ECGR 4146 PROJECT 1 PRESENTATION

GROUP 1

MARCH 28, 2018

Dhivya Sivalingam
Ryan Swaim
Andrew Szoke
Swathi Thirunarayanan

PROJECT OVERVIEW

Purpose: to “design, simulate, and synthesize a behavioral embedded microcontroller” in VHDL

Modules created:

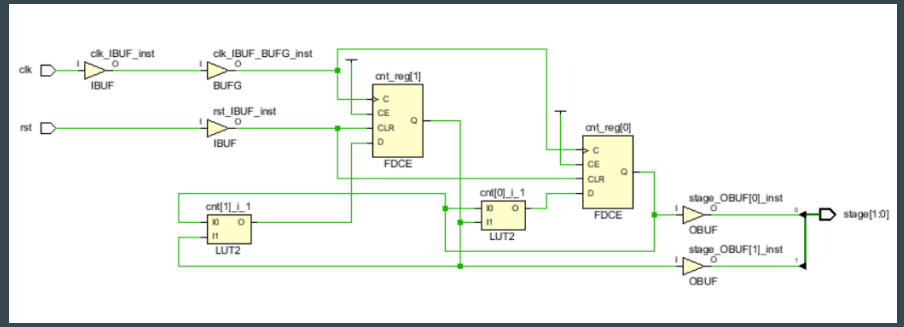
Registers	PC	Memory
Decode Logic	IR	IM
ALU	Muxes	Stage Counter

STAGE COUNTER

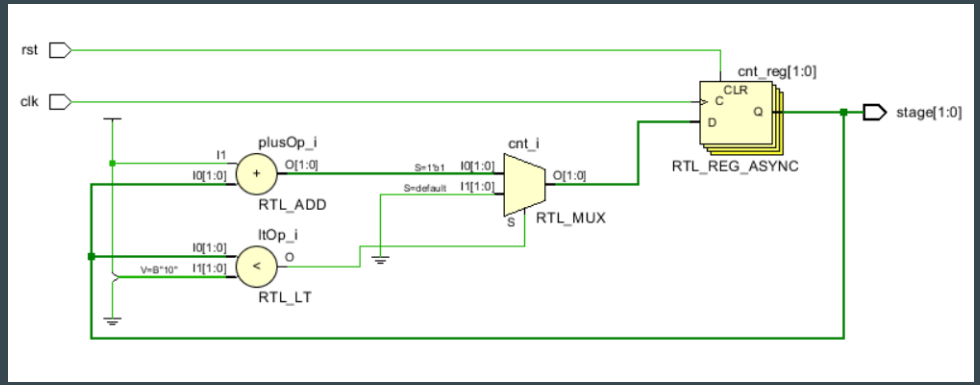
- Keeps track of which of the three instruction stages the processor is in
- **Stage 0:** Instruction is fetched by IR from the memory through the **`datain`** bus, PC is incremented
- **Stage 1:** Immediate value is loaded in the Immediate register with value from its memory based on **`IRbit7`**.
- **Stage 2:** Decode logic decodes the opcodes and corresponding operation is executed.

```
architecture Behavioral of phase_counter is
signal cnt,tmp : std_logic_vector(1 downto 0);
signal resetsig : std_logic;
begin
  process(clk,rst)
    begin
        if(rst ='1')then
            cnt <= "11";
        else if(rising_edge(clk)) then
            if(cnt<"10") then
            cnt <= cnt + 1;
            else cnt <= "00";
   --         if(resetsig = '1') then
   --         cnt <= "00";
            end if;
        end if;
    end if;
    end process;
--  resetsig <= cnt(1) and cnt(0);
     tmp <= "00" when cnt = "11" else cnt;
     stage <= tmp;
end Behavioral;
```

STAGE COUNTER (cont'd)

