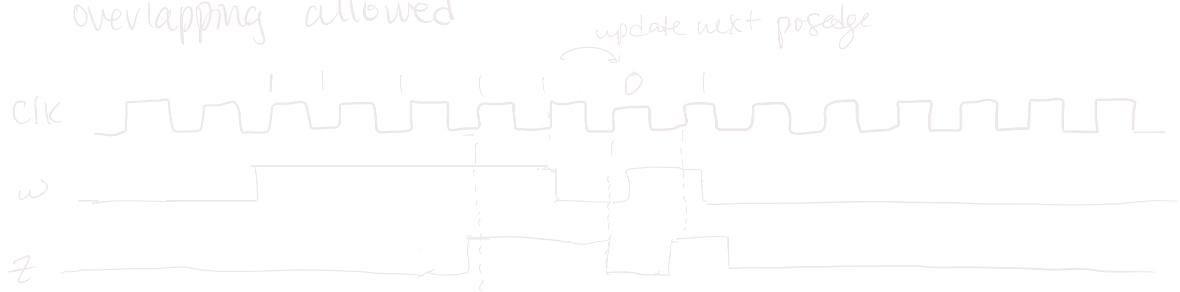
Part 1

1111 or 1101

input w → output z (1 if 1111 or 1101 (pulses)

overlapping allowed



2) resetn is active low. It is synchronous, because it is inside the "posedge" clock cycle code, so changes w/ the clock cycle. b/c if(!resetn)

Simulation to reset FSM to starting state: set resetn = 0

Part 2

FSM to control a datapath.

$Cx^2 + Bx + A$ ↳ 8 bit unsigned

R_x, R_a, R_b, R_c
x A B C

KEY1 is not clock, use Clock_50 50MHz clock

1) Currently $A^2 + C$

2) 1. load data into 4 registers C, X, A, B

2. compute Bx, load into B

3. compute $Bx + A$, load into B

4. compute x^2, load into A

5. compute Cx^2, load into A

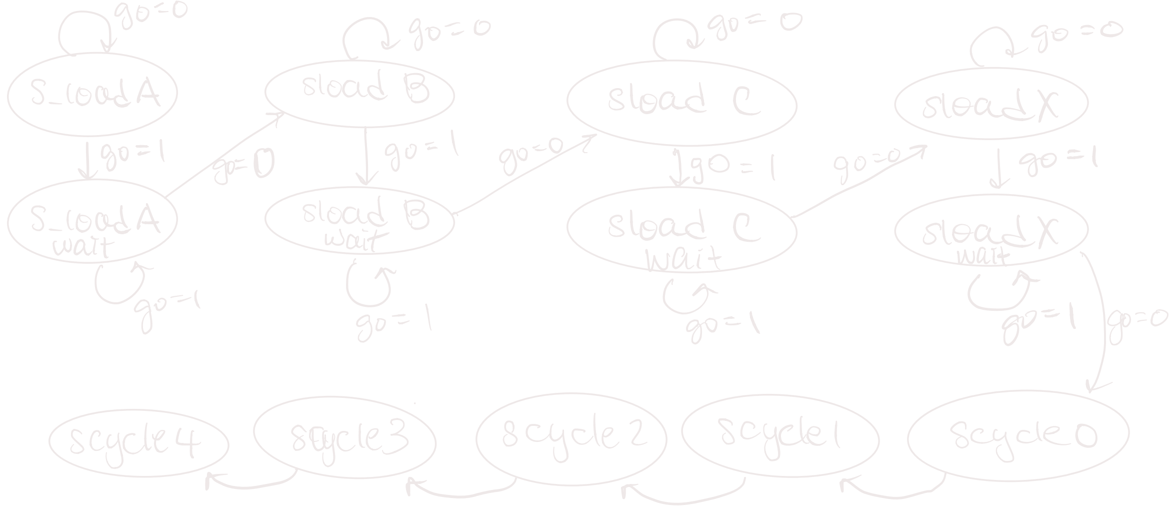
6. compute $cx^2 + Bx + A$, load into out register R

A=0, B=1, C=2, X=3

Steps	Register State				Control Signals
	RA	RB	RC	RX	
- load C into RC			C		- ld_C = 1
- load X into RX				X	- ld_x = 1
- load A into RA	A				- ld_a = 1, ld_alu_out = 0 ↓select signal
- load B into RB		B			- ld_b = 1, ld_alu_out = 0
- multiply RB & RX, store in RB		Bx			- alu-select_a = 01 alu-select_b = 11 alu_op = 1 ld_alu_out = 1 ld_b = 1
- add RB & RA, store in RB		Bx+A			- alu-select_a = 01 alu-select_b = 00 alu_op = 0 ld_alu_out = 1 ld_b = 1
- multiply Rx & Rx, store in RA	x^2				- alu-select_a = 11 alu-select_b = 11 alu_op = 1 ld_alu_out = 1 ld_a = 1
- multiply RA & RC store in RA	Cx^2				- alu-select_a = 00 alu-select_b = 10 alu_op = 1 ld_alu_out = 1 ld_a = 1
- add RA & RB store in R					- alu-select_a = 00 alu-select_b = 01 alu_op = 0 ld_r = 1

3) Controller FSM

- Need 2 more clock cycles, 5 states in total



KEY[0] = reset (act low)

KEY[1] = go