



L9: Lab 2 Tri-state, Glitches, Identifiers, and Q1 Review

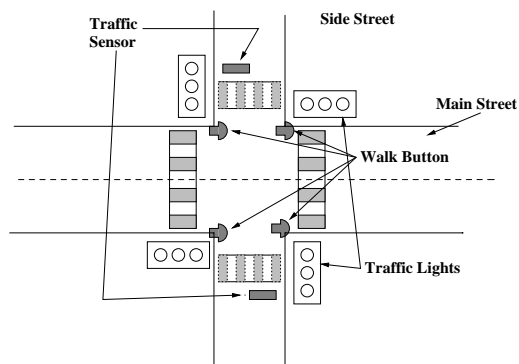


Lab 2 Assignment



Traffic Light Controller

It has main and side streets
with a walk light button.



The main street part of the
cycle is longer than that of
the side street.
(TBASE = TEXT)

But the side street has a
sensor which keeps it green
a bit longer. (TEXT)
The sensor MUST be
synchronized.

The walk button must be
latched and serviced at the
right time. It is to be
unlatched after it has been
serviced.

Walk is R-Y.
Blink is main Y and side R.
Blink interval (on or off) is
TBLINK.



Top Level Block Diagram



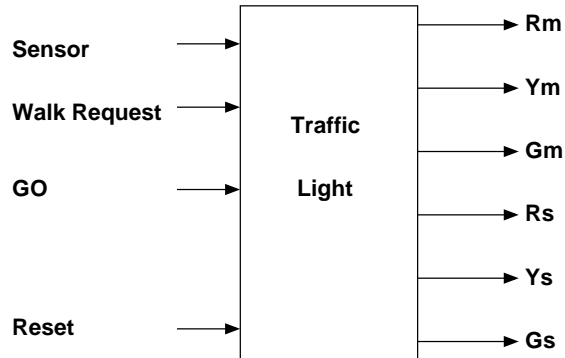
Design Procedure

Start with a simple block diagram.

Break the design down into more simple blocks.

Implement the simpler blocks and put it all together.

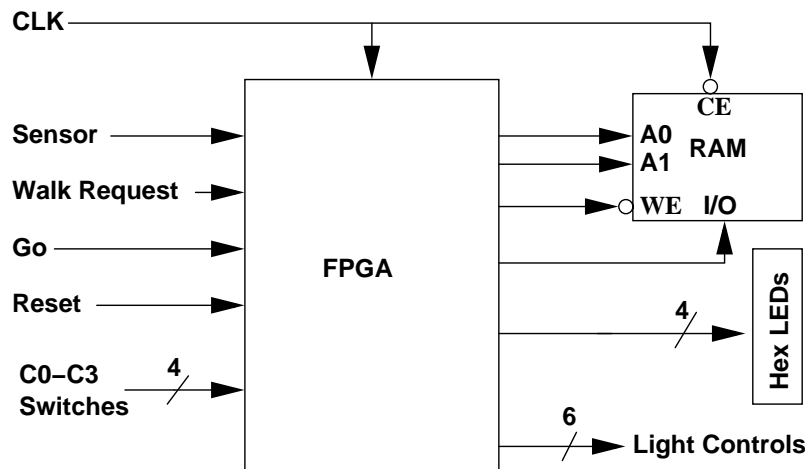
Note that this GO signal is similar to that discussed earlier, namely, a single pulse in response to a pushbutton (which could be of any Length).



Implementation Details

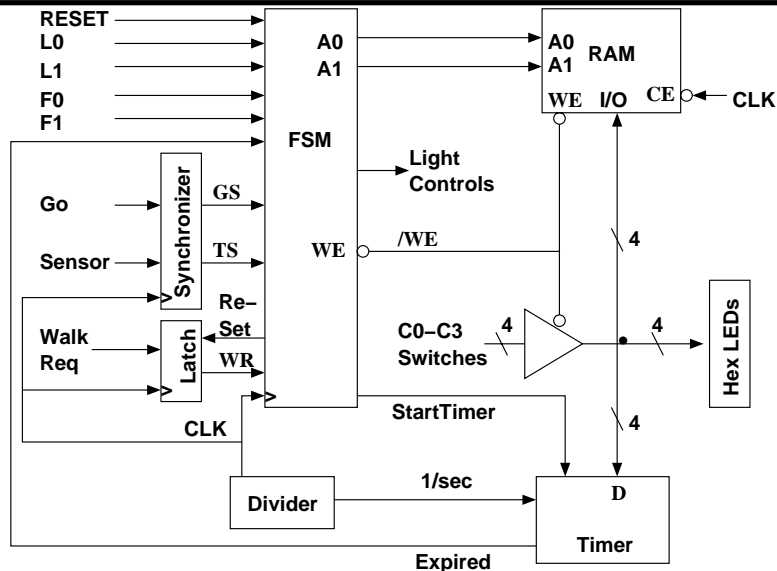


We want you to use a RAM chip. Do not include RAM in your FPGA. Use the HEX LEDs on the kit to display memory contents.





