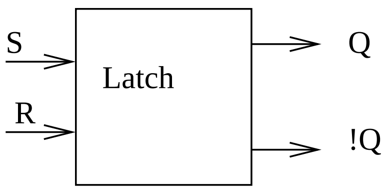
Flip - Flops, Registers and Counters S-R Latch

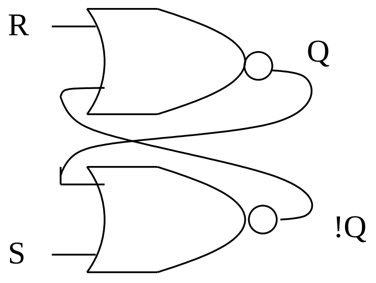
To store 1 or 0 using two inputs S , R

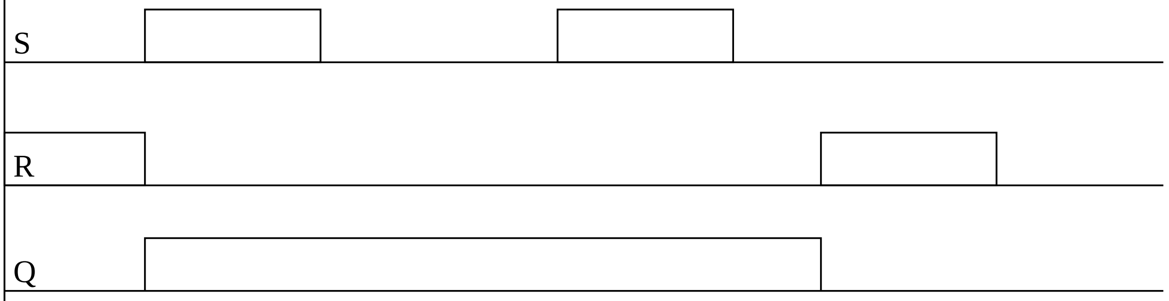


Symbol

S	R	Q	!Q
0	0	Q	!Q
1	0	1	0
0	1	0	1
1	1	0	0

Function Table

