ISTANBUL TECHNICAL UNIVERSITY

ELECTRONICS AND COMMUNICATION
ENGINEERING

INTRODUCTION TO EMBEDDED SYSTEMS

(EHB 326E)

TERM PROJECT

2019-2020 Autumn

MULTIPLE INTERRUPT HANDLING SYSTEM FOR PICOBLAZE MICROCONTROLLER

Osman Meriç **Mehmet Fatih Gülakar**

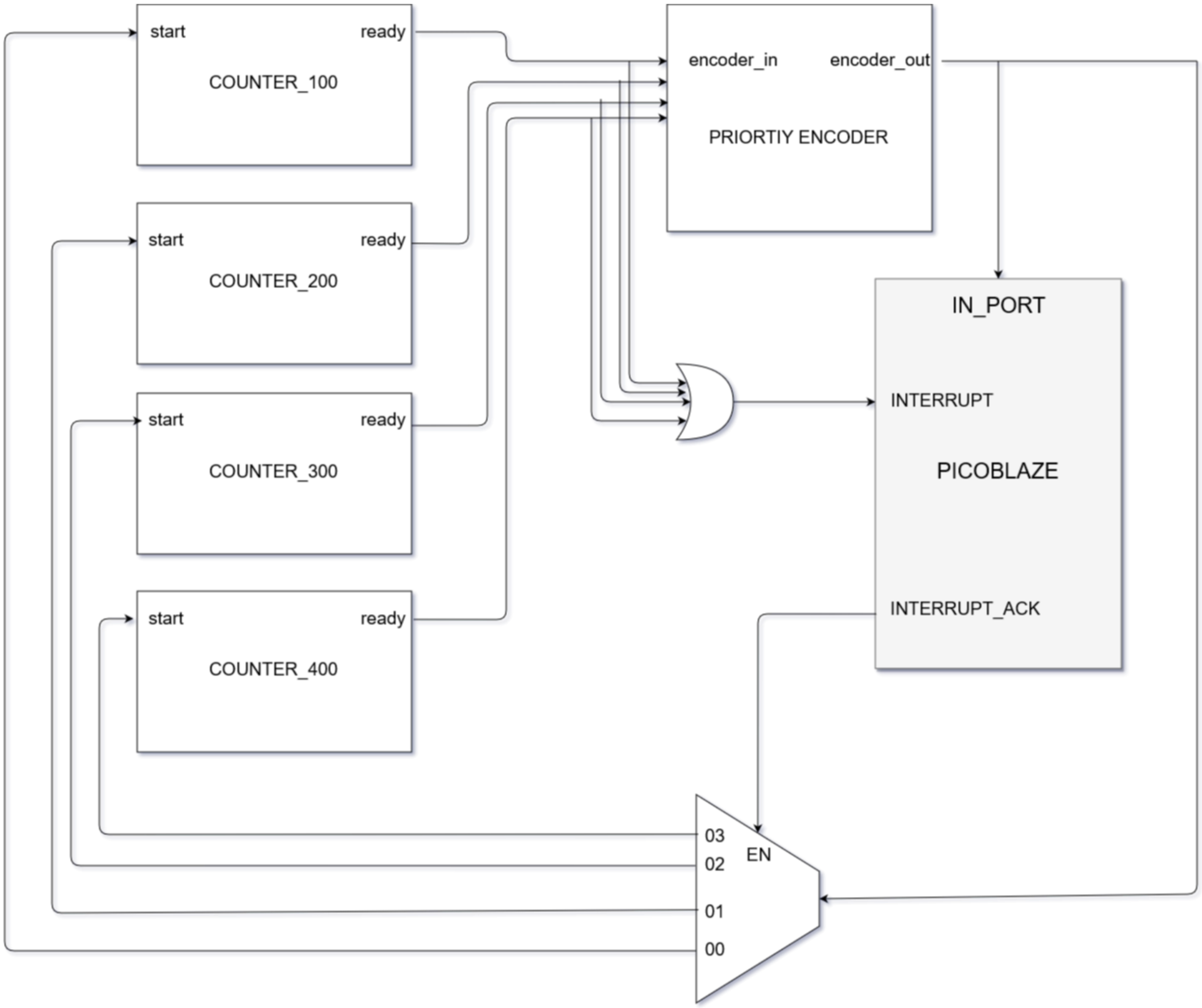
040150022 **040150015**

January 9th 2020

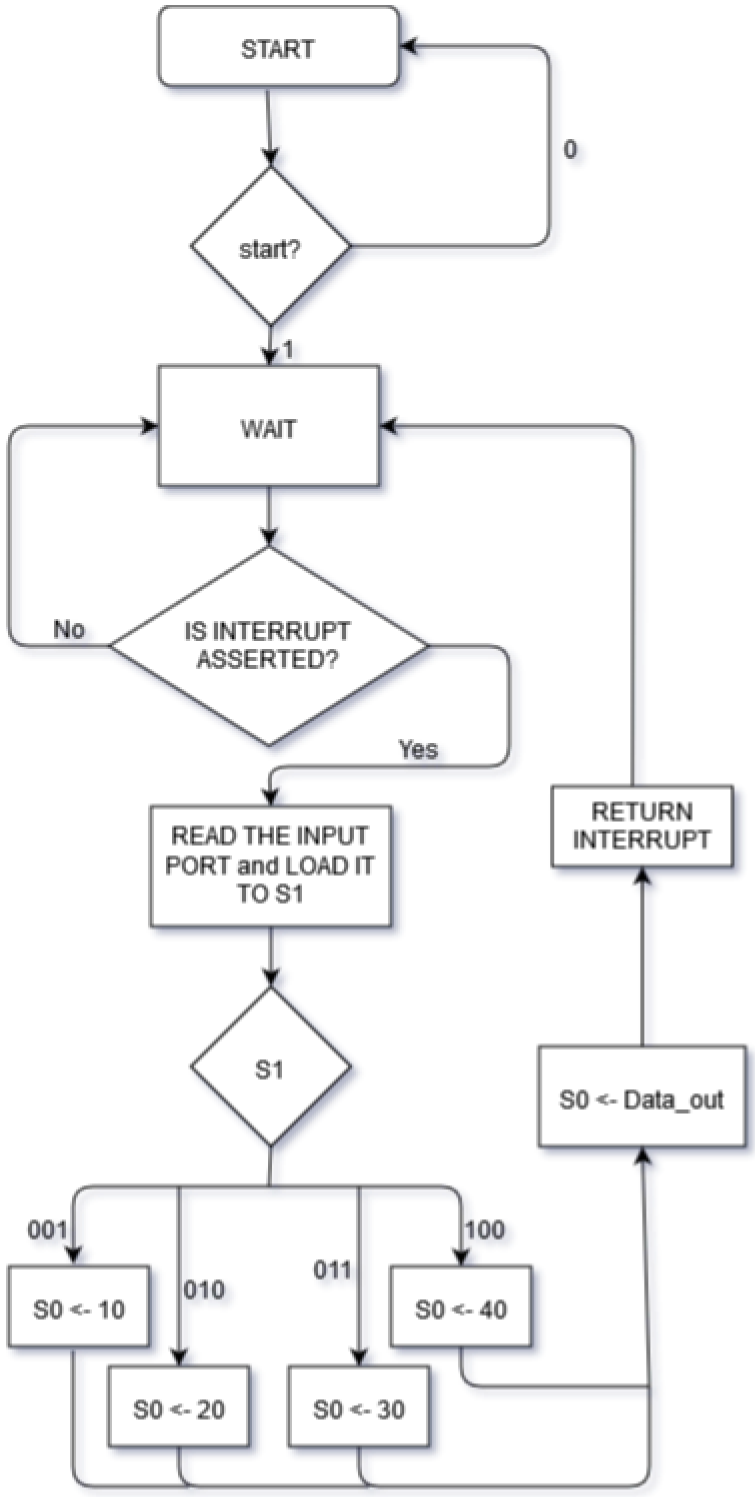
1) AIM

Interrupt is an input signal which alerts the processor and serves as a request for the processor to the interrupt the currently executing code, so that the event can be processed in a timely manner. Xilinx's Picoblaze microcontroller has an interrupt which has single bit only. Therefore, a designed system need some kind of modification to handle if there is more than one interrupt signal.

