Computer Organization
(Gate Level and Word Level Designs)

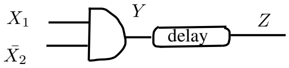
KR Chowdhary
Professor & Head
Email: kr.chowdhary@gmail.com
webpage: krchowdhary.com

Department of Computer Science and Engineering
MBM Engineering College, Jodhpur

November 14, 2013

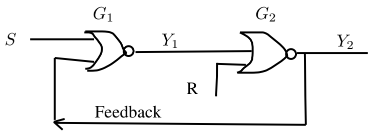
Sequential circuits

Propagation Delay(Δ):



$z(t+\Delta) = y(t) = x_1(t).\bar{x_2}(t)$. The y is internal state variable.

SR Flipflop:



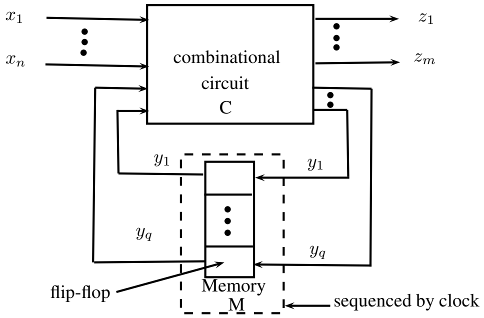
Simple SR Flip-flop

- For SR=00, $y_1 = \bar{y_2} = 0$, or $y_1 = \bar{y_2} = 1$.
- For SR=10, $y_1 = \bar{y_2} = 0$.
- For SR = 01, $y_2 = \bar{y_1} = 0$.

Race Condition:

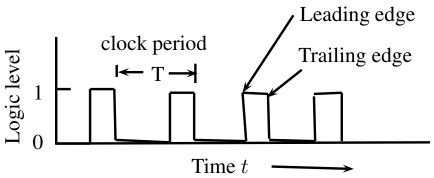
- Let SR =11. Thus, $y_1 = y_2 = 0$ after Δ time (i.e., propagation delay). Now let, SR becomes 00. After delay Δ, $y_1, y_2 = 11$. After further Δ time, $y_1, y_2 = 00$. Then it changes back to 00, and so on.
- Thus, output Oscillates between 00 and 11. For this to happens, $\Delta_1 = \Delta_2$.
- $SR = 11$ is forbidden state. $\Delta_1 \neq \Delta_2$?

Sequential circuits-2



- Race conditions are eliminated by timing signals (clock)
- Sequential circuits + clock = Synchronous circuits
- Circuits not timed by clock are asynchronous circuits (Prone to race conditions. Why?)
- Triggering takes place on rising or trailing edge of clock pulse. Before this, the signals are supposed to be stabilized.

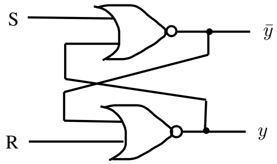
Clock



Waveform of a typical clock signal

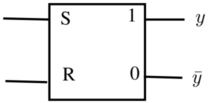
- Clock active and inactive states: 1, 0. They also correspond to transition from 0-to-1 or 1-to-0.
- The period, during the time clock is inactive, must be atleast as long as the worst case signal propagation through C, and active period must be long enough to allow M to make one complete transition.
- Behaviour of seq. ckt is described by State table.
- What happens if clock period is long and signal passed through forward path feeds back to I/P during true period of CLK?

SR Flip-flop



SR Flip-flop Logic diagram

$\equiv$

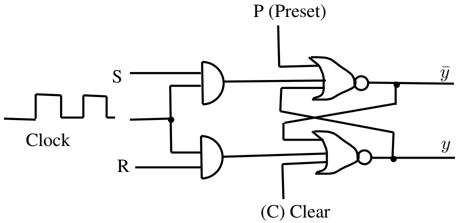


SR FF Symbol

- For SR=00, there is no change in output $y, \bar{y}$.
- SR=11 is forbidden. The S(set)=1, R(reset)=0 Sets true o/p y to 1
- SR=01 Resets true o/p o/p y to 0.
- Let Δ is propagation delay of a gate, and $\tau \approx 2\Delta$.