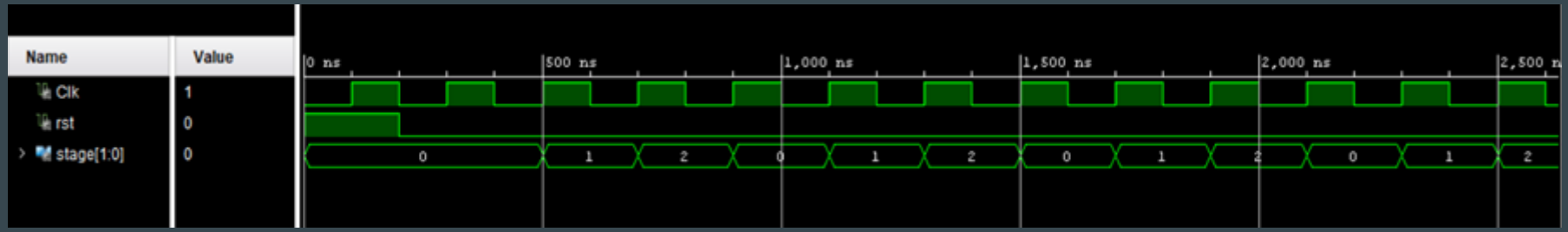


SYNTHESIZED DESIGN



ELABORATED DESIGN

STAGE COUNTER (cont’d)



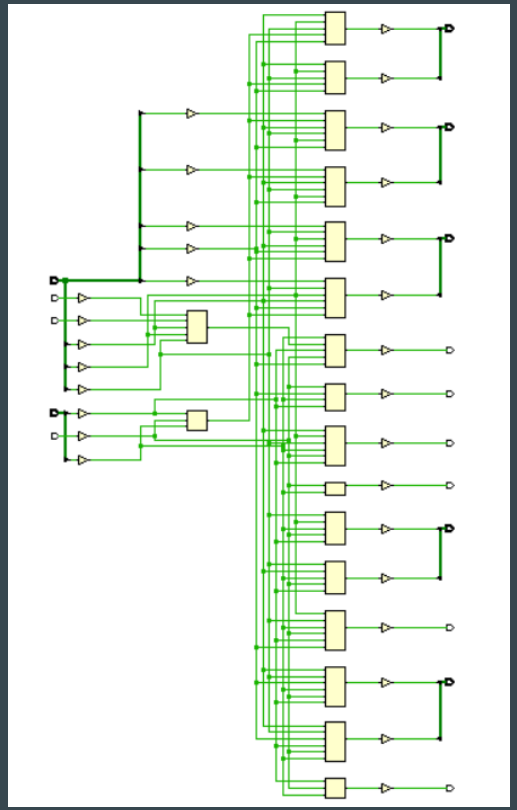
TESTBENCH RESULTS

MEMORY

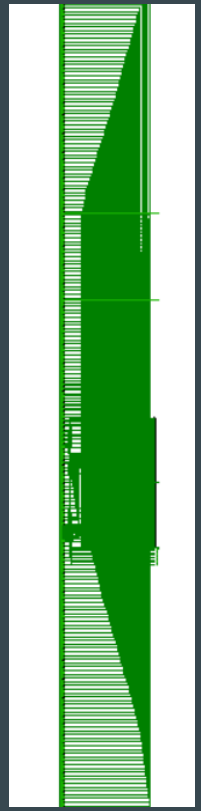
- “Array of arrays” of bus width (8 bits)
- Edge-triggered clock and reset inputs
- Toggled by **`readwrite`** flag from the CPU
- Synchronous write and asynchronous read

```
process(clk,rst) is
    begin
       if Rst = '1' then
                -- Clear Memory on Reset
        ramArray <= (others => (others => '0'));
       else
       ramArray(0) <= x"e4";
       ramArray(1) <= x"00";
       ramArray(2) <= x"e0";
       ramArray(3) <= x"80";
       ramArray(4) <= x"48";
       ramArray(5) <= x"88";
       ramArray(6) <= x"0d";
       ramArray(7) <= x"26";
       ramArray(8) <= x"ec";
       ramArray(9) <= x"01";
       ramArray(10) <= x"23";
       ramArray(11) <= x"ff";
       ramArray(12) <= x"04";
       ramArray(13) <= x"d4";
       ramArray(14) <= x"40";
       ramArray(15) <= x"ff";
       ramArray(16) <= x"0f";
       ramArray(128) <= x"02";
       ramArray(129) <= x"01";
       ramArray(130) <= x"00";
           if rising_edge(Clk) then
    if readwrite = '1' then
    ramArray(to_integer(unsigned(Address))) <= Dataout;
    end if;
    end if;
    end if;
    end process;
   Datain <= ramArray(to_integer(unsigned(Address)));
end RAM;
```

MEMORY (cont'd)

