Departamento de Engenharia Electrotécnica e de Computadores

Digital Systems Design 2016/2017

4º year, 1º semester
Exam 17 January 2017

Maximum duration: 3h00m, closed book.

[1.5 points]
1 - Explain how to assess the quality of a behavioral simulation process of a RTL model and describe how did you exploit that during the functional verification stage in the development of the last laboratory project (and if you did not make use of that, explain how could you have done that).

[2.5 points]
2 - During the design of a digital system, various verification processes must be performed at different design stages. Two important verification stages are the functional verification, by logic simulation, of the system under design before (behavioral simulation) and after (post-synthesis simulation) performing the RTL synthesis.

[1 point]
a) Explain the differences between the models that are simulated in these two stages.

[1.5 points]
b) If the post-synthesis simulation permits to verify the correctness of the circuit under design, explain why is it fundamental to do first the behavioral simulation, in the perspective of the verification and debug procedure (do not consider the fact that the post-synthesis simulation is much slower than the behavioral simulation).

[2.5 points]
3 - The construction of a reset signal is essential to define a coherent initial state in all registers of a synchronous digital system. If the reset signal is used directly from an external input (see example below) and if it is activated asynchronously with the clock signal (e.g. by pressing manually a push-button), this may lead to a wrong initialization of the registers controlled by this signal.

```
module toplevel( input reset, input clock, ... );
...
always @(posedge clock or posedge reset)
if (reset)
// initialization actions
else
// non-reset behavior
```

[1 point]
a) Explain why this wrong behavior may occur.

[1.5 points]
b) Explain how to solve this problem, assuming a clocked synchronous system with a single clock domain and an asynchronous reset signal, as shown in the example above.

[4 points]

4 - In the model represented by the Verilog code below, the memories M2 e M3 contain the same set of data:

```
// signals 'addr[5:0]' and 'enable' are inputs
reg [ 7:0] M1[0:63];
reg [19:0] M2[0:255], M3[0:255];
reg [19:0] p1, p2;
reg [ 7:0] i, j, a2, a3;
reg [19:0] yout;

always @(posedge clock)
if (reset)
  // initialize all register with zeros
else
begin
  yout <= enable ? ( p1 + p2 ) >> 1 : yout;
  a2   <= j + i;
  a3   <= j + i + 1;
  p1   <= M2[ a2 ];
  p2   <= M3[ a3 ];
  i    <= enable ? (i + 2) : 8'd0;
  j    <= M1[ addr ];
end
```

[2 points]

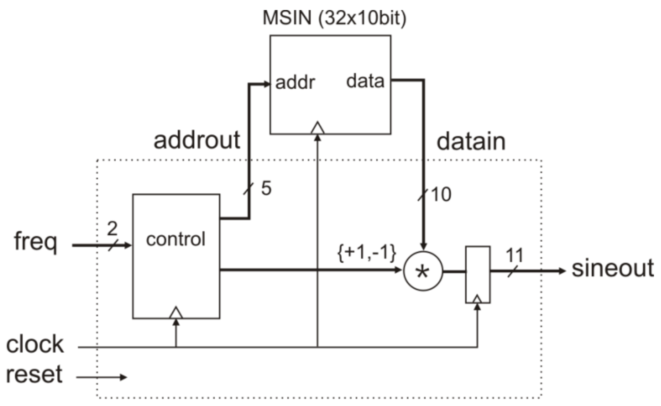
a) Draw a block diagram representing the circuit described by the Verilog code above. Assume that M1[ ], M2[ ] and M3[ ] are implemented by read-only memories with asynchronous (combinational) reads. To simplify your diagram do not represent the structures that implement the reset action of the registers.