$$\begin{aligned} y(t+2\Delta) &= \overline{R(t+\Delta) + \overline{[S(t) + y(t+\Delta)]}} \\ &= \bar{R}(t+\Delta)[S(t) + y(t+\Delta)] \\ y(t+\tau) &\approx \bar{R}(t)[S(t) + y(t)] \end{aligned}$$

Clocked SR Flip-flop

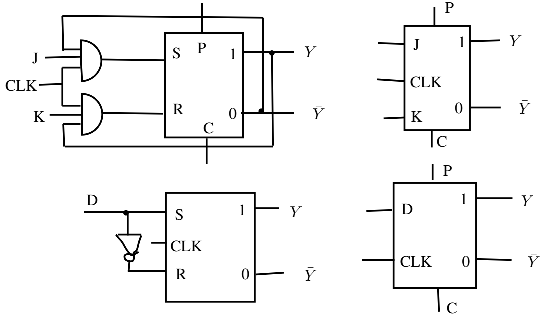


SR flip-flop Truth-Table-1

current state (y)	Input SR 00	01	10
0	0	0	1
1	1	0	1

- From table-1 above, using minimization, $y(t+1) = \bar{R}(t)[S(t)+y(t)]$, where $y(t+1)$ is new state and $y(t)$ is previous state.
- Other FFs are derived from SR.
- No state change occurs without clock pulse
- The clock pulse should be sufficiently long, to allow the inputs to stabilize
- P, C inputs are Preset/Clear to Set/ Clear without clock

JK and D Flip-flop



Logic circuit

Symbol

table-2

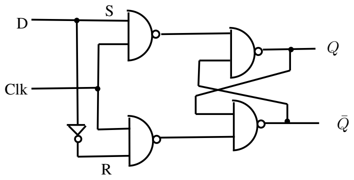
state y	Input JK 00	01	10	11
0	0	0	1	1
1	1	0	1	0

Y = True output, C = Clear signal

- In JK (derived from SR FF). From table-2, using minimization, $y(t+1) = J(t)\bar{y}(t) + \bar{K}(t)y(t)$
- for JK=00, there is NO change in output, 10 and 01 causes set and reset of FF, and JK=11 toggles the FF (i.e., I/P 11 is not forbidden).
- D (Delay)FF has single input only. With $D = 1$, it sets, and with $D = 0$ it resets the FF. $y(t+1) = D(t)$

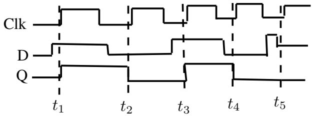
Gated D-Latch

- Most commonly used FF are D-FF because they are useful for temporary storage of data
- SR input will always be 01/10 only. 00/11 input are non-existent, and undefined.
- During the positive half of clock, D should be stable. Else transition will occur at Q.



Truth-table

Clk	D	Q(t+1)
0	X	Q(t)
1	0	0
1	1	1



Symbol

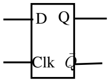
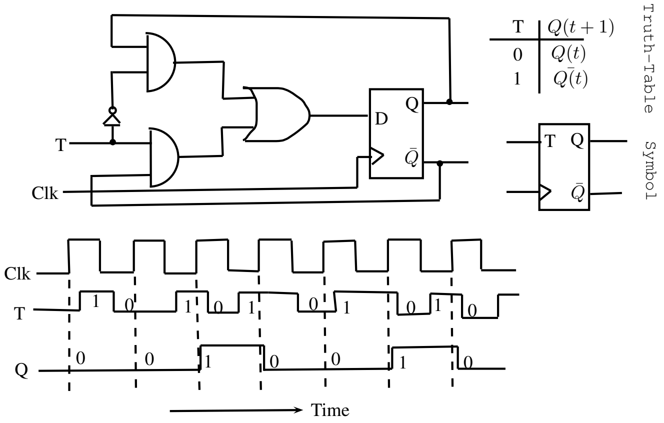


Figure 1: D-flipflop.

