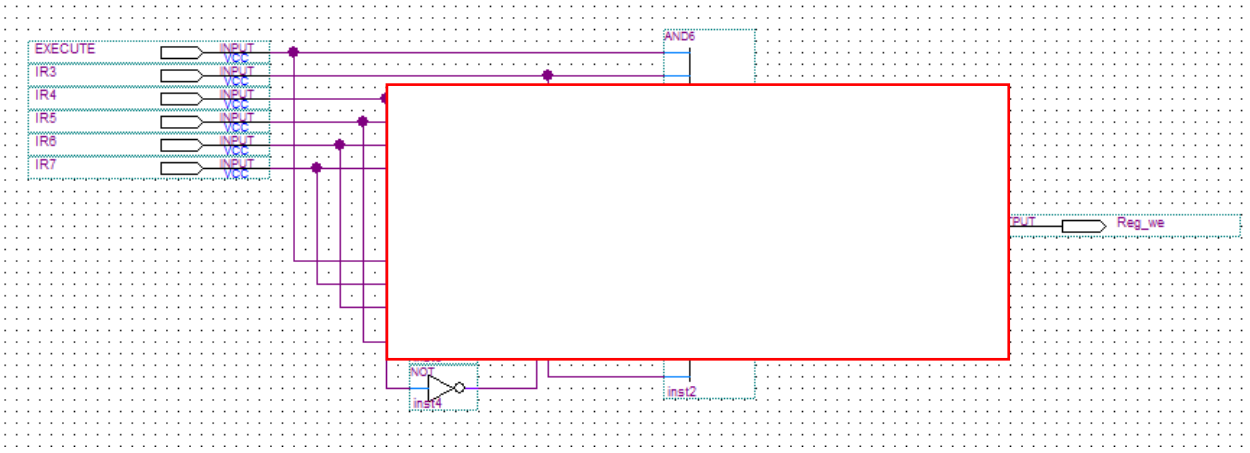

Figure 3: REG_WE

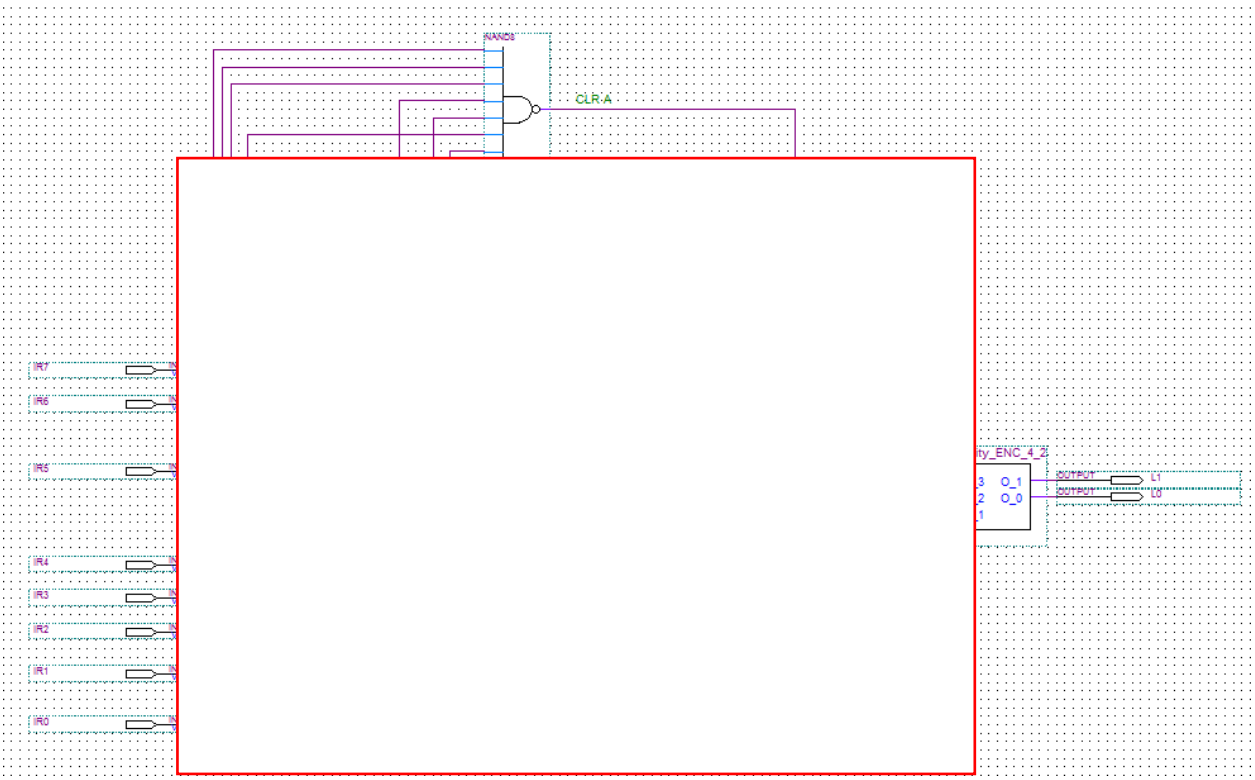

Figure 4: LA_SEL updated with instruction for XCH routed to MUX_2 (bypass adder and logic) and instruction for CLR A routed to all 3 stages of Priority encoder

