Introduction to Embedded Systems

Homework 4

Ömer CAM

040190072

Muhammed Velihan Bağcı

040170093

Levent Keskin

040180726

Müştak Erhan YALÇIN

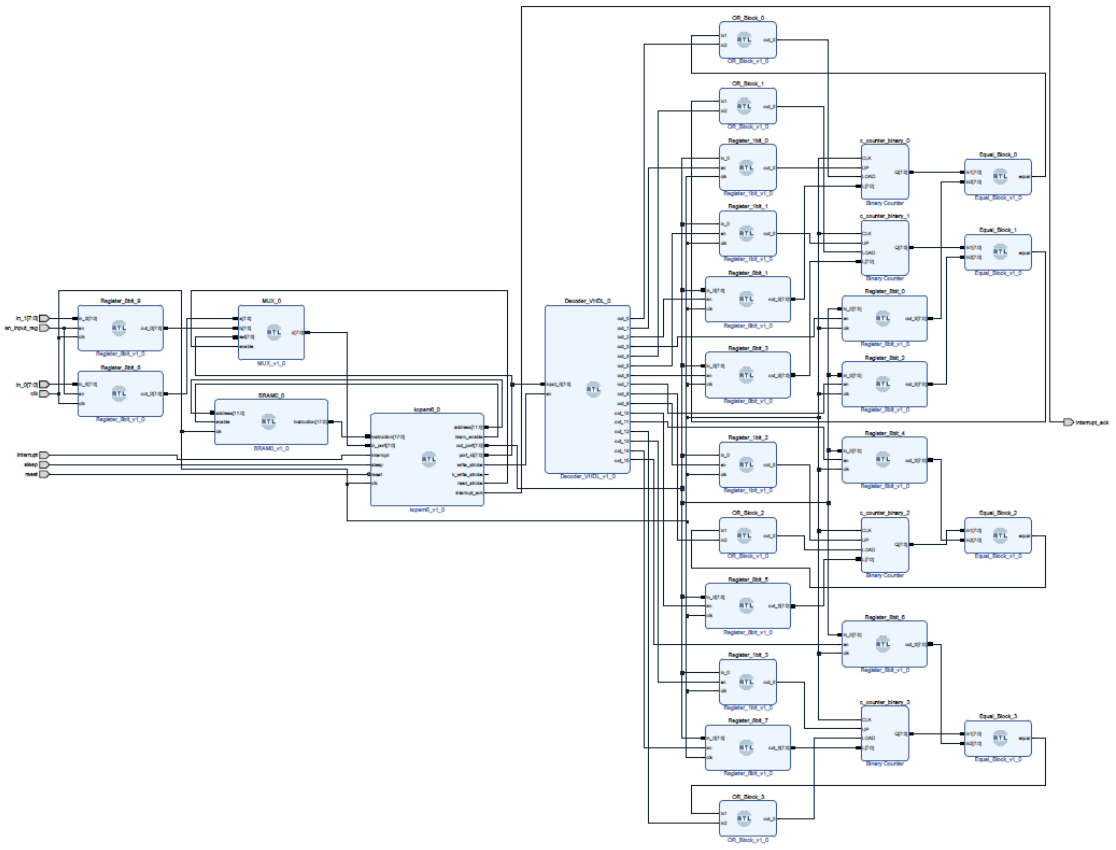
FIDEx Code:

```
#ifdef proc::xPblze6
#set proc::xPblze6::scrpdSize, 64
#set proc::xPblze6::clkFreq, 100000000
#set IOdev::BRAM0::en, TRUE
#set IOdev::BRAM0::type, mem
#set IOdev::BRAM0::size, 4096
#set instmem::pageSize, 4096
#set instmem::pageCount, 1
#set instmem::sharedMemLocation, loMem
#set IOdev::BRAM0::value, instMem
#set IOdev::BRAM0::vhdlEn, TRUE
#set IOdev::BRAM0::vhdlEntityName, "BRAM0"
#set IOdev::BRAM0::vhdlTmplFile, "ROM_form.vhd"
#set IOdev::BRAM0::vhdlTargetFile, "BRAM0.vhd"
#endif
#ORG ADDR, 0
        int     enable
        call    initlize
loop:   jump    loop;
parse_ins:
        load    s0, 0
        rdprt   s1, (s0)
        load    s3, 31
        and     s1, s3
        load    s0, 1
        rdprt   s2, (s0)
        ret
execute_ins:
        wrprt   s2, (s1)
        ret
```