Detailed Functional Diagram



L9 6.111 Spring 2003

Introductory Digital Systems Laboratory 5



FSM Inputs and Outputs



Inputs

- RESET (from a switch)
- GS Synchronized GO signal
- F1, F0 Function Selection (from switches)
- L0, L1 RAM Location Selection (from switches)
- TS Synchronized sensor signal
- WR Walk Request (Re-settable latch set by pushbutton)
- Expired True when the timer has timed out

Outputs

- A0, A1 SRAM Address
- N_WE SRAM Write Enable, also Gates Switches onto I/O
- StartTimer Resets one second increment timer
- Light Controls
 - Rm, Ym, Gm, Rs, Ys, Gs

L9 6.111 Spring 2003

Introductory Digital Systems Laboratory 6



Functions and RAM Locations



■ Function Specifications

F1 F0	Are provided by function control switches.
0 0	Examine memory location provided by the switches.
0 1	Store value in memory location specified by switches.
1 0	Run traffic lights.
1 1	Blink.

■ Meaning of locations of the SRAM

L1 L0	Location specification is provided by switches.
0 0	TYEL Time for yellow light
0 1	TBASE Base Interval
1 0	TEXT Extension Interval
1 1	STBLINK Blink Interval



Lab 2 Schedule



■ The use of Lab 2 for Phase II is optional.

Feb 26	Lab Assignment (done today)
March 4	Design Conference (oral)
March 10	Lab 2 check-off Lab 2 report due
March 13	Revised Lab 2 report for Phase II (optional)



Tristate Loadable Counter



```

-- Use tri-state logic to multiplex IO pins.
library ieee;
use ieee.std_logic_1164.all;
-- needed for integer + signal
use ieee.std_logic_unsigned.all;
entity ldcntc is
-- can pass in parameter width if
-- instantiated as a component.
-- Otherwise 3 is the value of width.
generic (width : integer := 3);
port (clk, ld, oe, cnt_enb : in std_logic;
      data : inout
      std_logic_vector(width - 1 downto 0));
end ldcntc;
-- purpose: count with an output enable

```

```

architecture archldcnt of ldcntc is
signal counter :
std_logic_vector(width - 1 downto 0);
begin
data <= counter when oe = '1'
else (others => 'Z');
-- N.B. Z must be UPPER CASE!
cnt: process (clk)
begin
if rising_edge (clk) then
if ld = '1' and oe = '0' then
counter <= data;
elsif cnt_enb = '1' then
counter <= counter + 1;
end if;
end if; -- rising_edge (clk)
end process cnt;
end architecture archldcnt;

```



Tristate Simulation



```

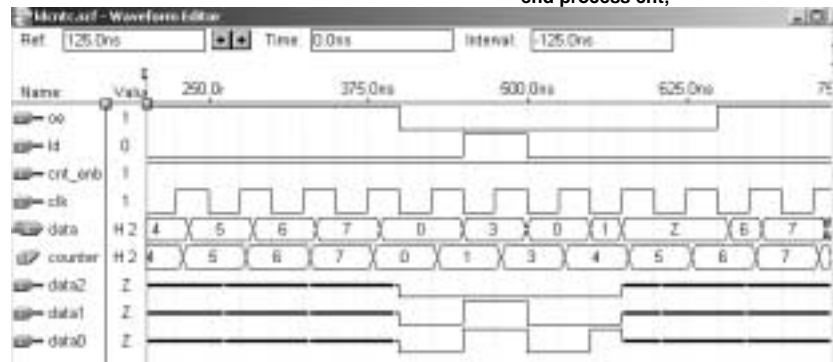
data <= counter when oe = '1'
else (others => 'Z');
-- N.B. Z must be UPPER CASE!
cnt: process (clk)

```

```

begin
if rising_edge (clk) then
if ld = '1' and oe = '0' then
counter <= data;
elsif cnt_enb = '1' then
counter <= counter + 1;
end if;
end if; -- rising_edge (clk)
end process cnt;