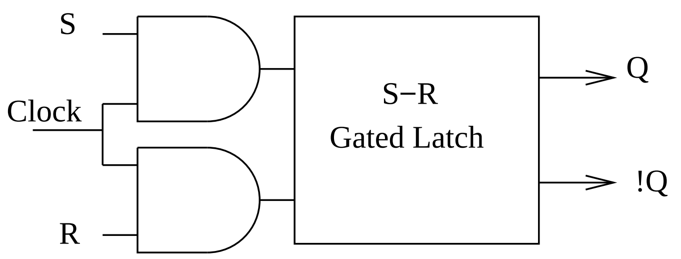




Timing

Gated S-R Latch

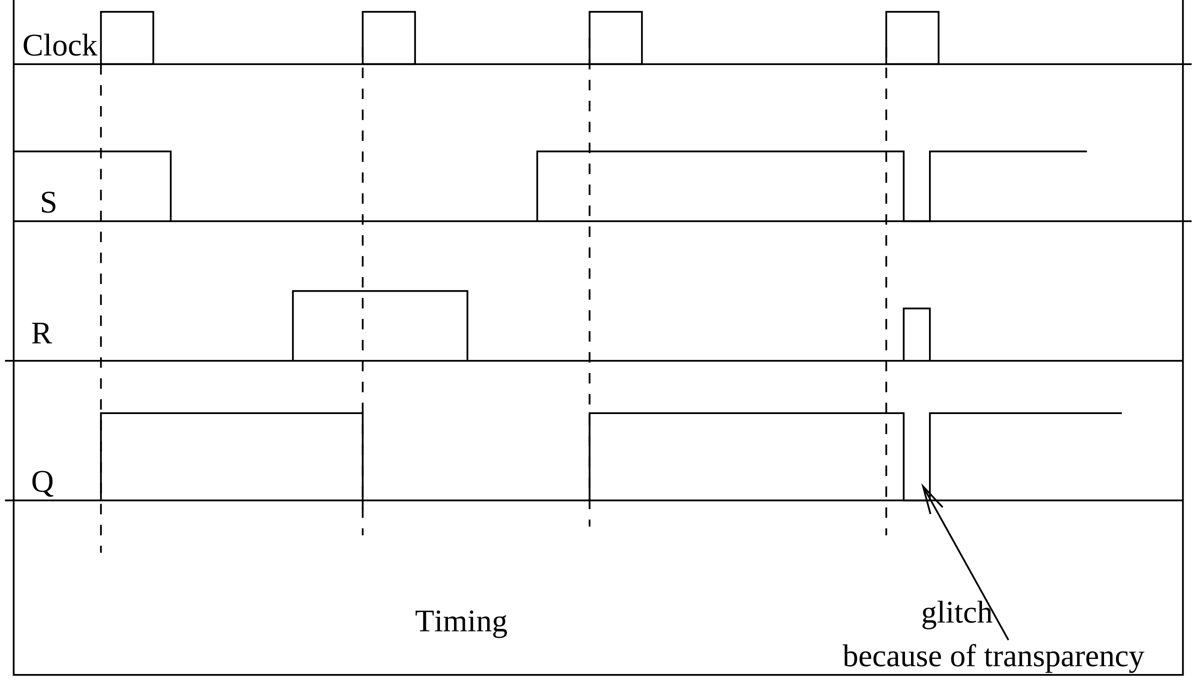
Only change content of latch when Enable signal "clock" =1



Symbol

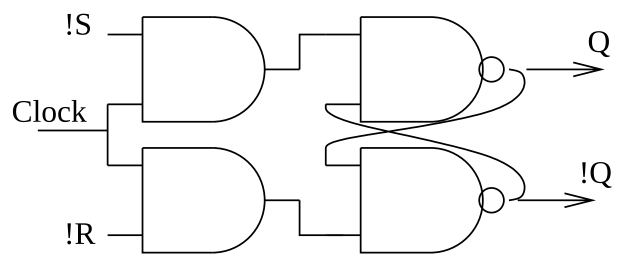
Clock	S	R	Q
0	X	X	Q
1	1	0	1
1	0	1	0
1	0	0	Q