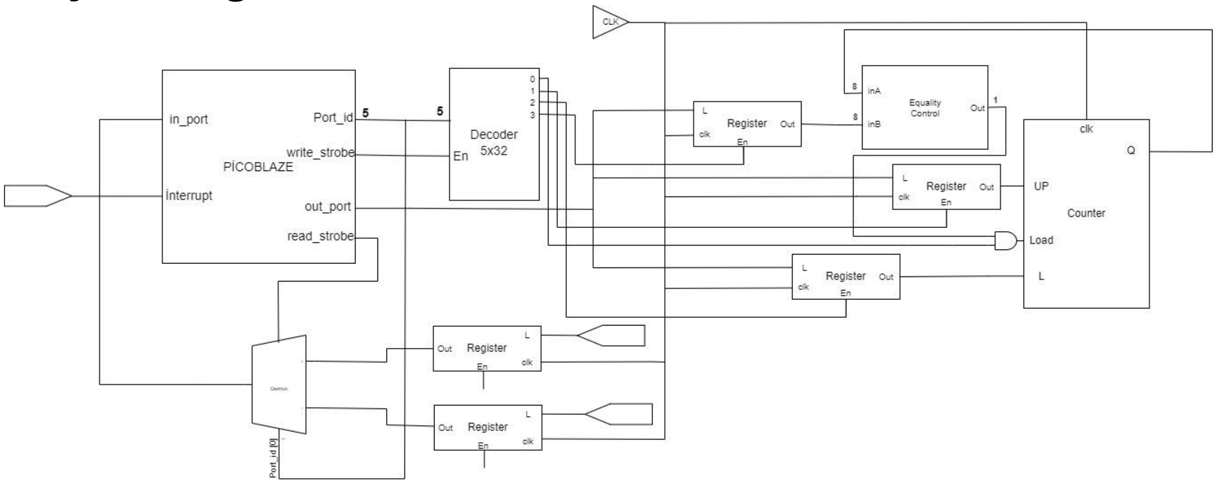
This assembly code, enables interrupts and initializes the 4 counters as their start value as 0, final value as 0xFF and make them counts up. When an interrupt is triggered, reads the first register, which holds instruction, and parse the instruction and saves in s1 register. Then, it reads the second register which holds the new value of the target register, and saves it in s2 register. Then writes s2 value from output and makes port id as s1. After that, program enables interrupts again and returns to the main loop.

```
initlize:
        load    s1, b10000
        load    s2, 0xFF
        wrprt   s2, (s1)
        load    s1, b01000
        load    s2, 0
        wrprt   s2, (s1)
        load    s1, b00100
        load    s2, 1
        wrprt   s2, (s1)
        load    s1, b00000
        wrprt   s2, (s1)
        load    s1, b10001
        load    s2, 0xFF
        wrprt   s2, (s1)
        load    s1, b01001
        load    s2, 0
        wrprt   s2, (s1)
        load    s1, b00101
        load    s2, 1
        wrprt   s2, (s1)
        load    s1, b00001
        wrprt   s2, (s1)
        load    s1, b10010
        load    s2, 0xFF
        wrprt   s2, (s1)
        load    s1, b01010
        load    s2, 0
        wrprt   s2, (s1)
        load    s1, b00110
        load    s2, 1
        wrprt   s2, (s1)
        load    s1, b00010
        wrprt   s2, (s1)
        load    s1, b10011
        load    s2, 0xFF
        wrprt   s2, (s1)
        load    s1, b01011
        load    s2, 0
        wrprt   s2, (s1)
        load    s1, b00111
        load    s2, 1
        wrprt   s2, (s1)
        load    s1, b00011
        wrprt   s2, (s1)
        ret
isr:
        call    parse_ins
        call    execute_ins
        reti    enable
#ORG ADDR, 0xFFF
        jump    isr
```

RTL Schematic:



Project Design:



A Mux was placed in the input port of Picoblaze in the project design. Mux's inputs are designed to receive 8 bit instruction and 8 bit data. Mux's control input is connected to Picoblaze’s port id. Thus, Picoblaze can choose whether to read 8 bit instruction register or 8 bit data register according to the port id’s least significant bit when interrupt is triggered. Picoblaze's port id output is connected to a 5x32 Decoder but 16 of 32 output pins are used. This Decoder briefly sends signals that determine which Counter to work with and what action to take with this Counter by enabling relavent register.