T-Flip-flop

- A T-FF changes its state every clock cycle(toggles) if its input T is equal to 1.
- The triggering (level change at o/p) takes place at positive of CLK.



T	$Q(t+1)$
0	$Q(t)$
1	$\bar{Q}(t)$

Figure 2: T-Flip-flop.

Registers and Shift Registers

- Registers store machine words. Shift and rotate data.
- below given circuit is shift-right register.

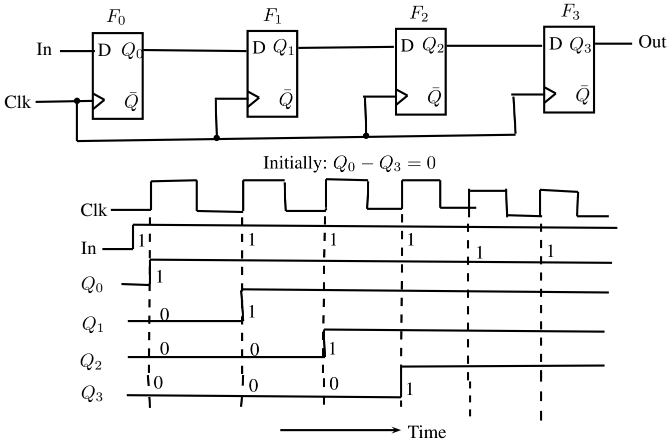


Figure 3: Shift-right register.

Parallel Shift Registers

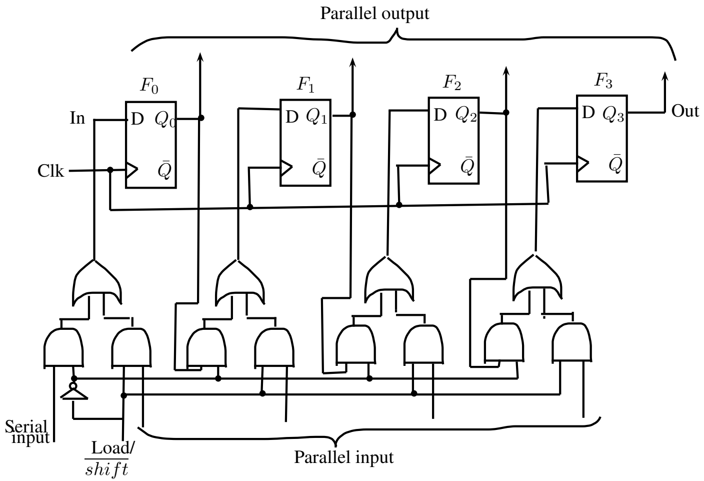


Figure 4: Parallel Shift register.

4-bit up-counter

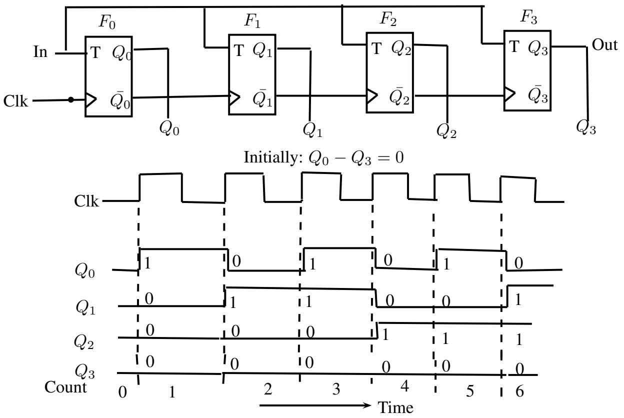
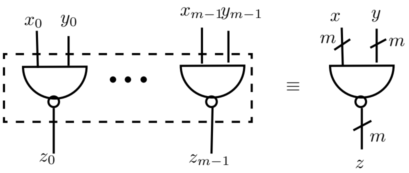
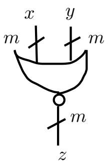


Figure 5: Asynchronous counter.