Function table



Timing

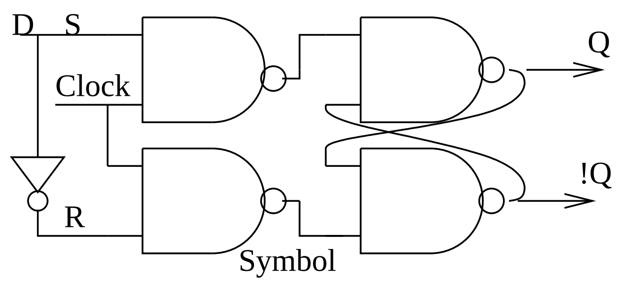
S-R Latch and D- Latch with NAND gates



Symbol

Clock	!S	!R	Q
0	X	X	Q
1	1	0	0
1	0	1	1
1	1	1	Q

Function table



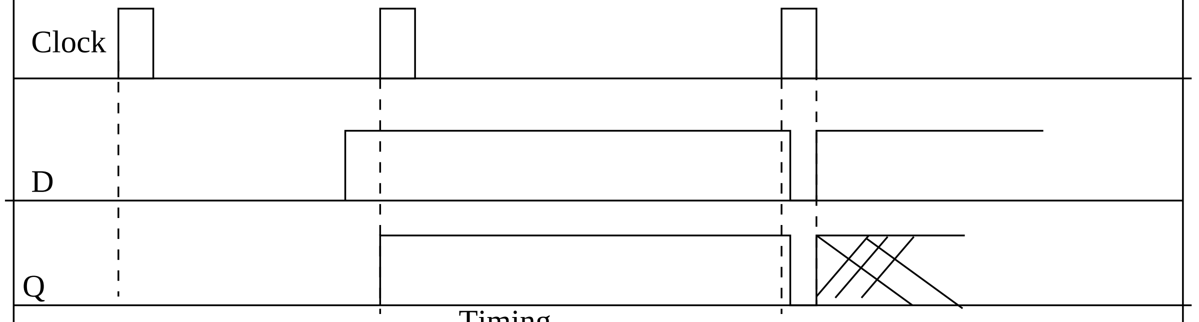
Symbol

Clock	S	R	Q
0	X	X	Q
1	1	0	1
1	0	1	0
1	0	0	Q

Function table

For D Latch: If clock=1 Q=D

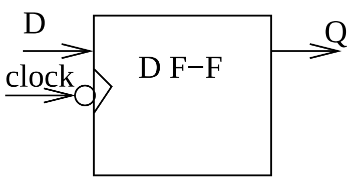
If clock=0 Q=Q



Timing

Master-Slave Flip-Flop

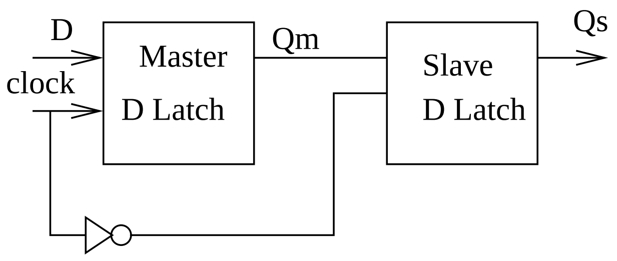
Makes -ve edge triggered Flip-Flop



Symbol

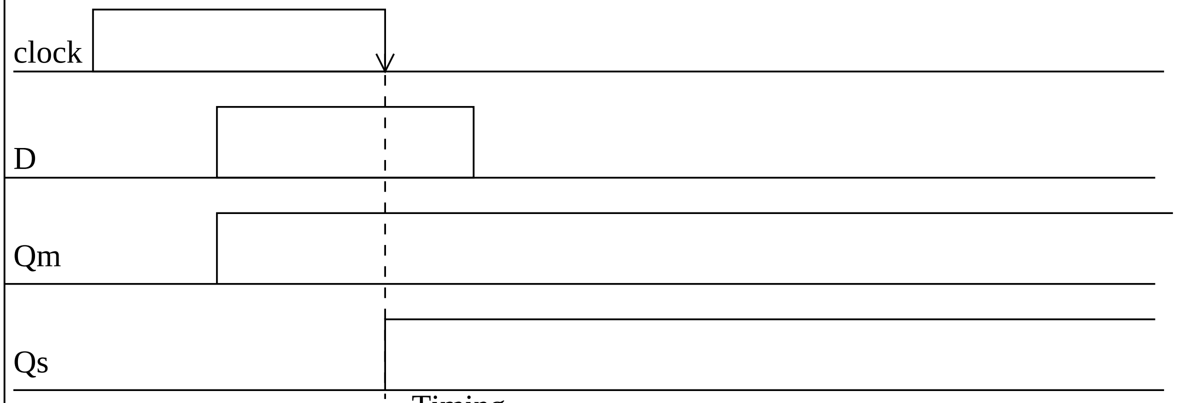
clock	D	Q
↓	D	D
1/0	X	Q

Function Table



When clock=1 Qm=D, and QS=QS because clocks=0

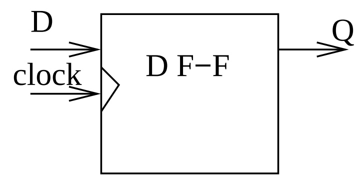
When clock=0 Qm=Qm (D) and Qs= Qm(D)