```

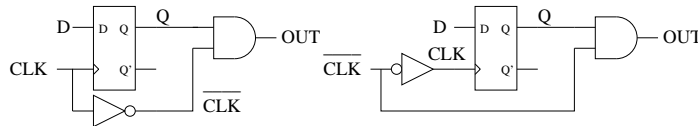




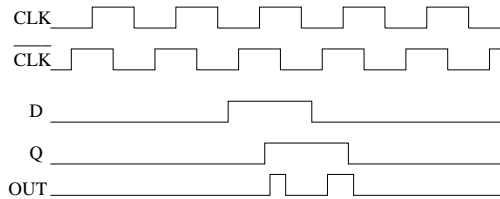
Avoiding Glitches - Gating



- **Gate the output with the clock (carefully).**
 - This is another way of saying “Don’t look at a combinational output until the output has settled down into its final state”.
 - Make sure that the gated pulse does NOT have a glitch.



Suppose the inverter delay is large compared to the CLK to Q delay.
Then there can be a glitch on the circuit to the left but not on the circuit to the right.



Remember –
Do NOT use glitchy signals for CLK, PR, CLR, S, or R.
Clock data into a register AFTER signals are stable.



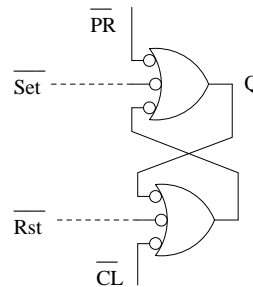
Avoiding Glitches - Register



- **Register the output.**
 - If a flip-flop does NOT change state upon the occurrence of a clock pulse then it has no glitches, i.e.,
 - 0 to 0 is guaranteed not to glitch to 1 in between and
 - 1 to 1 is guaranteed not to glitch to 0 in between.
 - We assume that setup and hold times are honored.
 - Some high performance flip-flops do not adhere to this. They are only concerned that the ff end up in the “right” state and not whether they have glitches on the output.

This is the output latch of an edge triggered flip-flop.

The set or reset pulses (negative true) are full pulses and only one occurs if the setup and hold times are adhered to.





VHDL Identifiers & Reserved Words



- **Case Insensitive (but best not to rely on this)**
 - First character must be a letter.
 - Letters, Digits, and Underscores (only)
 - Two underscores in succession are not allowed.
 - The last character cannot be an underscore.
 - Using reserved words as identifiers is NOT allowed.
 - Reserved words are AQUA in emacs.
 - Using reserved words usually provokes an understandable error comment.
 - Legal examples are CLK, Three_state_enable, h23, Reg_12.
 - Illegal examples are _clk, 3_state_enable, large#num, clk_.
 - Some reserved words are
 - Abs, access, impure, postponed
 - In other words, there are too many to remember!
 - To see what is a reserved word, notice the color in your editor.
 - This is another good reason for “incremental” compilation.
 - Start with code that compiles and add a block at a time.



Miscellaneous



- s'event is read as “s tick event” where s is a signal name.
- Rising_edge(s) is the same as (s'event and s = '1')
 - as synthesis only worries about 1 and 0.
- s'event - true if an event occurred in the current delta cycle
- Don't cares are represented by a - (hyphen).
 - '-' for a character “- - -” for a string (vector)
 - (others => '1') for something independent of length
- & (ampersand) to concatenate strings or signals
 - “01” & “111” is the same as “01111”.
 - '0' & “1111” is the same as “01111”.
- A + B is valid only if A and B are of the same length.
 - The result is of the same length as A (or B).
 - If you want the result to be one bit longer then use
 - C <= ('0' & A) + ('0' & B) - - of course, C must be a BIT longer.



Array Attributes



■ One dimensional

signal s : std_logic_vector (7 downto 3);

```
s'left   = 7
s'right  = 3
s'length = 5
s'high   = 7
s'low    = 3
```

■ Two dimensional

type rom is array (0 to 6, 3 downto 0) of std_logic;

```
r'left(1) = 0           r'right(1) = 6
r'left(2) = 3           r'right(2) = 0
r'high(1) = 6           r'low(1)   = 0
r'high(2) = 3           r'low(2)   = 0
r'length(1) = 7
r'length(2) = 4
```



Quiz I



■ Friday, February 28, 2003

- 1:00 – 2:00 PM, Room 50-340 – Walker
- CLOSED BOOK – No notes, etc.