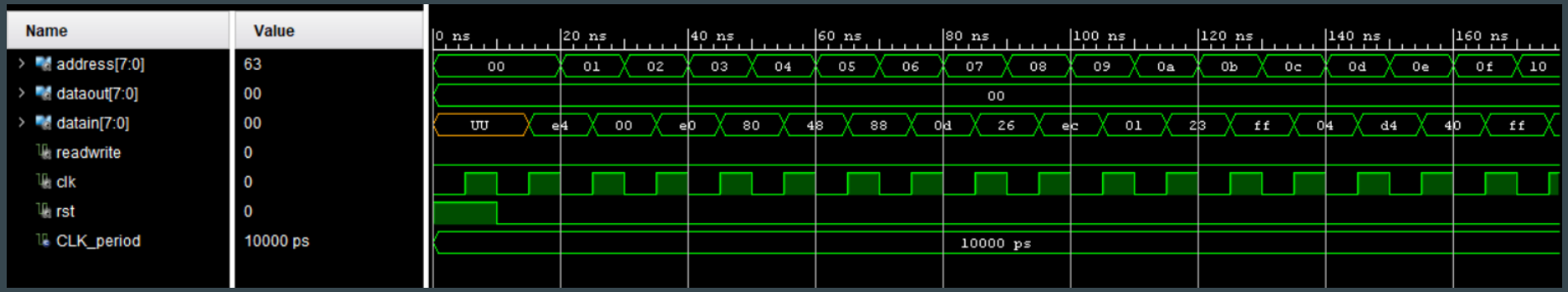


SYNTHESIZED DESIGN



ELABORATED

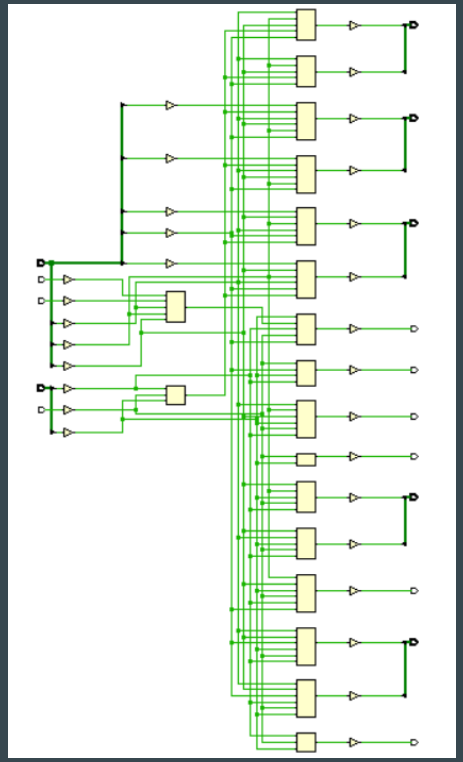
MEMORY (cont'd)



