Verilog HW#3: Vector Inner Product

Chun-Jen Tsai
National Chiao Tung University
5/15/2020

Verilog HW#3

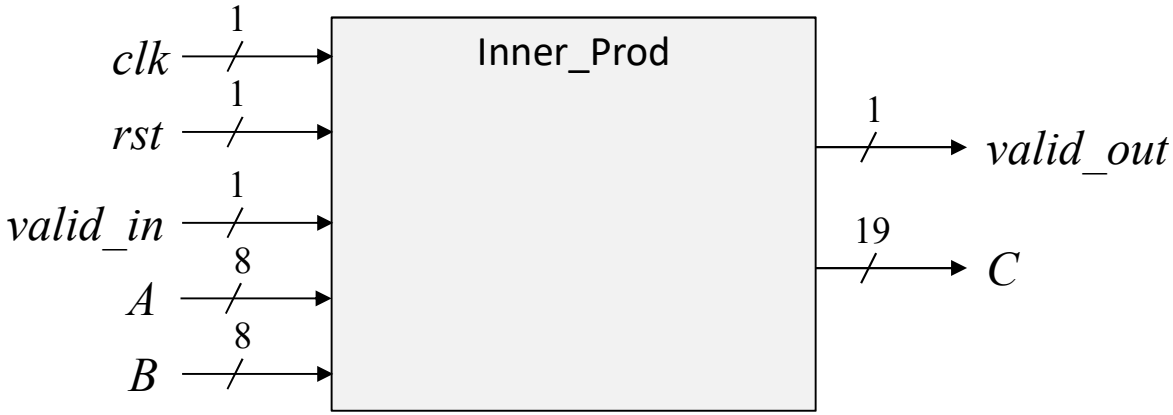
- Goal: Design a synchronous circuit that reads two 1×8 vectors from its two input ports, computes their inner product, and send the result to an output port.

  Each input 1×8 vector contains eight 8-bit unsigned binary numbers. The inner-product output contains a 19-bit unsigned binary number.

- Deadline: 5/25, 23:55pm. You must upload your Verilog module to the E3 website by the deadline.

Block Diagram of the Module

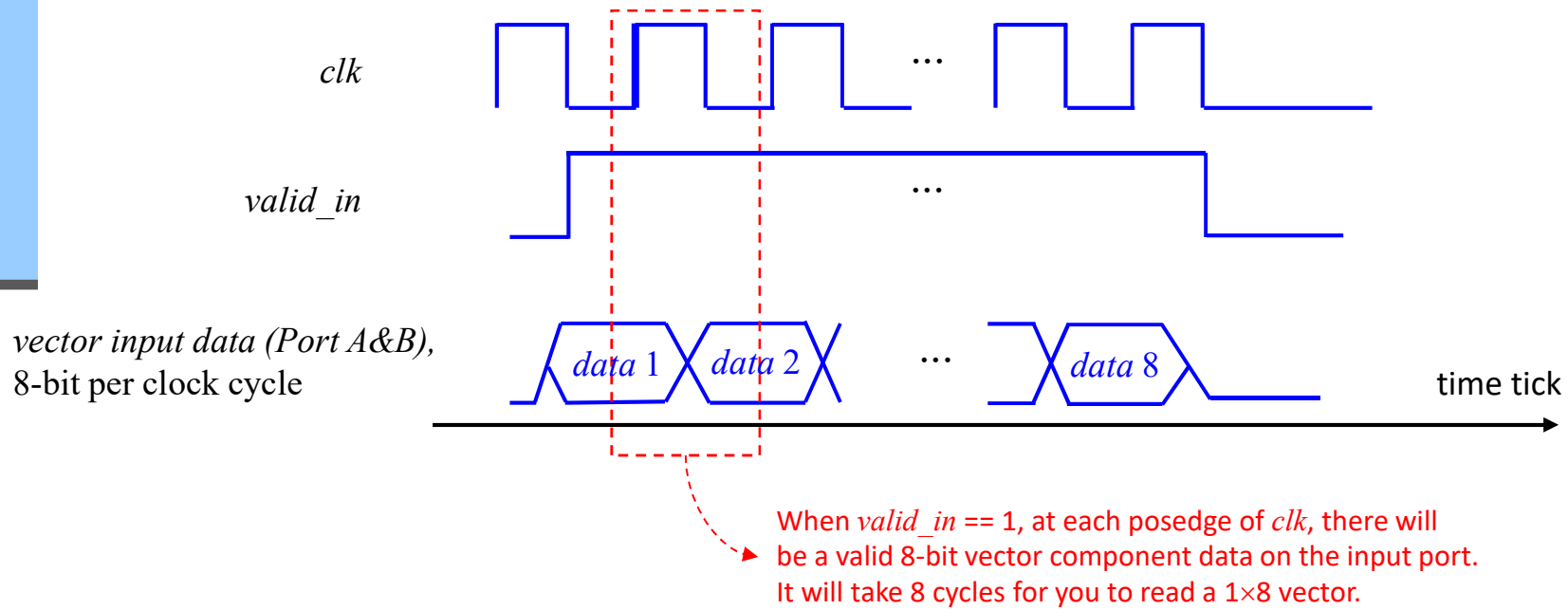
- Your module should be called Inner_Prod, as follows:



  - *clk* is the 100MHz reference clock.
  - *rst* is the reset signal for the circuit.
  - *A* and *B* are the input ports for two 1×8 vectors.
  - *C* is the output port for the inner-product value.
  - *valid_in* and *valid_out* will be explained in the next two slides.