■ Venue

- Problem Sets 1 and 2, Lab 1, Lectures 1 – 7
- VHDL entities and simple architectures
 - Details of syntax, e.g., placement of semicolons not important
- Understand concurrent statements:
 - Assignment, instantiation, when-else, with-select-when, process
- Understand sequential statements (only within a process):
 - Assignment, if-then-elsif-else, case-when

■ General Topics

- Boolean Algebra, Elementary Logic Expressions
- Combinational Logic, Karnaugh Maps, MSP, MPS
- Latches and Edge Triggered Flip-flops
- PALs (PLDs)

■ The following slides are copies from earlier lectures.



Sum-of-Products & Product-of-Sum



- Product term (or minterm): ANDed product of literals – input combination for which output is true

A	B	C	minterms	
0	0	0	$\bar{A} \bar{B} \bar{C}$	m0
0	0	1	$\bar{A} \bar{B} C$	m1
0	1	0	$\bar{A} B \bar{C}$	m2
0	1	1	$\bar{A} B C$	m3
1	0	0	$A \bar{B} \bar{C}$	m4
1	0	1	$A \bar{B} C$	m5
1	1	0	$A B \bar{C}$	m6
1	1	1	$A B C$	m7

short-hand notation form in terms of 3 variables

F in canonical form:

$$F(A, B, C) = \sum m(1,3,5,6,7)$$

$$= m1 + m3 + m5 + m6 + m7$$

$$F = \bar{A} \bar{B} C + \bar{A} B \bar{C} + A \bar{B} C + A B \bar{C} + ABC$$

canonical form \neq minimal form

$$F(A, B, C) = \bar{A} \bar{B} C + \bar{A} B \bar{C} + A \bar{B} C + ABC + AB\bar{C}$$

$$= (\bar{A} \bar{B} + \bar{A} B + A \bar{B} + AB)C + AB\bar{C}$$

$$= ((\bar{A} + A)(\bar{B} + B))C + AB\bar{C}$$

$$= C + AB\bar{C} = AB\bar{C} + C = AB + C$$

- Sum term (or maxterm) - ORed sum of literals – input combination for which output is false

A	B	C	maxterms	
0	0	0	$A + B + C$	M0
0	0	1	$A + B + \bar{C}$	M1
0	1	0	$A + \bar{B} + C$	M2
0	1	1	$A + \bar{B} + \bar{C}$	M3
1	0	0	$\bar{A} + B + C$	M4
1	0	1	$\bar{A} + B + \bar{C}$	M5
1	1	0	$\bar{A} + \bar{B} + C$	M6
1	1	1	$\bar{A} + \bar{B} + \bar{C}$	M7

short-hand notation for maxterms of 3 variables

L9 6.111 Spring 2003

F in canonical form:

$$F(A, B, C) = \Pi M(0,2,4)$$

$$= M0 \cdot M2 \cdot M4$$

$$= (A + B + C)(A + \bar{B} + C)(\bar{A} + B + C)$$

canonical form \neq minimal form

$$F(A, B, C) = (A + B + C)(A + \bar{B} + C)(\bar{A} + B + C)$$

$$= (A + B + C)(A + \bar{B} + C)$$

$$= (A + B + C)(\bar{A} + B + C)$$

$$= (A + C)(B + C)$$

Introductory Digital Systems Laboratory 17



K-Map Examples



Cin		A			
		00	01	11	10
0	0	0	0	1	0
	1	0	1	1	1

Cout =

C		A			
		00	01	11	10
0	0	1	0	0	1
	1	0	0	1	1

$$F(A,B,C) = \sum m(0,4,5,7)$$

F =

C		A			
		00	01	11	10
0	0	0	0	1	1
	1	0	0	1	1

F(A,B,C) =

C		A			
		00	01	11	10
0	0	0	1	1	0
	1	1	1	0	0

F' simply replace 1's with 0's and vice versa

$$F'(A,B,C) = \sum m(1,2,3,6)$$

F' =

L9 6.111 Spring 2003

Introductory Digital Systems Laboratory 18



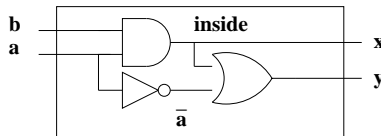
Not Needing Mode BUFFER



■ We will not use mode BUFFER.