Timing

©N. Mekhiel

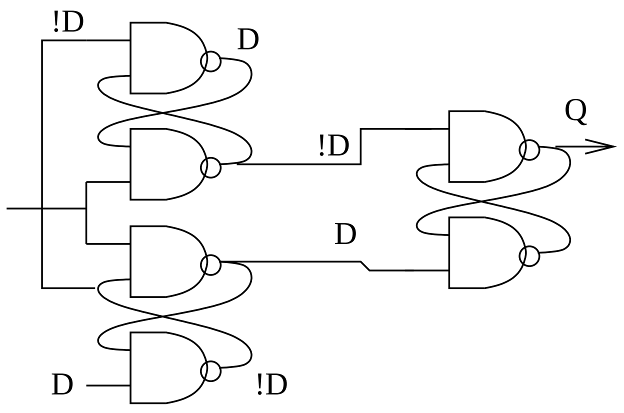
Positive edge triggered F-F

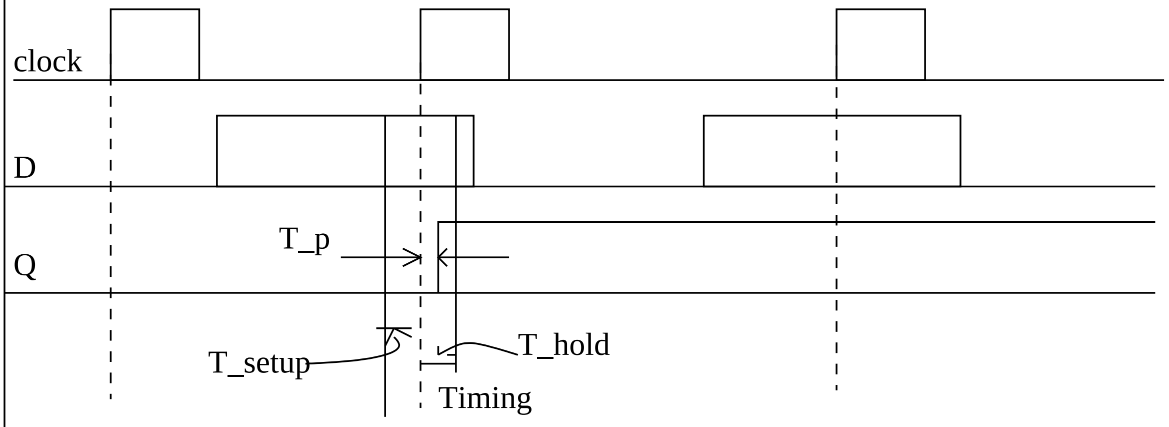


Symbol

clock	D	Q
↑	D	D
1/0	X	Q

Function Table

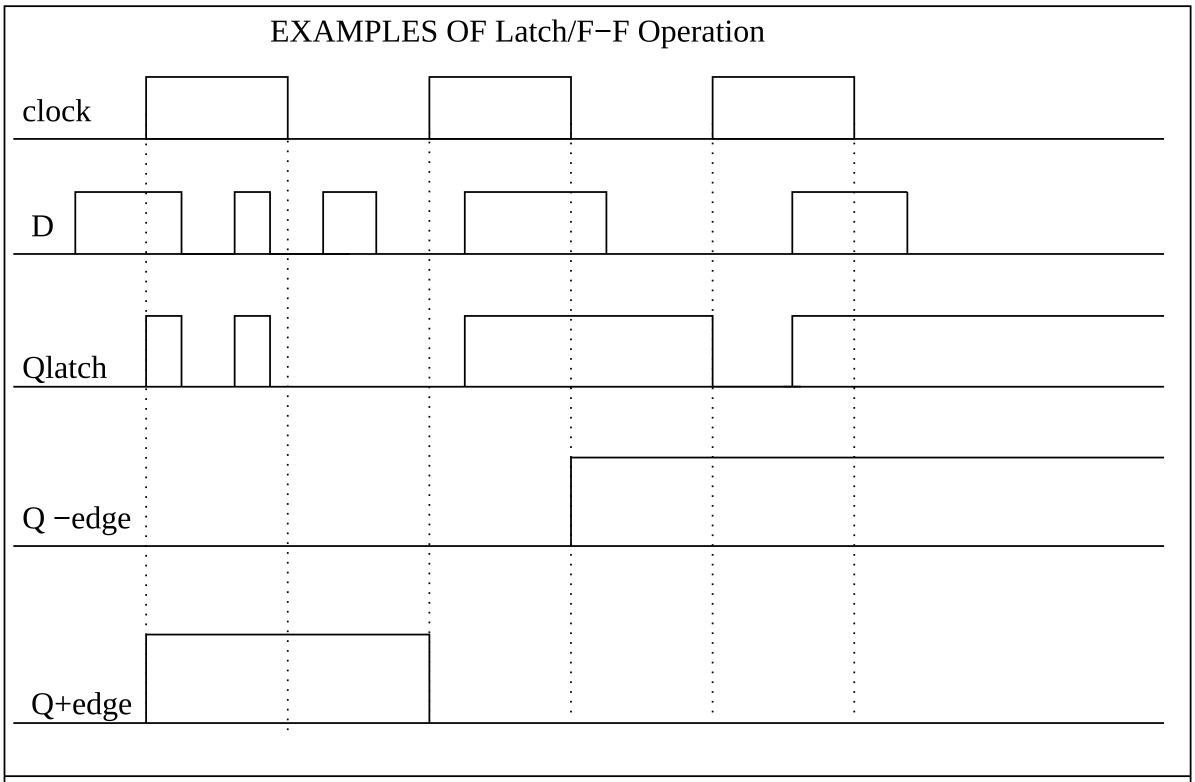


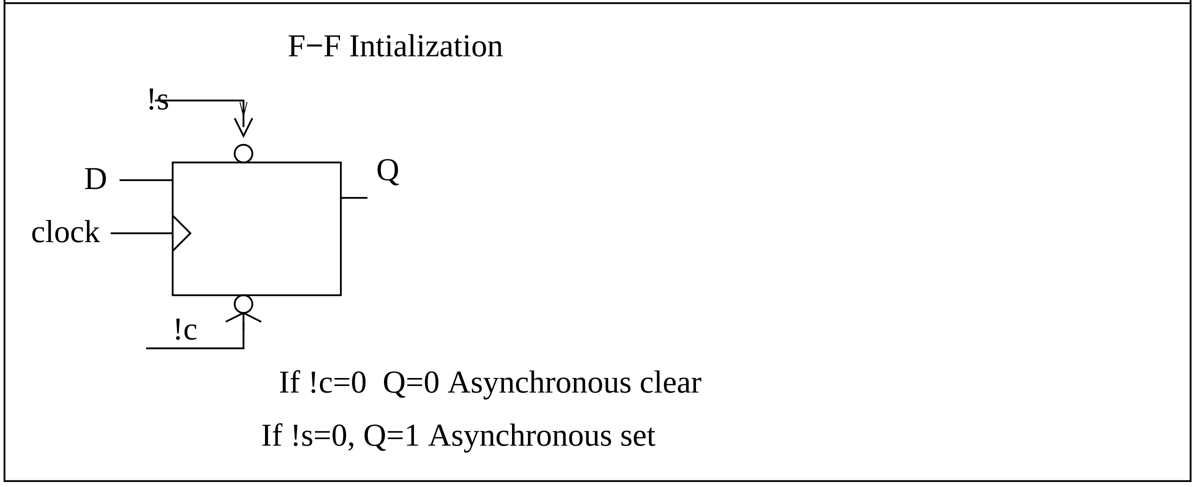


