Lab 2: Matrix Multiplication Simulation

National Chiao Tung University
Chun-Jen Tsai
09/19/2017

Lab 2: Matrix Multiplication Simulation

- In this lab, you will design a circuit to do a 3×3 matrix multiplication on Vivado Simulator.
  - Two register arrays with of 3×3 matrices will be given to you in the sample Verilog simulation project.
  - You must design a Verilog program to compute their multiplication, and print the result from the testbench.
  - You must use no more than 9 multipliers to implement your circuit
- The deadline of the lab is on 9/26, by 5:00pm.

The Input Matrix Format

- Each input matrix has 9 unsigned 8-bit elements of values between 0 ~ 127. The declaration of the A and B matrices in Verilog should be as follows:

```
reg [0:9*8-1]  A = 72'h_4F_7E_57_0F_14_7B_21_4C_54;
reg [0:9*8-1]  B = 72'h_17_28_3A_40_2F_33_6C_22_77;
```

  - Each matrix is stored in a 72-bit register, each number in the matrix is of 8-bit
  - The matrix is stored in row-major format

- The output matrix has 9 unsigned 17-bit elements

The Specification of the Multiplier

- The matrix multiplier module is defined as follows:

```
module mmult(
  input  clk,                  // Clock signal.
  input  reset_n,              // Reset signal (negative logic).
  input  enable,               // Activation signal for matrix
                               //   multiplication (tells the circuit
                               //   that A and B are ready for use).
  input  [0:9*8-1] A_mat,      // A matrix.
  input  [0:9*8-1] B_mat,      // B matrix.

  output valid,                // Signals that the output is valid
                               //   to read.
  output reg [0:9*17-1] C_mat // The result of A x B.
);
```

  - In the project workspace that is given to you, this module is empty. You must design this module by yourself.

Computation of $A_{3\times3} \times B_{3\times3}$

- A 3×3 matrix multiplication is composed of 9 inner products:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{pmatrix}$$

- You can compute the outputs in each column of the C matrix in parallel in one clock cycle
  - At each clock cycle, you use nine multipliers
  - Three columns of the C matrix takes three cycles to compute!

The Testbench of the mmult() Module

- We will provides a testbench for you to test the mmult() module you create

- The testbench is composed of three parts:
  - Simulation of the clock and reset signals
  - Instantiation of the mmult() module and generates its input signals
  - Print the output matrix to the console window

Simulation of Cock and Reset Signals

- Digital systems usually requires clock and reset signals

```
reg clk = 1;       // Clock signal
reg reset_n = 1;   // Reset signal

// 100MHz clock generator
always
  #5 clk = !clk;

// Reset signal simulator
event reset_trigger;

initial begin
  forever begin
    @ (reset_trigger);
    @ (negedge clk);
    reset_n = 0;
    @ (negedge clk);
    reset_n = 1;
  end
end
```

```
// To issue a reset, you must
// trigger a reset event by the
// following code:

#10 -> reset_trigger;
```

Instantiation & Invocation of mmult()

```
reg  [0:9*8-1]  A, B;   // 3x3 matrices
wire [0:9*17-1] C;
reg  enable;
wire valid;

// Instiantiates a 3x3 matrix multiplier
mmult uut(
  .clk(clk), .reset_n(reset_n), .enable(enable),
  .A_mat(A), .B_mat(B), .valid(valid), .C_mat(C)
);

initial begin
  // Add stimulus here
  A = 72'h_4F_7E_57_0F_14_7B_21_4C_54;
  B = 72'h_17_28_3A_40_2F_33_6C_22_77;

  // Issue a reset signal
  #10 -> reset_trigger;

  // Wait 100 ns for global reset to finish
  #100 enable = 1;
end
```

