Hardware System Design Mid-term Exam Answer

2019. 04. 16

Q1. (Matrix-vector multiplication in hardware, 10 pts)

Answer)
(1) Total execution cycles = (N/T)*(N/T)*(C+T+T)

First we group N rows of the matrix into N/T groups. Each each group of T rows are decomposed to N/T tiles. For each tile, it takes C (for data transfer from CPU) + T (T times of writing T weights to T buffers in parallel) + T (T times of vector element broadcast and MAC operation).

(2) Total execution cycles = $(N/T)^2*(C+T+T) = 10000/T^2 * (10+2T) <= 5{,}000 \rightarrow T^2 - 4T - 20 >= 0$

T is a positive integer. Thus, the minimum T satisfying the inequality is 7.

In order to obtain 2X speedup, the MV accelerator needs to be equipped with 7 MAC units and associated buffers/registers/memory. In this case, running 7 MAC operations in parallel gives only 2X speedup due to the overhead of communication (C cycles) and buffer set-up (T cycles) while spending only T cycles in parallel computation in total C+2*T cycles for each tile.

Q2. (Verilog Implementation, 10 pts)

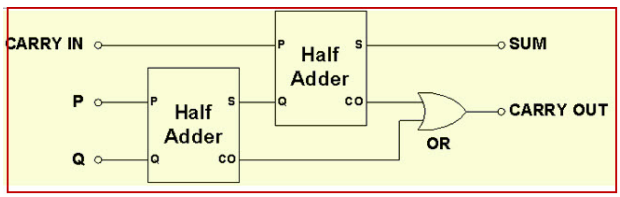
Answer)

(1)

```
module half_adder_1b (
    input ain, bin,
    output sum, carry_out
    );

    assign {carry_out, sum} = ain + bin;

endmodule
```

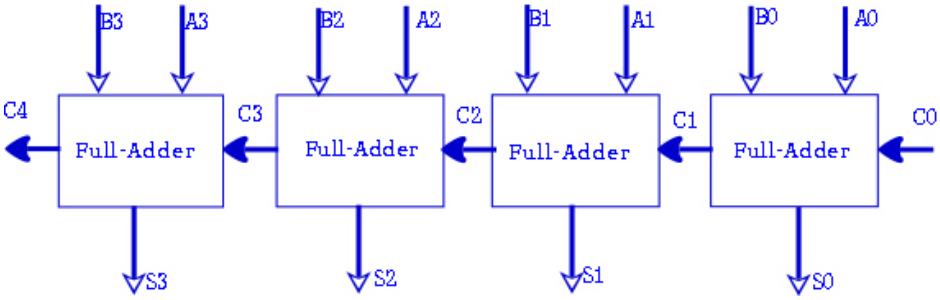
(2)

```
module full_adder_1b (
    input ain, bin, cin,
    output sum, carry_out
    );

    wire sum_from_ha1, carry_from_ha1 carry_from_ha2;

    half_adder_1b HA0(ain, bin, sum_from_ha1, carry_from_ha1);
    half_adder_1b HA1(sum_from_ha1, cin, sum, carry_from_ha2);

    assign carry_out = carry_from_ha1 + carry_from_ha2;

endmodule
```



(3)

```
module full_adder_4b (
    input [3:0] ain, bin,
    input cin,
    output [3:0] sum,
    output carry_out
    );

    wire [3:0] carry_from;

    full_adder_1b FA0(ain[0], bin[0], cin, sum[0], carry_from[0]);
    full_adder_1b FA1(ain[1], bin[1], carry_from[0], sum[1], carry_from[1]);
    full_adder_1b FA2(ain[2], bin[2], carry_from[1], sum[2], carry_from[2]);
    full_adder_1b FA3(ain[3], bin[3], carry_from[2], sum[3], carry_out);

endmodule
```



Q3. (Synthesizable code, 10 pts)

Answer)

(1) Latch Inference: ⓐ, ⓒ

(2)

ⓐ

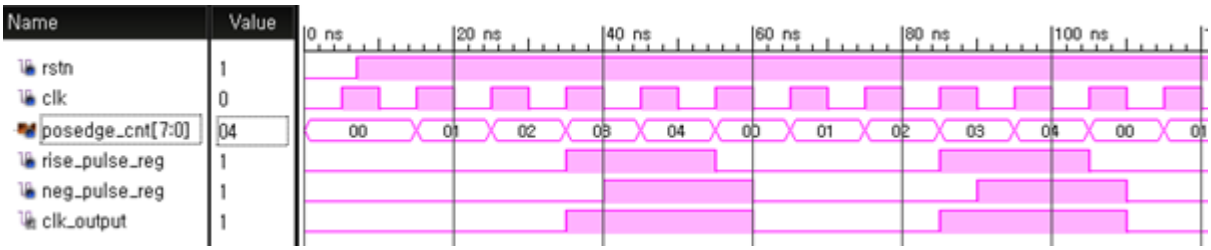
```
always @(enable or data)
    if(enable) y = data;
    else y = 0/1/x;

// OR
always @(enable or data) begin
    y = 0/1/x;
    if(enable) y = data;
end
```

ⓒ

```
always @(select or data)
   case(select)
       2'b00: y = data[select];
       2'b01: y = data[select];
       2'b10: y = data[select];
       default: y = 0/1/x;
endcase
```

Q4. (Simulation, 10 pts) Draw waveform of all the signals in the following Verilog code of signal generator.

Answer)



Q5. (Simulation, 10 pts) Draw waveform of all the signals of the following Verilog design of alarm system.

Answer)