

# Lab1: Design a Sequential Binary Multiplier



National Chiao Tung University  
Chun-Jen Tsai  
9/20/2016

# Lab 1: Design an 8-bit Multiplier

---

- ❑ In this lab, you will design an 8-bit sequential binary multiplier and use ISim to verify your design
  - You should review your old textbook on Digital Circuit Design by Mano for general digital circuit design techniques. Some design guideline of the sequential binary multiplier is in section 8.7 of Mano's book.
  - You must design your multiplier using only adder, shifter, multiplexor, and gate-level operators. You can not use the multiplication operator of Verilog.
- ❑ You must demo to your TA during the Lab hours on 9/27 that your circuit works

# Module Specification

- ❑ The input/output ports of the multiplier is as follows:

```
module SeqMultiplier(  
    input wire clk,  
    input wire enable,  
    input wire [7:0] A,  
    input wire [7:0] B,  
    output wire [15:0] C  
);
```

‘clk’ is the system clock,  
‘enable’ activates the multiplication operation,  
‘A’ is the 8-bit unsigned multiplicand input,  
‘B’ is the 8-bit unsigned multiplier input, and  
‘C’ is the 16-bit unsigned product output.

# Sequential Binary Multiplier Behavior

- ❑ The behavior of sequential binary multiplication of 10111 and 10011 is as follows:

$$\begin{array}{rcl} & 00010111 & \rightarrow \text{multiplicand} \\ \times & 00010011 & \rightarrow \text{multiplier} \\ \hline & 00010111 & \\ & 00010111 & \\ & 00000000 & \\ & 00000000 & \\ + & 00010111 & \\ \hline & 000110110101 & \rightarrow \text{product} \end{array}$$

At every clock cycle, the circuit reads one bit from the multiplier and adds the multiplicand to the product before shifting it.

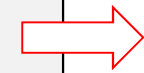
# C Model of the Multiplier

- ❑ The C code that performs the sequential multiplication:

```
typedef unsigned char Byte;
typedef unsigned short Word;

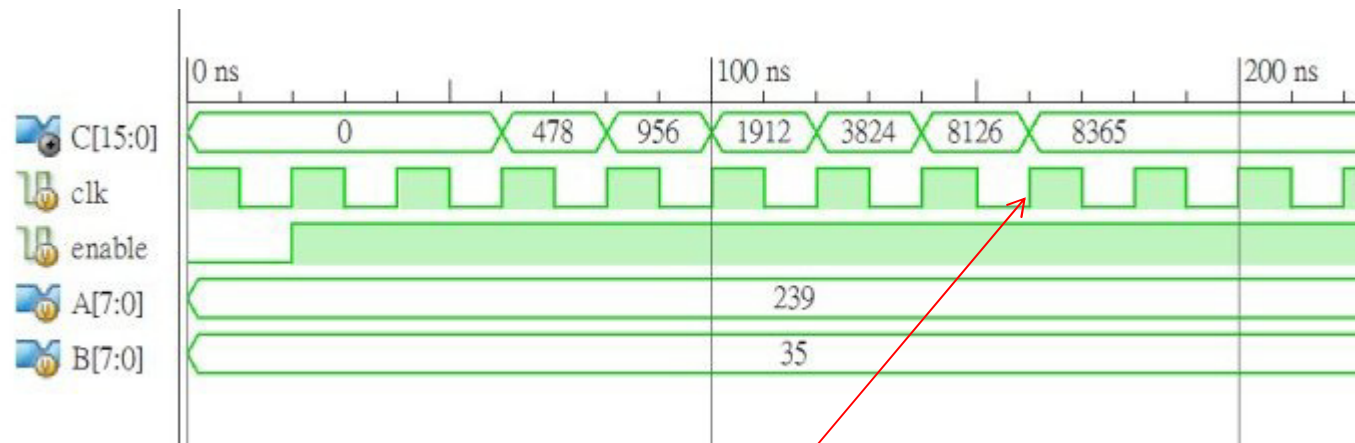
SeqMultiplier(Byte A, Byte B, Word *C)
{
    int idx;

    *C = 0;
    for (idx = 8; idx > 0; idx--)
    {
        *C = *C << 1;
        if ((B & 0x80) == 0x80)
        {
            *C += A;
        }
        B = B << 1;
    }
}
```

 Your task for lab1 is  
to rewrite this C code  
using Verilog!

# Example of Simulated Timing Diagram

- An example of the timing diagram of  $239 \times 35 = 8365$



Stable output 8365 happens  
7 cycles after enable == 1