STEPS TO DESIGN COMPLEX DIGITAL SYSTEMS

Steps to Design Complex Digital Systems

- This is an example design flow; different specifications may require or favor a different approach

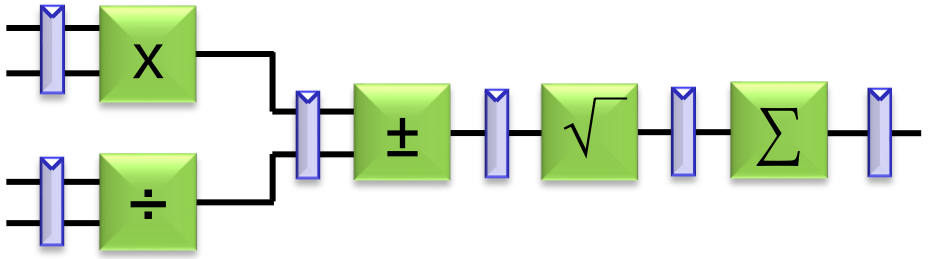
1) Nail down answers to:
   What outputs do I want
   and **when** (i.e., in which relative clock cycles)?
2) Make a list of the input signals
3) Figure out which datapath and memory elements are needed to calculate the desired outputs
4) Draw a block diagram of the datapath and memory elements

Steps to Design Complex Digital Systems

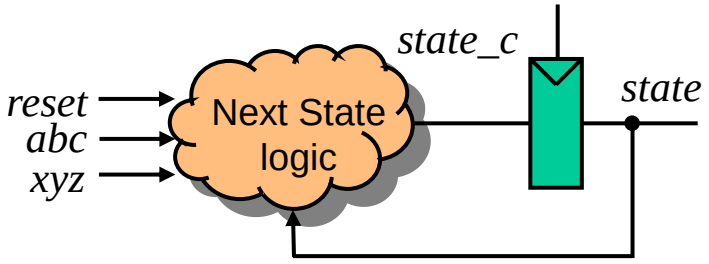
- This is an example design flow; different specifications may require or favor a different approach

5) Design a pipelined block diagram that meets the computational requirements

   a) The datapath may require many pipeline stages

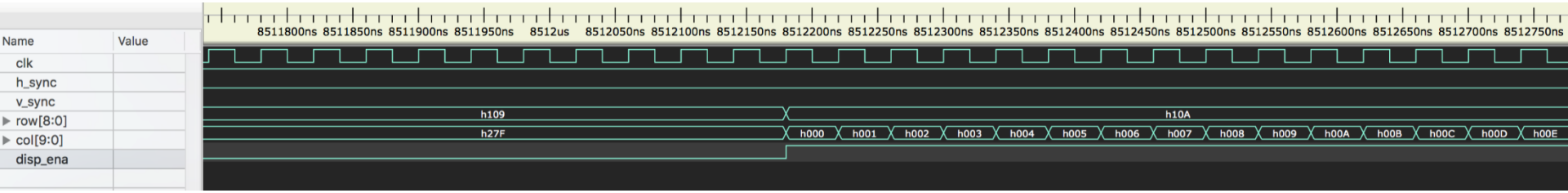


   b) The controller(s) and/or counter(s) are typically simple single-stage blocks



Steps to Design Complex Digital Systems

6) Draw a complete and detailed timing diagram that enables the hardware in the pipelined block diagram to meet the computational requirements. The diagram must include:
   - All system inputs
   - All key internal signals including any different versions of the same signal in different pipeline stages
   - All system outputs



7) Iterate steps #1 and #2 as many times as necessary until you are quite sure it will meet all specifications

Steps to Design Complex Digital Systems

8) Design the controller(s)
   - Plan states and counter(s)
   - Draw state graphs
   - Add all key signals to the Timing Diagram; make sure the essential signals are available during the clock cycle when needed
9) Stare at all diagrams and modify your design until you are quite sure it will work
10) *Begin* thinking about things like **`case`** statements, **`if/then/else`** statements etc. Type in verilog. Begin debugging.