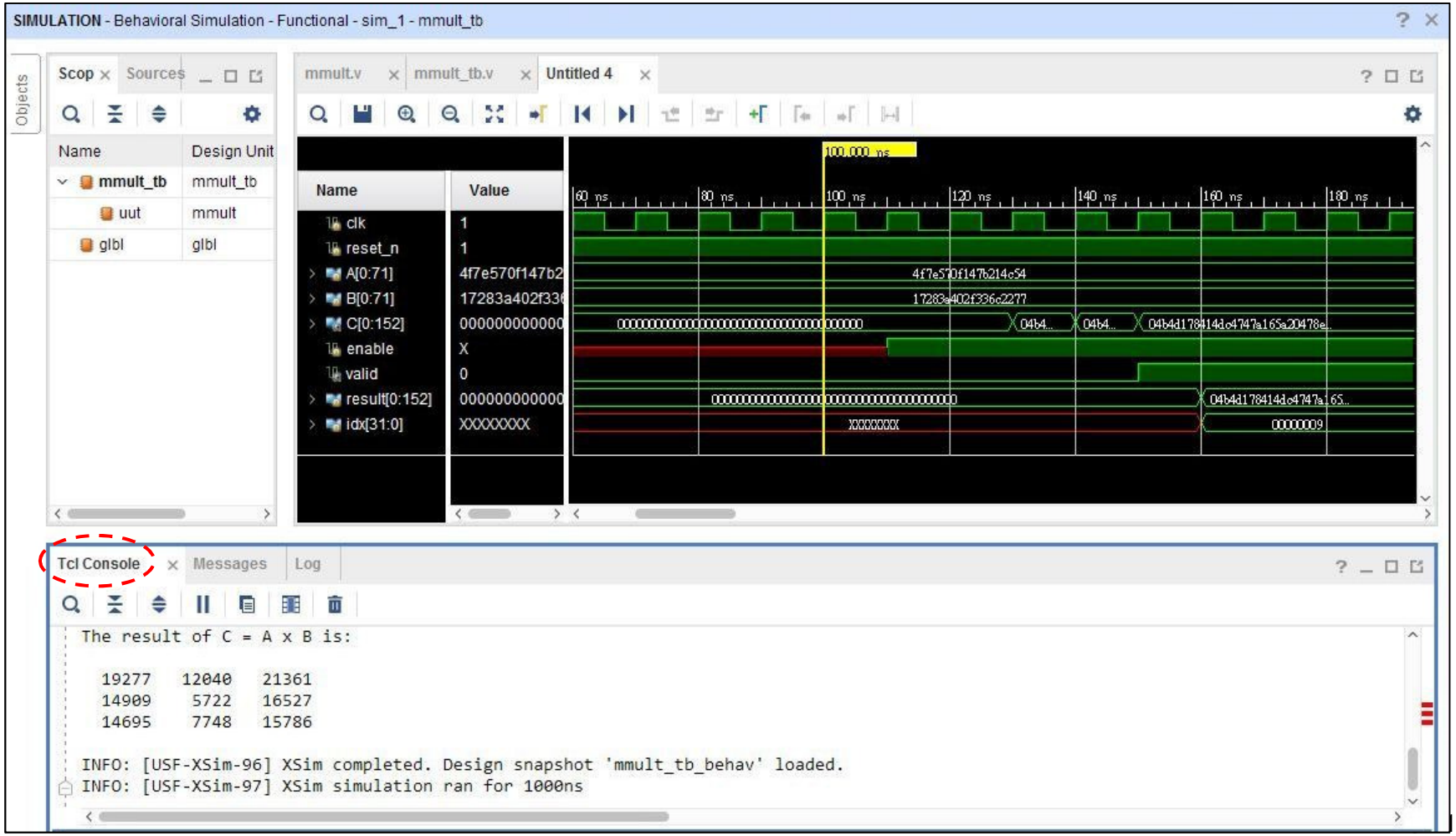
Print the Output Matrix

- In simulator, you can use print the output to console:

```
always @(*) begin
  @(posedge valid);

  // Wait one clock cycle so that the output is saved in result[].
  #10 $display("\nThe result of C = A x B is:\n");

  for (idx = 0; idx < 9; idx = idx+1)
  begin
    $write(" %d ", result[idx*17 +: 17]);
    if (idx%3 == 2) $write("\n");
  end

  $write("\n");
end

always @(posedge clk) begin
  if (~reset_n) result <= 0;
  else if (valid) result <= C;
  else result <= result;
end
```

The Simulation Output

- The $display() function sends output to "Tcl Console"



Lab 2 Sample Project Workspace

- You can download a sample project workspace of Lab2 from E3. In the workspace, there is a sample testbench file mmult_tb.v and an empty module file mmult.v.
- You must rewrite mmult.v so that the simulation result is correct.
  - You should upload your lab2 solution to E3 before the deadline
- During the demo time on 9/26, there is an on-line test that asks you to do some matrix multiplications
  - You can download and use your code from E3 during the test as a reference