- This makes it easier to use components as VHDL is very strongly typed and one has to declare signal types that are the same as those used by the component. This eliminates confusion between modes OUT and BUFFER.
- When we want to use an output internally we will declare a signal in the ARCHITECTURE to use internally and assign the output to this internal signal.

```
Library ieee;
use ieee.std_logic_1164.all;
entity foo is
  port(a, b: in std_logic;
        x, y: out std_logic);
end foo;
```



```
architecture no_buffer_mode of
foo is
  signal inside: std_logic;
begin
  inside <= a AND b;
  x <= inside;
  y <= inside OR (not a);
  -- really wanted y <= x OR (not a);
end no_buffer_mode;
```



Designs



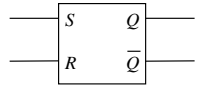
■ Designs consist of an ENTITY/ARCHITECTURE pair.

- They are usually in the same file. This is a good idea.
- A file can contain multiple ENTITY/ARCHITECTURE pairs.
 - However one should declare ENTITY/ARCHITECTURE pairs before they are used in another architecture.
- Altera's MAX+PlusII requires the file name to be xxx.vhd where xxx is the name of an ENTITY in the file.

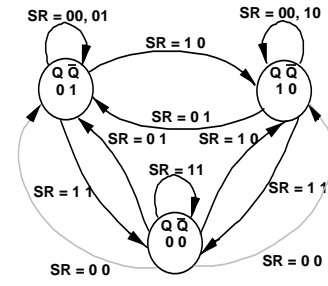
■ ARCHITECTURE bodies can have two types of statements.

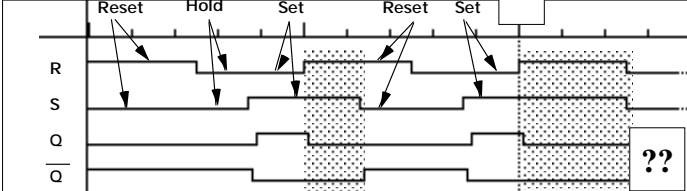
- CONCURRENT
 - Signal assignment
 - Instantiation
 - When/else
 - With/select
 - Process (as a wrapper for sequential statements)
- SEQUENTIAL (only within a process – more later)
 - Signal assignment
 - If/then/elsif/else
 - Case/when

NOR-based Set-Reset (SR) Flipflop



S	R	Q	\bar{Q}
0	0	Q	\bar{Q}
1	0	1	0
0	1	0	1
1	1	0	0

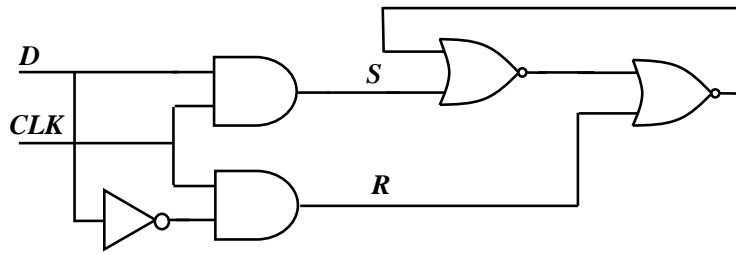
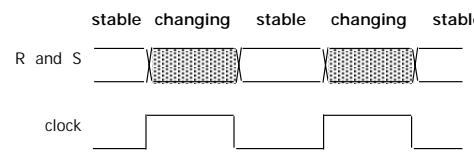


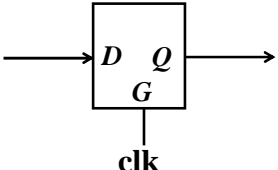


- Flip-flop refers to a bi-stable element (edge-triggered registers are also called flip-flops) – this circuit is not clocked and outputs change “asynchronously” with the inputs

L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 21

Making a Clocked Memory Element: Positive D-Latch

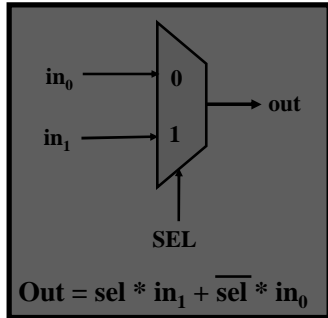


- A Positive D-Latch: Passes input D to Q when clk is high and holds state when clock is low (i.e., ignores input D)
- A Latch is level-sensitive: invert clock for a negative latch

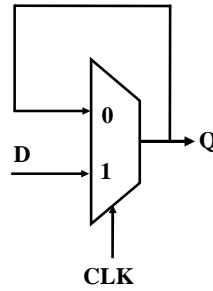
L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 22