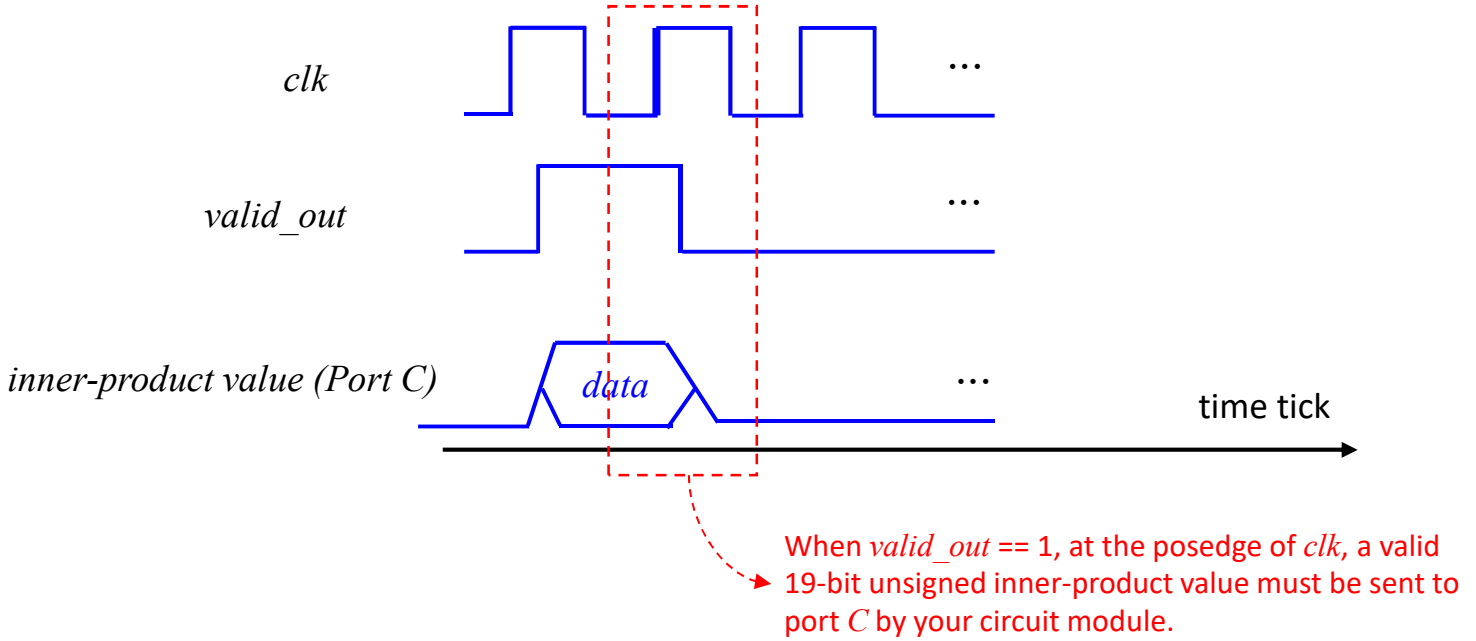
Input of the Vectors

- Since a 1×8 vector has eight 8-bit binary numbers, we will send the components of a vector into the circuit module sequentially, one component per clock cycle.
  - *valid_in* signals the begin and the end of the vector data, as the following timing diagram shows:



Output of the Inner-Product Value

- After both vector A and B are read into the circuit module, it should take only one clock cycle to compute the inner-product and send the result to port C.
  - Your circuit must raise the *valid_out* signal by one clock cycle to inform the testbench the transmission of the output value:



Inner Product Computation

- Assume $A = (a_1, a_2, \ldots, a_8)$, $B = (b_1, b_2, \ldots, b_8)$, then we have $C = a_1b_1 + a_2b_2 + \ldots + a_8b_8$.

- Since data pairs a_i and b_i arrives at the input ports sequentially, you should try to compute a_ib_i as they arrives, and accumulate the partial result into C so that you use only one multiplier in your module.

Initialization of Circuit Signals

- The circuit module must be able to compute inner product repeatedly, therefore you must initialize your internal signals for each computation. The time instants when you should reset your signals are as follows:
  - A the beginning of the simulation, when the *rst* signal is set (positive active), you should initialize your internal signals.
  - After the inner product, when the *valid_out* is set, you should trigger the initialization of your internal signals as well.

Requirements for Verilog HW#3

- The module you designed must be declared as follows:

```
module Inner_Prod(input clk,
                  input rst,
                  input valid_in,
                  input unsigned [7:0] A,
                  input unsigned [7:0] B,
                  output valid_out,
                  output unsigned [18:0] C);

  /* Implement your design here. */
endmodule
```

- Your circuit must be able to perform inner product repeatedly every time *valid_in* is activated.
  - You can assume that once *valid_in* is activated, it is guaranteed to last for 8 clock cycles.
  - Note: the sample testbench TAs provided on E3 only tests your circuit once, you must try repeated testing by yourself!