2) BLOCK DIAGRAM OF THE SYSTEM

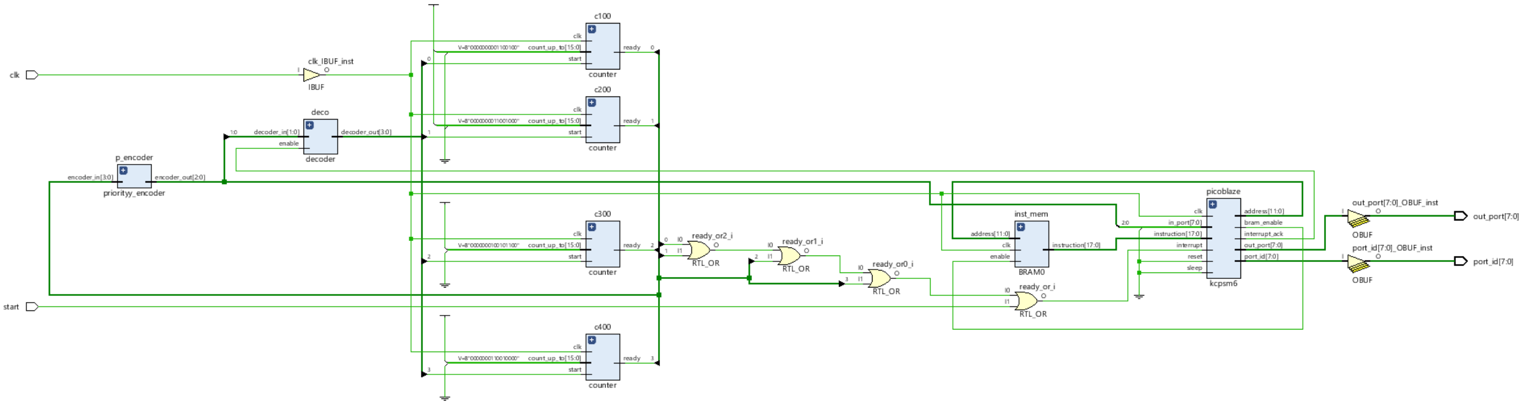


3) ALGORITHM

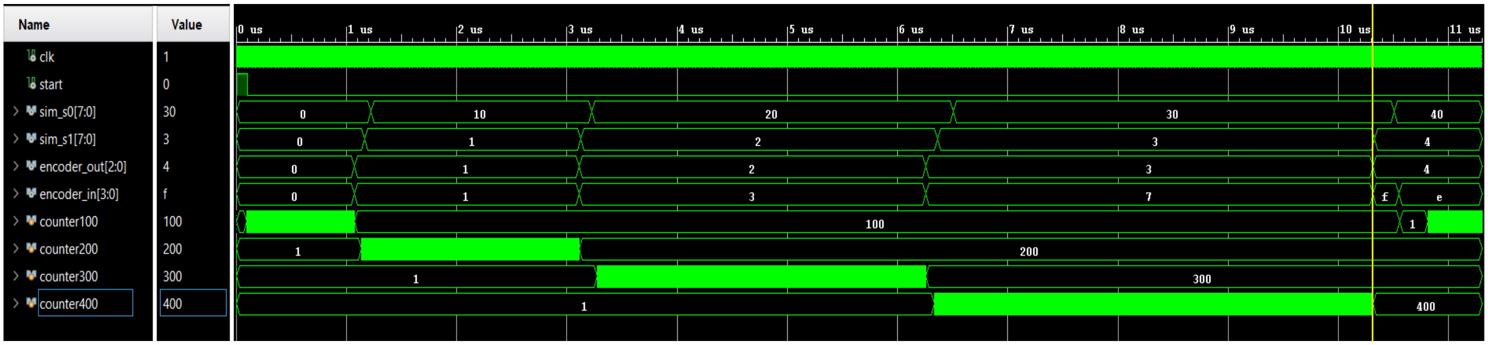


When an interrupt is asserted, Picoblaze reads the input_port and writes the value to S1, then according to the value of S1, program determines which subroutine will be called. In each subroutine, different values are written to S0. (simply 10 times s1). Then, value inside S0 register is loaded to Data_out port of Picoblaze, and program returns the interrupt service routine.

4) SCHEMATICS GENERATED BY VIVADO



5) SIMULATION RESULTS



Name	Value
clk	1
start	0
sim_s0[7:0]	30
sim_s1[7:0]	3
encoder_out[2:0]	4
encoder_in[3:0]	f
counter100	100
counter200	200
counter300	300
counter400	400

As can be understood from the simulation results, during the first 1.25 microseconds, counter100 starts to count firstly. When it finishes its counting, it makes its “ready” output high, encoder_out becomes 1, therefore an interrupt is asserted. Picoblaze checks its in_port and stores it to the s1 register, to determine the interrupt service routine to run. During the interrupt service routine, s1 is compared with 1, 2, 3 and 4 separately, s0 is changed thereafter. The reason for the short delay between s1 and s0 is the comparison of s1 with various values, which is the few clock cycles, depending on the value of s1.

6) ASSEMBLY CODE

```
INT ENABLE
loop:
        JUMP loop

ISR:
        RDPRT s1, 100

        COMP s1, 1
        JUMP Z l100

        COMP s1, 2
        JUMP Z l200

        COMP s1, 3
        JUMP Z l300

        COMP s1, 4
        JUMP Z l400

        RETI ENABLE

l100:
        LOAD s0, 10
        RETI ENABLE
l200:
        LOAD s0, 20
        RETI ENABLE
l300:
        LOAD s0, 30
        RETI ENABLE
l400:
        LOAD s0, 40
        RETI ENABLE


#ORG ADDR, 4095
JUMP ISR
```