TESTBENCH RESULTS

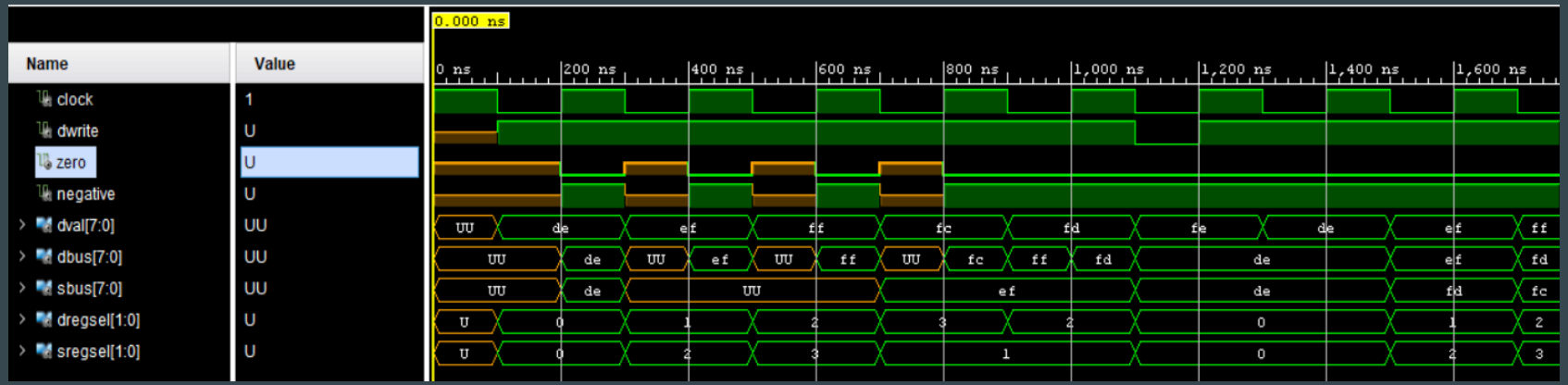
REGISTERS

- 8-bit, edge-triggered
- Each register in **`register_file`** entity are within a **`registerfs`** wrapper
  - **`register_file:`** It has **`dwrite`** (acts as enable), **`dval`** (output of register mux) as input. **`zero`**, **`negative`** and registers **`r0`** – **`r3`** are the output.
  - **`Registerfs`**: Synchronous and registers are updated on rising edge of clock.



SYNTHESIZED DESIGN

REGISTERS (cont'd)



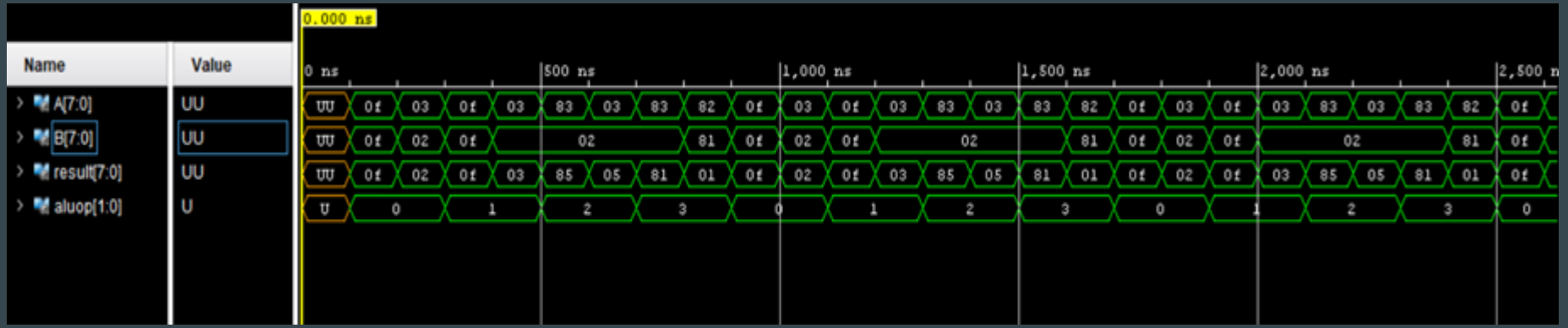
TESTBENCH RESULTS

ALU

- Two 8-bit inputs **`A`** and **`B`** and one 2-bit **`aluop`** selector
- One 8-bit **`result`** vector

```
when "10" =>
res := to_integer(signed(A)) + to_integer(signed(B));
result <= std_logic_vector(to_signed(res,8));

when "11" =>
temp1 := resize(signed(A),temp1'length);
temp2 := resize(signed(B),temp2'length);
tempr := temp1 - temp2;
result <= std_logic_vector(resize(signed(tempr),result'length));


when others =>
result <= A;

end case;
end process;
end Behavioral;
```

ALU (cont'd)