Timing

©N. Mekhiel

Examples for Latch and F-F Operations

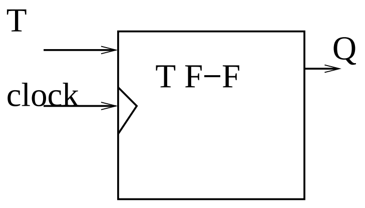


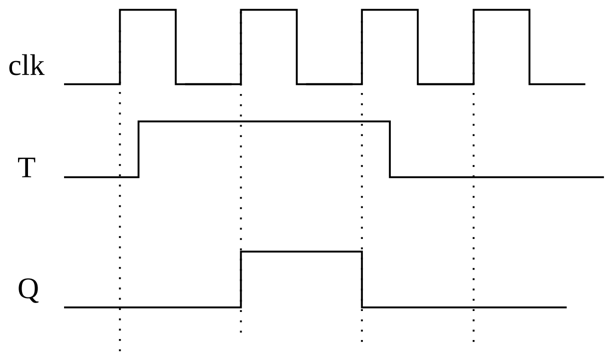


If !c=0 Q=0 Asynchronous clear

If !s=0, Q=1 Asynchronous set

T Flip Flop and J-K Flip Flop

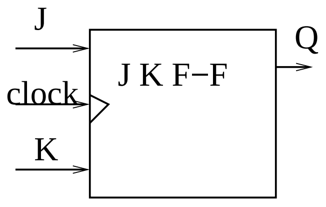