[2 points]

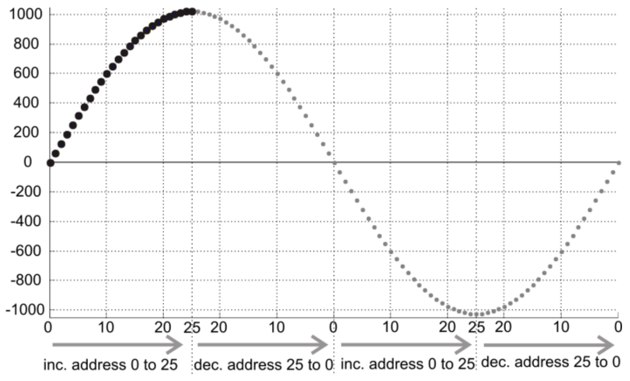
b) During a meeting of the design team, Mr. X said that it would be possible to redraw this function using less area and performing faster because the clock period to apply to the circuit is approximately 10X longer than the propagation delay of a 32 bit adder and 8X longer than the read propagation time of the memories M1, M2 and M3. Explain how to exploit this optimization and which implications this may have in the rest of the system that will use this block.

[4 points]

5 - The diagram below represents a digital sine wave generator. The output is a sequence of 11 bit signed values (two's complement) representing samples of a sine wave with frequencies selectable among 100 kHz, 50 kHz, 25 kHz and 12.5 kHz and sampling frequency equal to the main clock frequency (10 MHz).



A read-only memory external to the circuit under design (**MSIN**) contains a table with 26 samples representing the values of *sin(x)* in the first quadrant multiplied by 1023 (**x** values in the interval $[0,\pi/2]$, dark points in the figure below). By exploiting the symmetry of the function *sin(x)* and using an appropriate addressing to the memory **MSIN**, the samples of the function *sin(x)* can be generated for its whole domain. To generate a 100 kHz sine wave with a sample per clock period, the sequence of addresses to apply to **MSIN** should be 0, 1, 2, ..., 24, 25, 24,..., 2, 1, 0, 1, 2, ..., changing the sign of the value read from the memory during the 3rd and 4th sweep of the address space (3rd and 4th quadrants).



Using a 10 MHz clock signal and incrementing/decrementing the memory addresses by 1, a 100 KHz sine wave will be generated; if the address is incremented/decremented with a step of 0.5 (1 unit at each 2 clock cycles) the output frequency will be 50 kHz (similarly, a frequency of 25 kHz will be obtained with a step of 0.25 and 12.5 KHz with a step of 0.125).

Design a Verilog module that implements this system, following the coding guidelines and constraints for building synthezisable systems and adopting as the optimization criteria the minimization of the circuit area. The memory **MSIN** will be external to this circuit and contain the 26 samples of the function *sin(x)*.The read access of this memory is synchronous with the clock (the data read from the memory is presented in the data bus in the active clock transition). The address to the memory is the output **addrout** and the data read from the memory is the input **datain**. The reset signal should be synchronous with the clock and active with the logic high. The input **freq** defines the frequency of the output sine wave, according to the following table:

input **freq**	2’b00	2’b01	2’b10	2’b11
sine wave frequency	100 kHz	50 kHz	25 kHz	12.5 kHz

```
module sinegen( input clock,
                input reset,
                input  [ 1:0] freq,
                output [ 4:0] addrout,
                input  [ 9:0] datain,
                output [10:0] sineout
              )


endmodule;
```

[4.5 points]

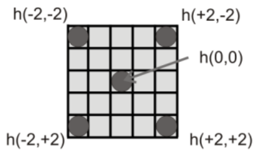
6 - In the design of a video processing system it is required to build a block to calculate in real-time a 2D convolution (5x5 pixel window) on a digital video signal. A video frame is received line by line (starting from the top of the frame) and within each line the pixels are received from left to right at the rate of the pixel clock (112 MHz). Two additional synchronization signals indicate the end of line and end of frame (these signals are not relevant for the system under design). A minimum of 256 clock cycles separate the end of one frame and the reception of the first pixel of the next frame.

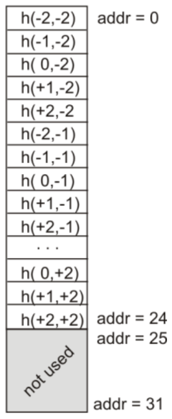
The 2D convolution between a digital image represented by the matrix *Pixelin(x,y)* and the impulse response (or *kernel*) with 5x5 coefficients, *h(i,j)*, can be expressed by the following equation, where *Pixelout(x,y)* represents the output pixel computed at coordinates *(x,y)*:

$$Pixelout(x,y) = \sum_{i=-2}^{+2} \sum_{j=-2}^{+2} Pixelin(x+i, y+j) \times h(i,j)$$