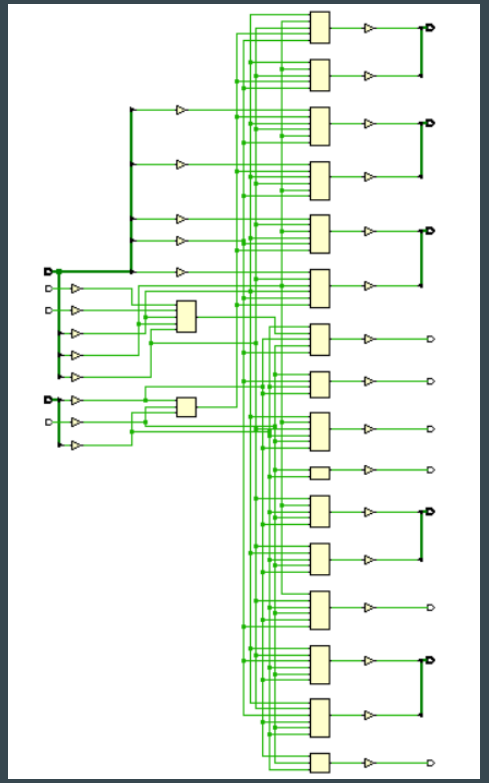
TESTBENCH
RESULTS

DECODE LOGIC

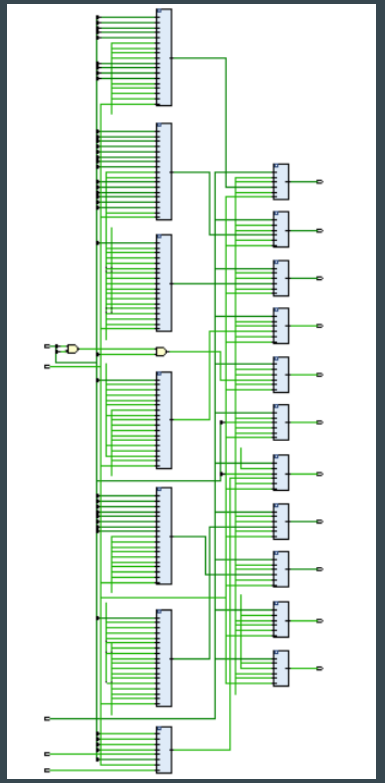
- Controlling logic of the simulation
- Contains four multiplexor components and one **`jump_logic`** component
- Breaks instructions down into their individual components
- Maps the components to the muxes, selects output and returns it

```
34  entity decode_logic is
35      Port (
36              instruction     :       in  STD_LOGIC_VECTOR (7 downto 0);
37              zero            :       in  STD_LOGIC;
38              negative        :       in  STD_LOGIC;
39              rst1            :       in  STD_LOGIC;
40              stage           :       in  STD_LOGIC_VECTOR (1 downto 0);
41              addrsel         :       out STD_LOGIC_VECTOR (1 downto 0);
42              irload          :       out STD_LOGIC;
43              imload          :       out STD_LOGIC;
44              regsel          :       out STD_LOGIC_VECTOR (1 downto 0);
45              dwrite          :       out STD_LOGIC;
46              aluop           :       out STD_LOGIC_VECTOR (1 downto 0);
47              readwrite       :       out STD_LOGIC;
48              pcsel           :       out STD_LOGIC;
49              pcload          :       out STD_LOGIC;
50              sregsel         :       out STD_LOGIC_VECTOR (1 downto 0);
51              dregsel         :       out STD_LOGIC_VECTOR (1 downto 0)
52      );
53  end decode_logic;
```

DECODE LOGIC (cont'd)



SYNTHESIZED DESIGN



ELABORATED DESIGN

DECODE LOGIC (cont'd)