Multiplexor based positive & negative latch

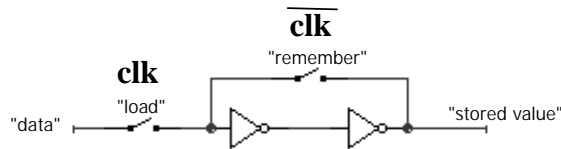
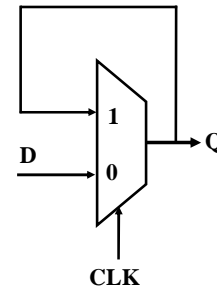
2:1 multiplexor



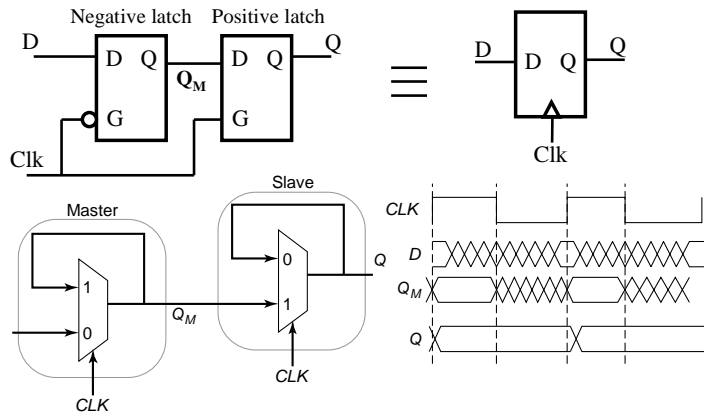
Positive Latch



Negative Latch



Building an Edge-Triggered Register



■ Master-Slave Register

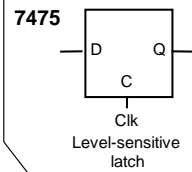
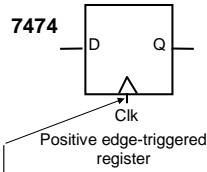
- Use negative clock phase to latch inputs into first latch
- Use positive clock to change outputs with second latch

■ View pair as one basic unit

- master-slave flip-flop twice as much logic



Latches vs. Edge-Triggered Register

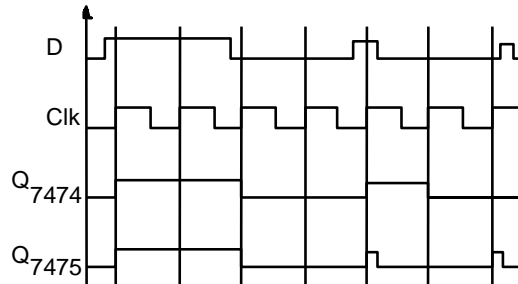


Bubble here for negative edge triggered register

Edge triggered device sample inputs on the event edge

Transparent latches sample inputs as long as the clock is asserted

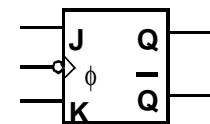
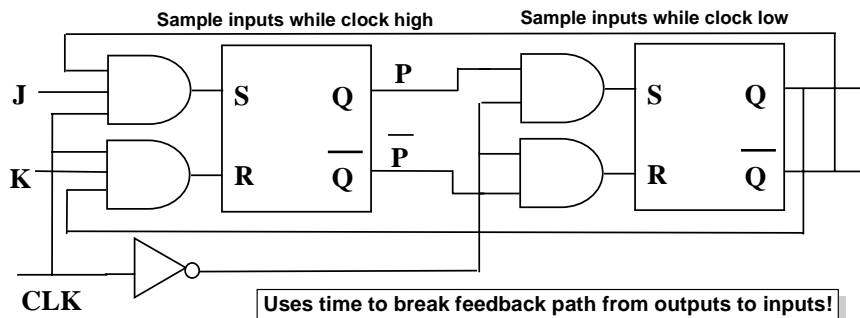
Timing Diagram:



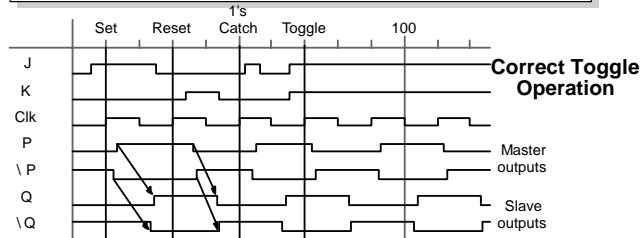
Behavior the same unless input changes while the clock is high