The system to implement uses a dedicated memory structure based on shift-registers that allows the parallel access to the 25 neighbor pixels *Pixelin(x+i, y+j)* of the pixel being computed, *Pixelout(x,y)*. At each pixel clock cycle a new pixel arrives and a new set of 25 neighbor pixels are presented to calculate a new output pixel. Both the pixel in and pixel out are 8 bit unsigned values

The 2D impulse response or *kernel* *h(i,j)* is formed by 25 fixed point signed values, with 3 bits representing the integer part and 7 bits the fractional part. These values are stored in memory **Mkernel (**32 x 10 bits, organized as shown in the figure below). This memory has an asynchronous read and can be written by an external control circuit during the 256 pixel clock interval between each two frames.





You can assume that the values *h(i,j)* guarantee that the result of each pixel will fit the unsigned 8-bit range (positive and less than 256).

It is known that, in the technology to be used for this project, a 8 x 10 bit multiplier (18 bit result) will have a propagation delay equal to 4 ns and a 20 bit adder will have a delay equal to 1.5 ns (this are approximate values).

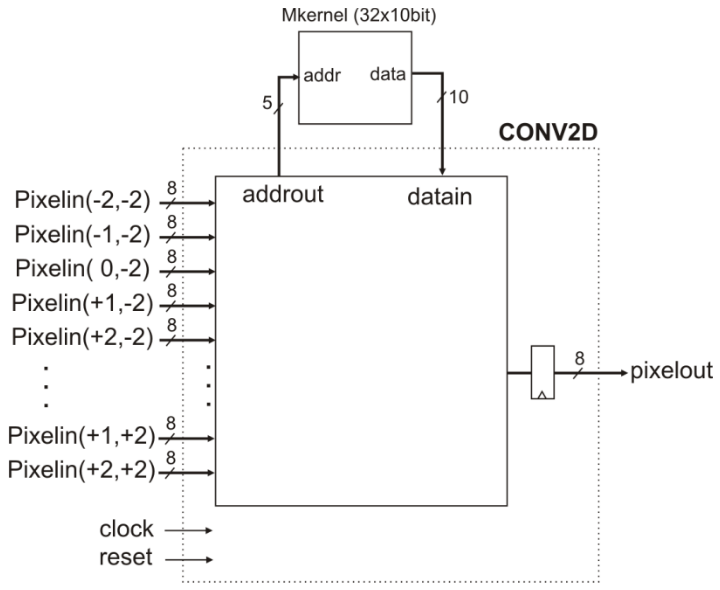


Figure on the right shows a simplified block diagram of the system, where block **CONV2D** is the module to design. The inputs **Pixelin(,)** are updated at each pixel clock and contain the 25 neighbor pixels in the 5x5 window around the pixel to compute. These are the values that must be multiplied by the coefficients *h(i,j)* and summed to produce the output pixel.

[1.5 points]
a) Show how to build an implementation for module **CONV2D** considering a clock signal with frequency equal to the pixel clock (112 MHz) and minimizing the latency of the circuit. Justify the proposed solution using block diagrams or sections of Verilog code to help understand your solution.

[1.5 points]
b) With the aim of decreasing the circuit area, it has been proposed to increase the clock frequency to 2X or 3X the pixel clock (224 MHz or 336 MHz, with periods equal to 4.46 ns and 2.98 ns, respectively). Show how to make use of a faster clock to reduce the circuit area and present an approximate number for the percentage of area reduction that may be achieved with your solution (ignore the issues related to the need of synchronization between the two clock domains).

[1.5 points]
c) This same system can be used to calculate 2D convolutions using smaller 3x3 kernels, instead of the original 5x5 (it is enough to set to zero, or consider as zeros, the 16 coefficients in the outside border of the 5x5 matrix – coefficients *h(-2,...), h(+2,...), h(...,-2) e h(...,+2)*. Consider that the control system that loads the **Mkernel** memory provides to your circuit one bit to select between a 5x5 kernel or a 3x3 kernel. Explain how to use that information to modify your system in order to reduce the dynamic power consumption when a 3x3 kernel is selected. Present an estimate of the decrease in the dynamic power consumption of the block **CONV2D**.

..:: the end ::..