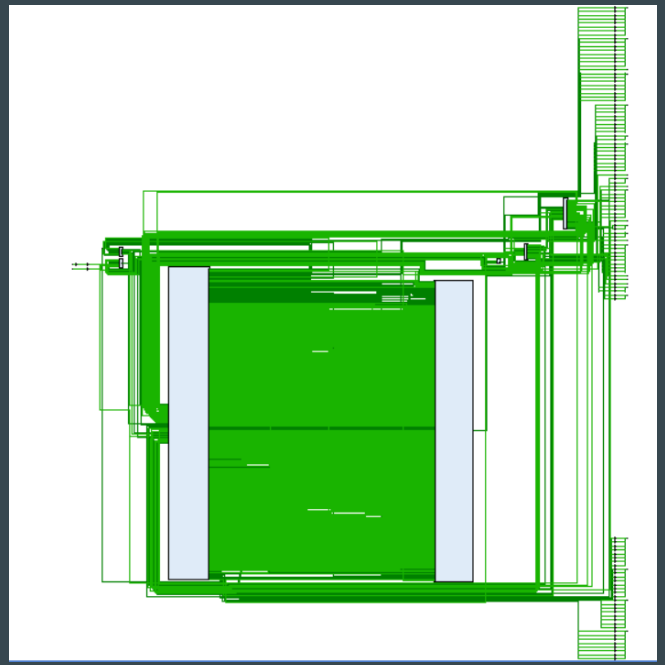
TESTBENCH

MICROCONTROLLER

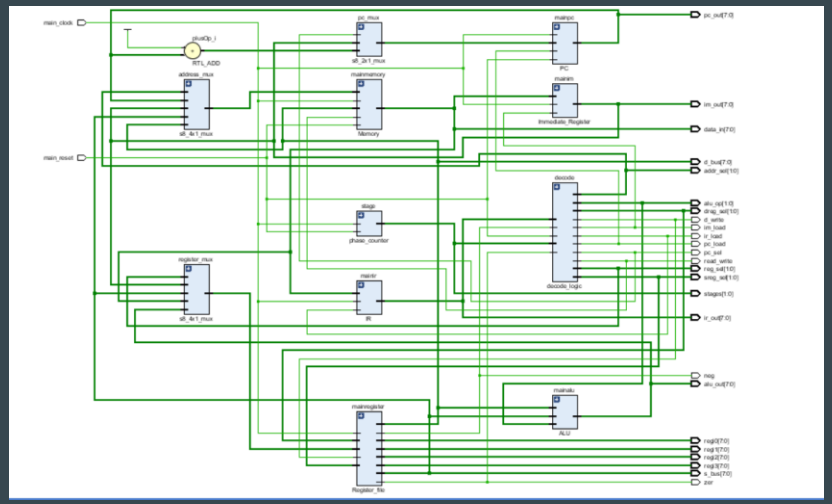
- Highest level of simulation, connects all lower-level outputs to muxes
- Components within the architecture:

```
        phase_counter                   s8_2x1_mux
PC

        ALU
s8_4x1_mux                      decode_logic

        register_file                   IR
                memory
```

MICROCONTROLLER (cont'd)



SYNTHESIZED DESIGN



ELABORATED DESIGN

ISSUES FACED

- Stage counter was unlike normal counter (increments after reset 0 value). In Stage counter, the counter should start from 0 after reset.(During reset counter is 0. So the counter value should be in the following manner if we reset the counter first:

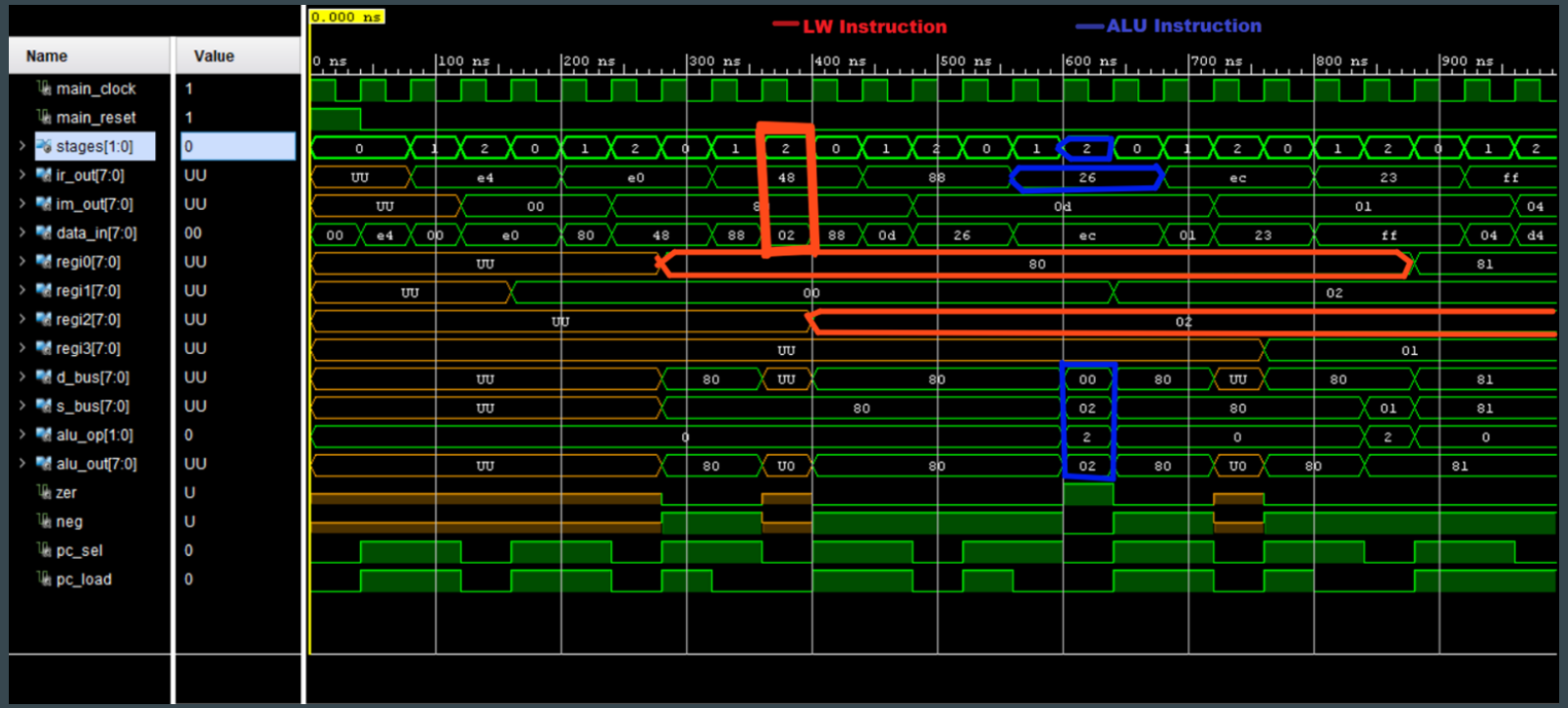
	Reset	Stages
Normal counter	0	1 2 3 .....
Stage Counter	0	0 1 2 0 1 2 ....

