## 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) 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

- 2) 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

Name	Value	8511800ns 8511850ns 8512190ns 8512950ns 8512050ns 8512100ns 8512150ns 85121500ns 85121500ns 85121500ns 8512000000000000000000000000000000000000	200ns 8512250ns 8512300ns 8512350ns 8512400ns 8512450ns 8512500ns 8512550ns 8512600ns 8512650ns 8512700ns 8512750ns
Vallie	value		
clk			
h_sync			
v svnc			
		h109 X	h10A
▶ row[8:0]		h275 X h00	
▶ col[9:0]			
disp ena			
uisp_ena			

3) 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

- 4) 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
- 5) Stare at all diagrams and modify your design until you are quite sure it will work
- 6) *Begin* thinking about things like **case** statements, **if/then/else** statements etc. Type in verilog. Begin debugging.