- Counters generate control and timing signals
- Q_0 freq.$=$ $clk/2$, q_1 fre. $=$ $clk/4$, ..., called ripple counter.
- If all FF change state together $\Rightarrow$ *synchronous counter*

Word Gates



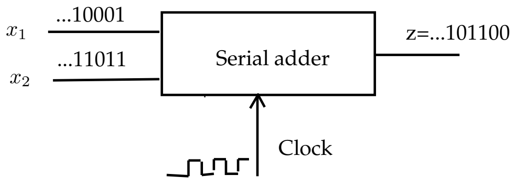


Similarly, the NOR gate is also represented.

$X = (x_0, x_1, \ldots, x_{m-1})$,
$Y = (y_0, y_1, \ldots, y_{m-1})$ are
(say) binary words.
$\therefore, z = f(X, Y)$,
if $z_i = f(x_i, y_i)$ for $i = 0, \ldots, m-1$.
$\therefore, Z = \overline{XY} \Rightarrow (z_0, z_1, \ldots, z_{m-1})$
$= (\overline{x_0 y_0}, \overline{x_1 y_2}, \ldots, \overline{x_{m-1} y_{m-1}})$.

Figure 6: m-bit NAND word gate.

Serial addition

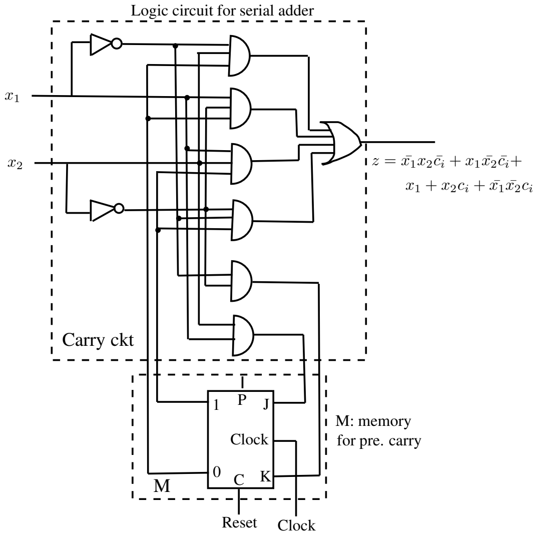


- Two-bit addition takes place for each clock pulse. $z(t) = x(t) + y(t)$.
- It may produce a carry bit $c(t)$. Thus two states (S_0, S_1) are possible, for $c(t-1) = 0$, and $c(t-1) = 1$. Thus, $z(t) = x(t) + y(t) + c(t-1)$.

	input x_1x_2			
state	00	01	10	11
s_0	$s_0, 0$	$s_0, 1$	$s_0, 1$	$s_1, 0$
s_1	$s_0, 1$	$s_1, 0$	$s_1, 0$	$s_1, 1$

- Each entry has form $s(t+1), z(t)$. y is FF's state variable, $y = 0$ for s_0, and $y = 1$ for s_1.

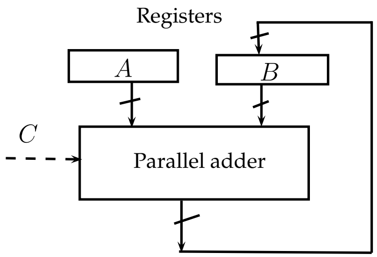
Logic circuit for Serial addition



Logic circuit for serial adder
x_1
x_2
Carry ckt
$z = \bar{x_1}x_2\bar{c_i} + x_1\bar{x_2}\bar{c_i} +$
$x_1 + x_2c_i + \bar{x_1}\bar{x_2}c_i$
1
P
J
Clock
0
C
K
M
M: memory
for pre. carry
Reset
Clock

Figure 7: Serial Adder

Word Level Design: Parallel addition



$$b_i \leftarrow a_i + b_i + c_{i-1}, \text{(for } i = 0 \ldots n-1, c_{-1} = 0)$$