- Understanding output of IR and IM register. The output of IR and IM registers were delayed by one clock cycle. That is, the output of stage 0 was found during next clock cycle (stage 1) in IR and the output of stage 1 was found during stage 2 in IM register.

EXAMPLE CODE

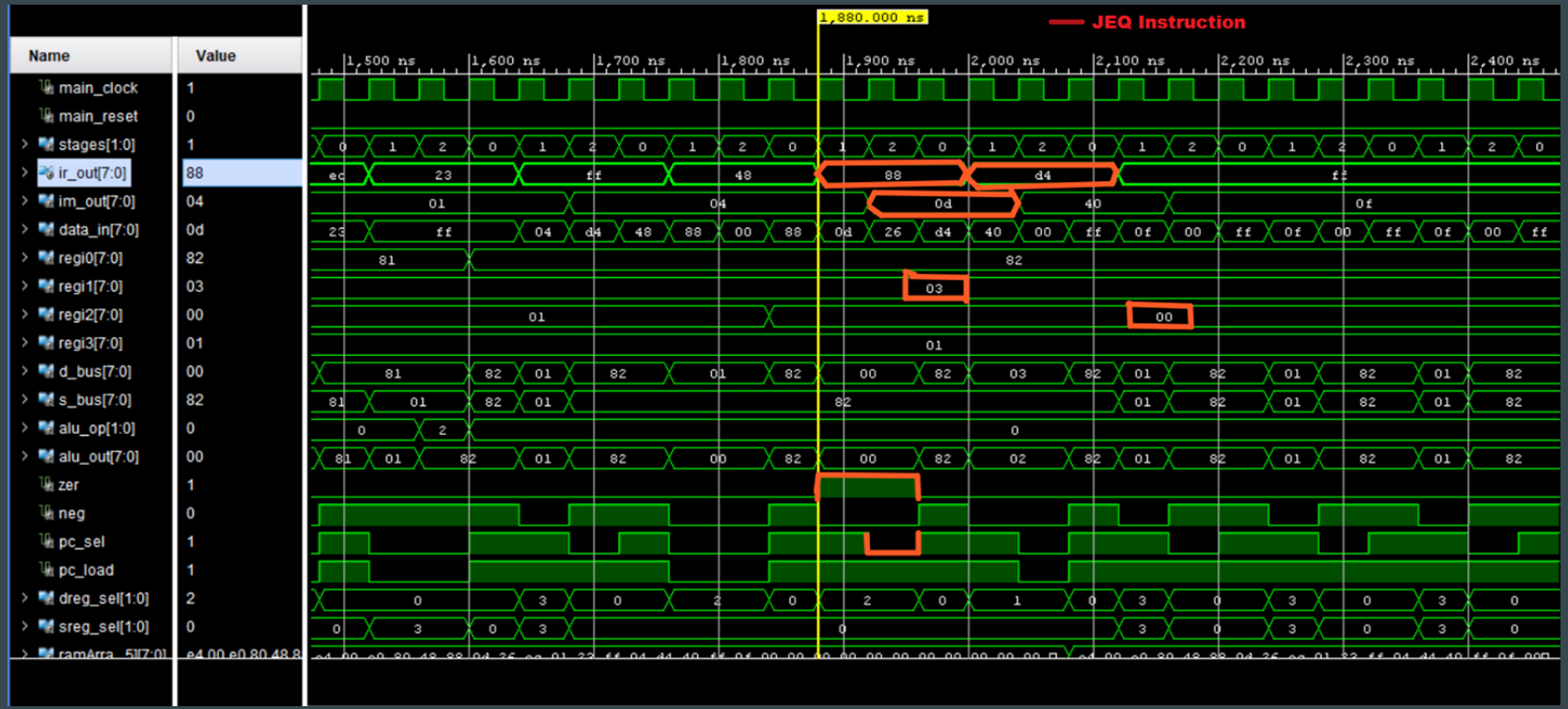
LI R1,0x00	e4 00
LI R0,0x80	e0 80
LW R2, (R0)	48
JEQ R2, end	88 0d
ADD R1, R2	26
LI R3, 0x01	ec 01
ADD R0, R3	23
JMP loop	ff 04
SW R1, 0x40	d4 40
JMP inf	ff 0f