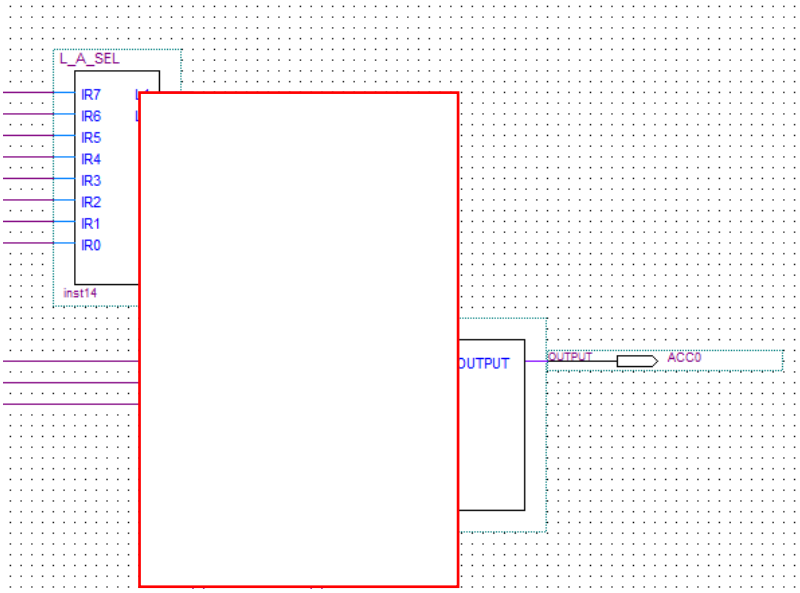

Figure 5: Updated LA_SEL block symbol

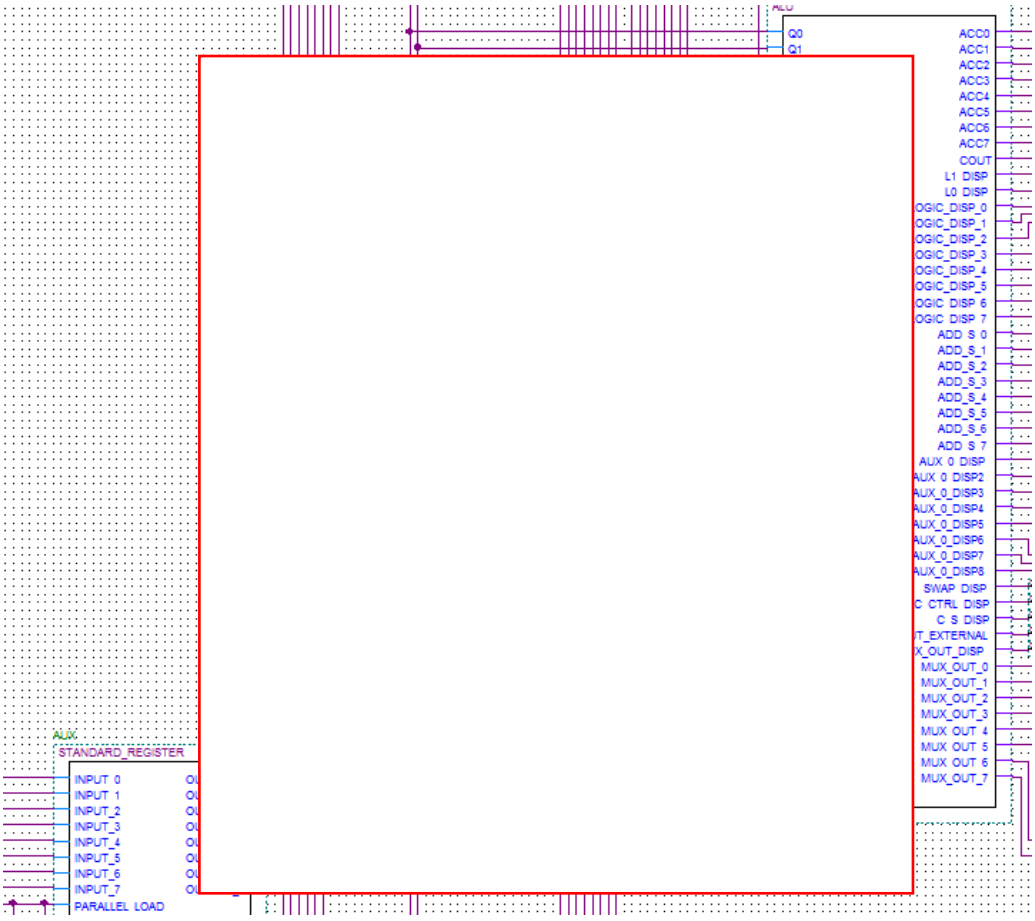

Figure 6: Updated ALU block symbol

In order to implement CLR A, the only update that was made was to the LA_SEL part. This update told the unit that selects the function performed in the ALU that if CLR A instruction shows up, then send zeroes to the accumulator. This was implemented using a NAND gate and AND gates, because if the CLR instruction is not present, that part of the AND gate will be 1, and not interfere with the other logic (Figure 4).

After updating the block diagrams to include the new instructions, a test program was written to make sure that the original instruction set had not been changed, and that the new instructions worked properly as well (Table 2).

Memory Address		**Machine Code**	**Data Value**
00H	MOV A, #80H	74H	
01H		80H	A=80H
02H	MOV R0, A	F8H	R0=80H
03H	SWAP A	C4H	A=08H
04H	XCH A, R0	C8H	A=80H, R1=08H
05H	CLR C	C3H	
06H	ADDC A, R0	38H	A=88H
07H	MOV R1, A	F9H	R1=88H
08H	ORL A, R0	48H	A=88H
09H	MOV R2, A	FAH	R2=88H
0AH	ANL A, R0	58H	A=09H
0BH	MOV R3, A	FBH	R3=08H
0CH	XRL A, R0	68H	A=00H
0DH	MOV R4, A	FCH	R4=00H
0EH	MOV A, R0	E8H	A=80H
0FH	CLR C	C3H	
10H	SETB C	D3H	
11H	CLR A	E4H	A=00H
12H	MOV A, #02H	74H	
13H		01H	A=02H
14H	HERE: CLR C	C3H	
15H	ADDC A, #0FFH	34H	A=01H, A=00H
16H		FFH	
17H	JZ END	60H	
18H		02H	
19H	SJUMP HERE	80H	
1AH		F9H	
1BH	END: SJUMP END	80H	
1CH		FEH	

Table 2: Test Program for Modified WIMP51

Conclusion

CpE 213
Project 1
3/4/2014

After doing this project, I was able to better understand how the WIMP51 works, as well as how to program using machine and assembly level code. Having to trace what was happening to the data during the testing process was the most helpful in understanding how the data flows in the WIMP51, while writing a test program using instructions such as JZ and SJUMP helped me to understand how those instructions work, as well as how to use them. JZ should be used when a countdown of any sort is taking place, and the next action should only take place when the result is 0. SJUMP should be used to repeat an action over and over. For the WIMP51, this is also used at the end of the program so that the program will stop its actions without altering the current data in the registers. Using the Altera board for an interface was helpful because not all parts were visible without manually choosing to view something, such as the accumulator or the register bank; making a conscious decision to view some of these registers was a good way to keep track of what was being viewed. This project was a good stepping stone into understanding how the WIMP51 works.