Figure 8: Word adder

K input m-bit multiplexer

- There are k input lines, each m-bit wide.
- The number of select lines p to select a unique input are given by expression $k = 2^p$, where total number of inputs are k.
- Source selection is determined by an *encoded* pattern of p-bits.
- $z_i = \sum_{i=0}^{k-1} x_{ij} a_i e$, for $j = 0, 1, ..., m-1$

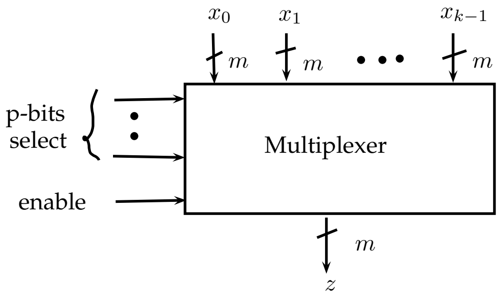


Figure 9: k-input m-bit multiplexer.

2 input 4-bit Multiplexer

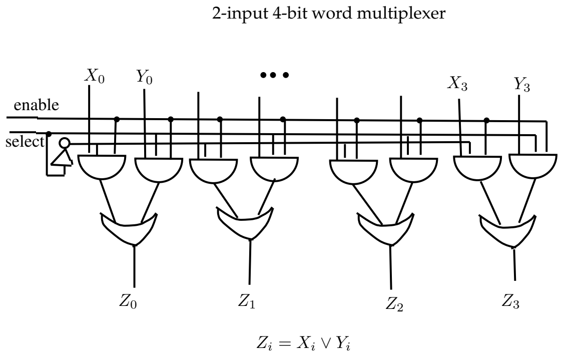


Figure 10: 2-1 multiplexer

- It has two input words, each 4-bit long (x_0, x_1, x_2, x_3 and y_0, y_1, y_2, y_3).
- Select line selects either $x_0x_1x_2x_3$ or $y_0y_1y_2y_3$, and delivers at o/p as $z_0z_1z_2z_3$.
- expression?

Decoders and Encoder

- **Decoder:** 1-out-of-2^n or $1/2^n$ decoder is a combinational circuit with n input lines and 2^n o/p data lines.
- For input x_0x_1=00, 01, 10, 11 the output $z_0z_1z_2z_3$ is 1000, 0100, 0010, 0001 respectively. Hence 2- to-4 decoder.

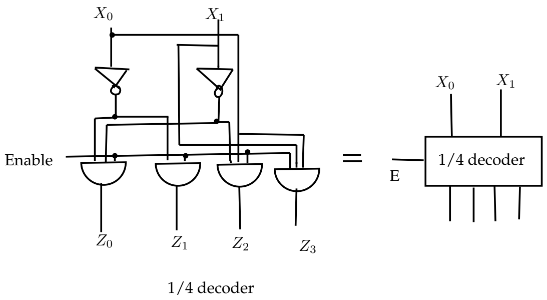


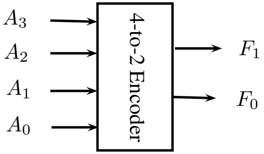
Figure 11: 2-4 decoder

- **Encoder:** generates the address. It has inverse function of decoder. Has 2^k i/p lines and k output lines.

4-to-2 encoder

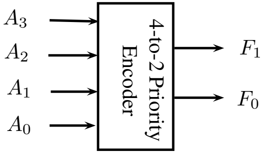
Simple Encoder: It is too much restricted. For the required outputs, the input should be exactly as shown. If input deviates from the given, the output is undecidable.

A_3	A_2	A_1	A_0	F_1	F_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



Priority Encoder: It generates the output corresponding to the bit it does for simple encoder. X indicates do'nt care bits.

A_3	A_2	A_1	A_0	F_1	F_0
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1



Practice problems

1. Find a minimum cost implementation of the function $f(x_1, x_2, x_3, x_4)$, where $f = 1$ if either one or two of the input variables have the logic value 1. Otherwise, $f = 0$.
2. Prove that associative rule does not apply to the *NAND* operator.
3. How you will construct a down counter, i.e, from 1111 to 0000 ?
4. The address lines $A_0 \ldots A_{15}$, use encoder to access the corresponding physical address (T/F).