MICROCONTROLLER (cont'd)



WAVEFORM

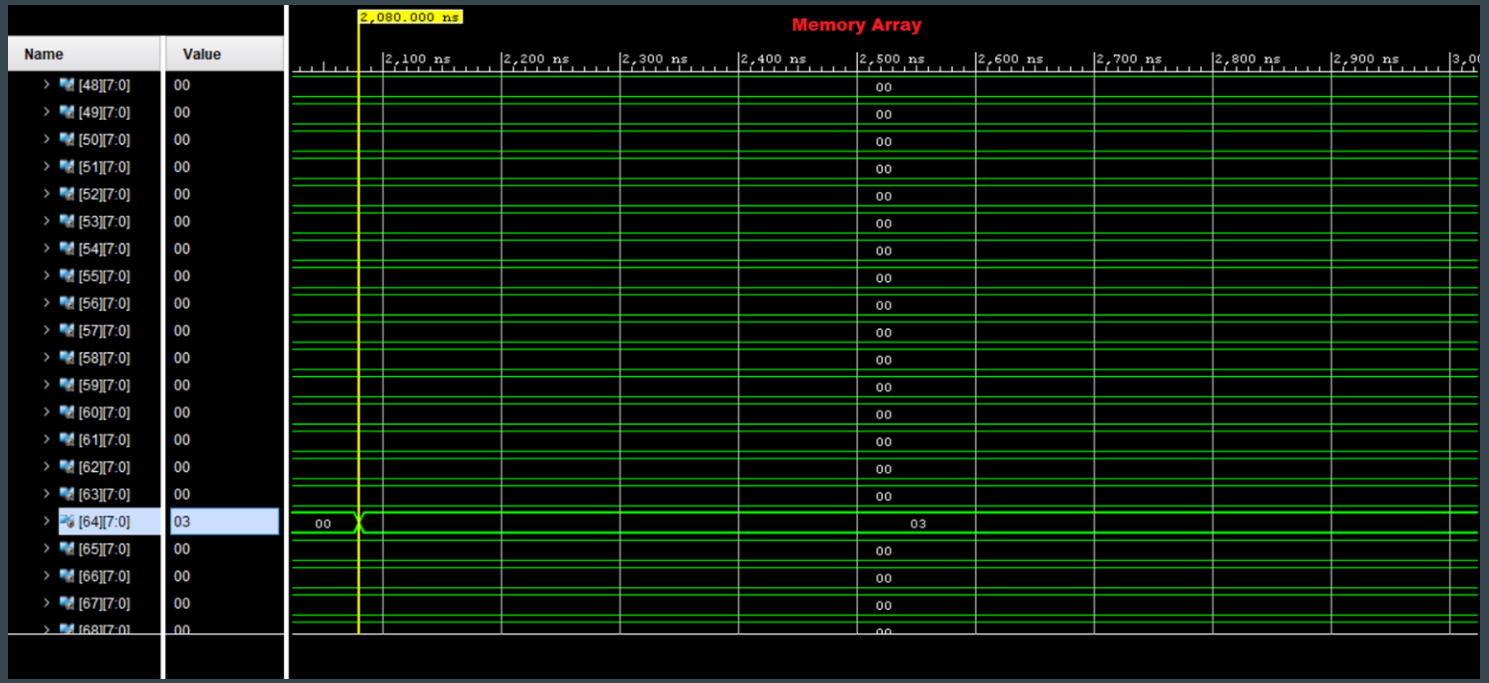
MICROCONTROLLER (cont'd)



WAVEFORM

RESULTS

MICROCONTROLLER (cont'd)



WAVEFORM

QUESTIONS?