J-K Master-Slave Register

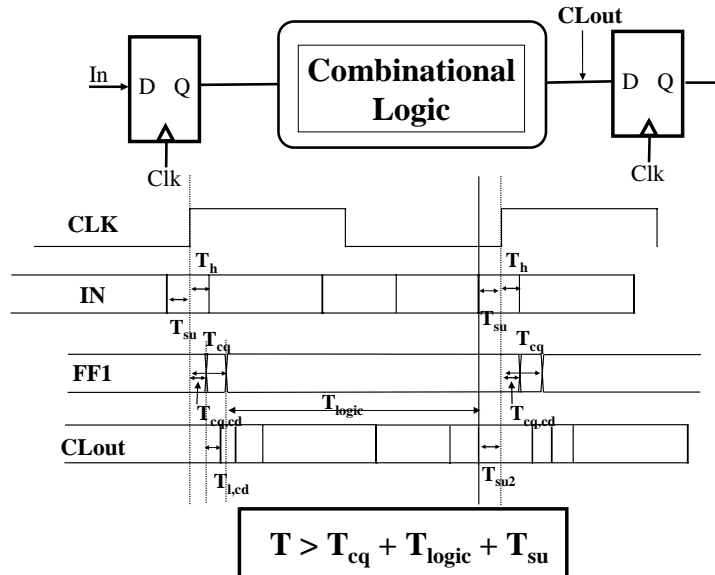


J-K Logic Symbol

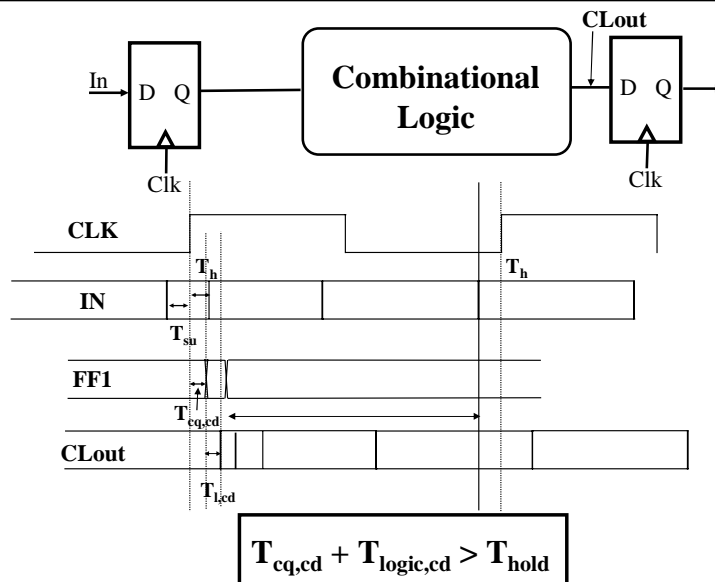




System Timing (I): Minimum Period



System Timing (II): Minimum Delay



Clocks are not perfect: Clock Skew

$T > \underline{\hspace{2cm}}$

L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 29

Catalog Counter: 74163

74163 Synchronous
4-Bit Upcounter

Synchronous Load and Clear Inputs

Positive Edge Triggered FFs

Parallel Load Data from D, C, B, A

P, T Enable Inputs: both must be asserted to enable counting

RCO: asserted when counter enters its highest state 1111, used for cascading counters "Ripple Carry Output"

CLR and LOAD are synchronous
If CLR = 0 then Q := 0
Else if LOAD then Q := D
Else if P * T = 1 then Q := Q + 1
Else Q := Q

L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 30



VHDL for 74163



```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity c74163 is
  port (LDN, CLRN, P, T, CLK: in std_logic;
        D : in std_logic_vector (3 downto 0);
        count : out std_logic_vector (3 downto 0);
        RCO : out std_logic);
end c74163;

architecture behavior of c74163 is
  signal Q: std_logic_vector (3 downto 0);

  begin
    count <= Q;
    RCO <= Q(3) and Q(2) and Q(1) and Q(0) and T;

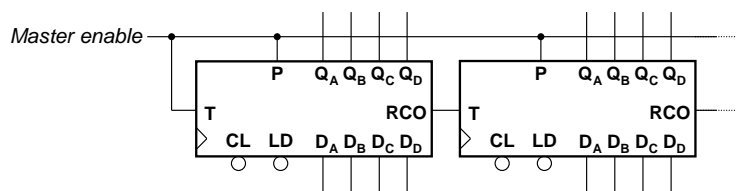
    process
    begin
      wait until (CLK'event and CLK = '1');
      if CLRN = '0' then Q<= "0000";
      elsif LDN = '0' then Q<= D;
      elsif (P and T) = '1' then Q<= Q+ 1;
      end if;
    end process;
  end architecture behavior;