A block (Equality Control) is written. It has two inputs: Max register value and Output of related counter. It checks whether ıt’s inputs are equal or not. If they are equal then sends high (1) signal, else ıt sends low (0) signal to the output. Up/Down register holds 1 bit data other registers hold 8 bit data. When up/down register is 1, counter counts up, else it counts down. Counter’s load input is a control input. When load is 1 next output value is set to the L[7:0] value, else ıt does nothing.

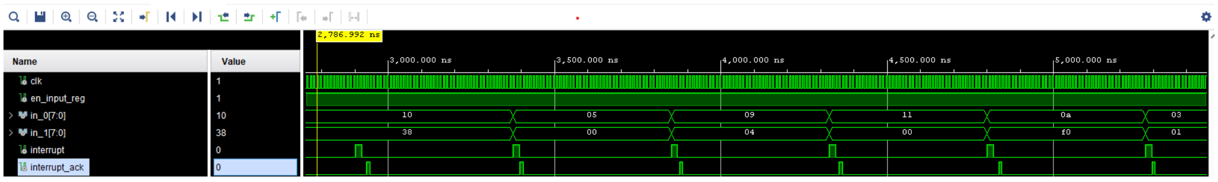
PORT_ID[7:5]	**PORT_ID[4:2]**		**PORT_ID[1:0]**	
NOT Interested	000	Reset	00	Select 1st counter
	001	Up/Down	01	Select 2nd counter
	010	Min	10	Select 3rd counter
	100	Max	11	Select 4th counter
	XXX	Not Interested		

When writing data, port id’s (instruction) most significant 3 bit are not used, port id’s[4:2] bit are correspond to operations and port id’s[1:0] bit are correspond to counters as shown above table.

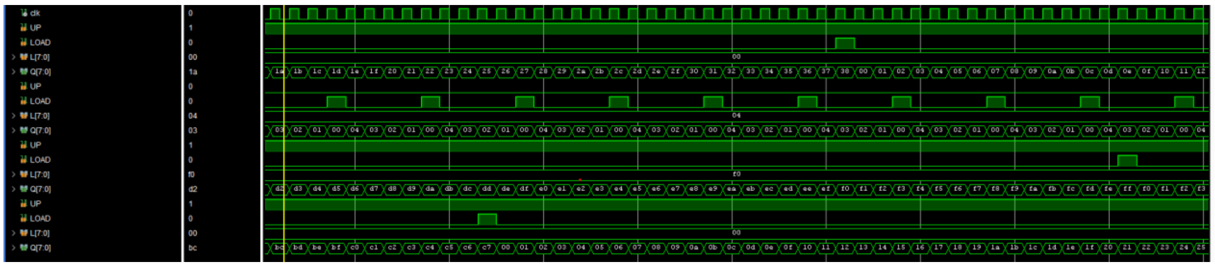
PORT_ID[0]	
0	Selects Instruction
1	Selects Value

When reading data, port id’s least significant bit is used. It corresponds what register to read as shown above table.

Simulation Result:



This image shows interrupt signals, 8 bit instruction register values and 8 bit data register values.



This image shows counter values. First counter counts up from 0 to 0x38. Second counter counts down from 0x4 to 0. Third counter counts up from 0xf0 to 0xff. Fourth counter counts up 0 to 0xff but due to instruction which is reset, it resets its value at 0xc7 which is the value the register have when the interrupt is done.

Conclusion:

Picoblaze reads successfully 8 bit instruction and 8 bit data value from registers shown in the first image of simulation result. And, second image of simulation result show it executes the instructions properly and set the value of related register. Thus, result of the simulation is as expected. It works properly. The design meets requirements.