Verilog HW#3: GCD Computation

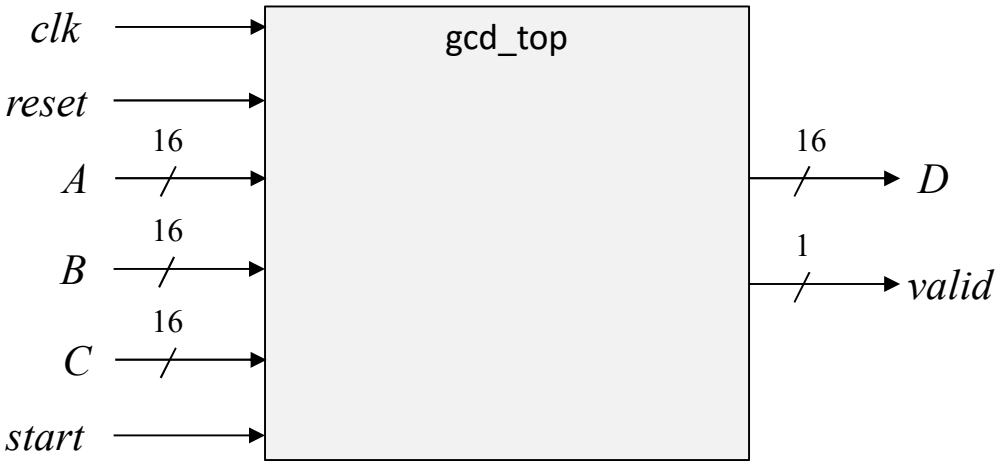
Chun-Jen Tsai
National Chiao Tung University
5/5/2021

Verilog HW#3

- Goal: Design a sequential circuit that computes the greatest common divider (GCD) of three 16-bit unsigned numbers.
  - In your design, you must implement the Euclidean algorithm to compute the GCD
  - You cannot use any division, modulus, or multiplication operations in your design
- Deadline: 5/17, 23:55pm. You must upload your Verilog files to the E3 website by the deadline.

I/O ports of the ‘gcd_top’ Module

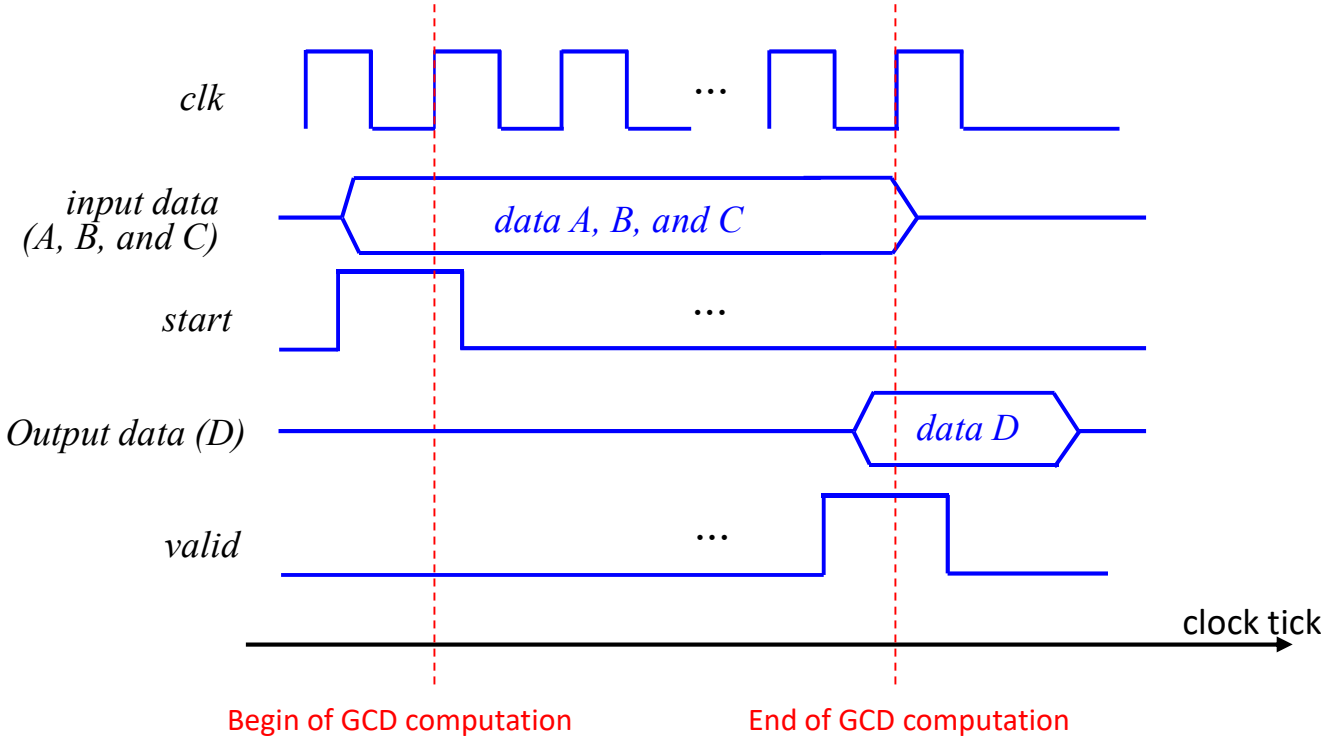
- Your module ‘gcd_top’ should have the following ports:



  - A, B, and C are the input 16-bit nonzero numbers.
  - D is the 16-bit GCD of A, B, and C.
  - *start* is the 1-bit signal that trigger the GCD computation
  - *valid* is a 1-bit signal that indicates the data D is ready for fetch.

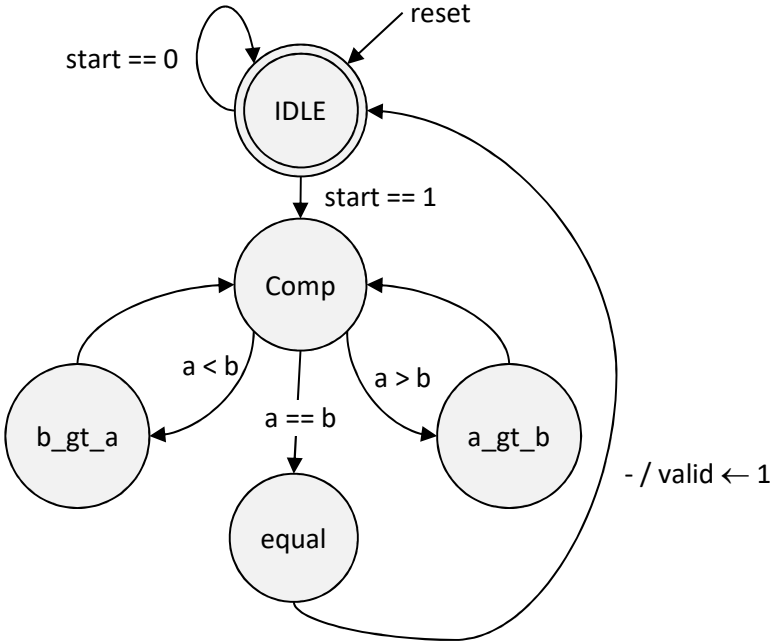
Timing of GCD Computation

- The begin and finish of GCD computation should be signaled by the *start* and *valid* signals
  - Note that both *start* and *valid* only active for one clock-cycle



About Your GCD Module

- Your design should have a `gcd` module that compute the GCD of *a* and *b* that contains an FSM
  - For the Euclidean algorithm, a two-state FSM is good enough. But you should implement the following FSM, just for practice:



Requirements for Verilog HW#3 (1/2)

- The module you designed must be declared as follows:

```
module gcd_top(input clk, input reset, input [15:0] A, input
[15:0] B, input [15:0] C, input start, output [15:0] D, output
valid);

  /* Implement your design here. */

endmodule
```

- Do not upload your testbench module to E3, just upload the `gcd_top()` module and its supporting modules (if you have more than one modules/files). The TA's will use their own testbench to test your module.

Requirements for Verilog HW#3 (2/2)

- The reset signal should set the circuit state to IDLE and clear all internal registers to zero
- The circuit can be invoked repeatedly
  - If the *start* signal is activated again before the previous computation is finished (i.e., before the signaling of *valid* is finished), it shall be ignored