```

- Our entire counter example can be defined in one process
 - Behavioral, rather than structural specification
 - Synthesis tools will map behavior to logic
 - This is the more typical style of coding



Cascading 74163s (the right way)



- '163 is enabled only if P and T are high
- RCO goes high when '163 is about to roll over to 0000
- Connecting RCO to P makes second '163 increment once whenever first '163 rolls over
- P on first '163 is connected to master enable

Counter Tricks

Configured to automatically reset at 1011

Counts 0000, 0001, ... , 1010, 1011, 0000, ...

Configured to automatically load 0100

Counts 0100, 0101, ... , 1110, 1111, 0100, ...

L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 33



MCM6264C 8k x 8 Static RAM

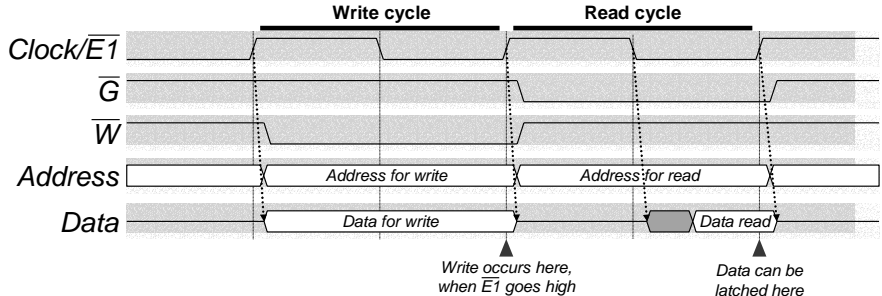
- Same (bidirectional) data bus used for reading and writing
- Chip Enables ($\overline{E1}$ and $E2$)
 - $\overline{E1}$ must be low and $E2$ must be high to enable the chip
- Write Enable (\overline{W})
 - When low (and chip is enabled), the values on the data bus are written to the location selected by the address bus
- Output Enable (\overline{G})
 - When low (and chip is enabled), the data bus is driven with the value of the selected memory location

HC	1	25	VCC
A0	2	23	WE
A1	3	20	E1
A2	4	18	A0
A3	5	14	A1
A4	6	12	A2
A5	7	11	A3
A6	8	7	A4
A7	9	6	A5
A8	10	5	A6
CS0	11	16	CS0
CS1	12	17	CS0
CS2	13	16	CS0
Y0	14	15	CS0

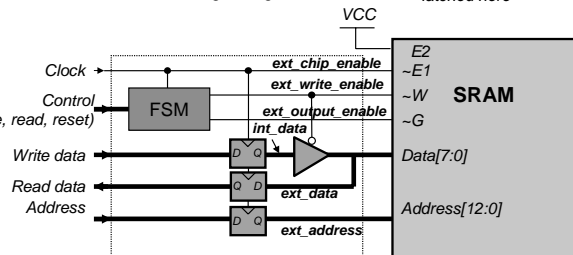
L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 34

Sample Memory Interface Logic





- Drive memory enable with clock
 - Ensures data and memory busses are stable for writes
 - Minimum clock period is twice memory access time



L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 35

Testing Memory

- An idea that almost works
 1. Write 0 to location 0
 2. Read location 0, compare value read with 0
 3. Write 1 to location 1
 4. Read location 1, compare value read with 1
 5. ...
- What is the problem?
 - Suppose the memory was missing (or output enable was disconnected)

Address	0	1	2	3				
Data	0	0	1	1	2	2	2	2
Control	Write	Read	Write	Read	Write	Read	Write	Read

Data bus is undriven but wire capacitance briefly maintains the bus state: memory appears to be ok!

L9 6.111 Spring 2003
Introductory Digital Systems Laboratory 36



Testing Memory



Write to all locations, then read back all locations

- Separates read/write to the same location with reads/writes of different data to different locations
- (both data and address busses are changed between read and write to same location)

- Write 0 to address 0
- Write 1 to address 1
- ...
- Write ($n \bmod 256$) to address n
- Read address 0, compare with 0
- Read address 1, compare with 1
- ...
- Read address n , compare with ($n \bmod 256$)