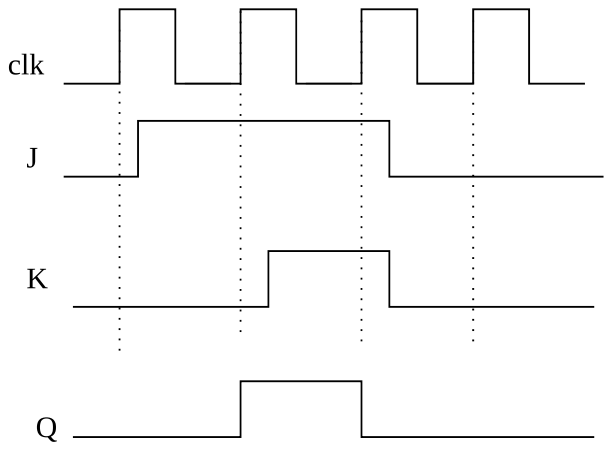


at +edge of clock Q=!Q if T=1
and Q=Q if T=0

Implementation, use D F−F

D=T.!Q + !T.Q



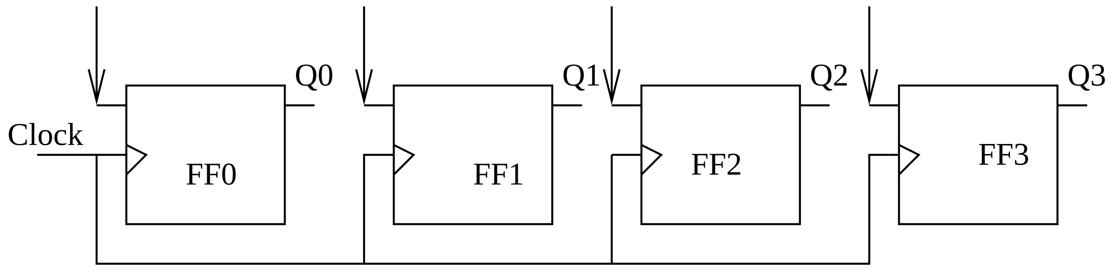


at rising edge of clock:

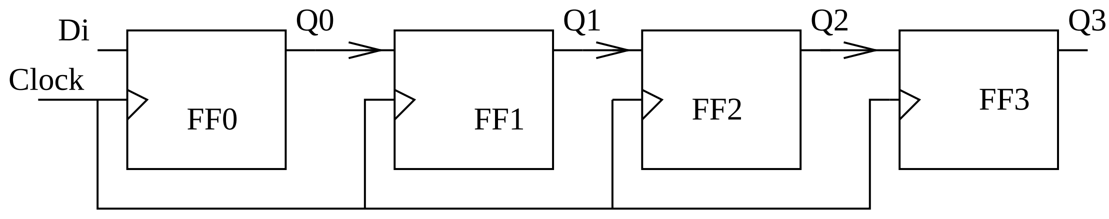
Q=0 if J=0, K=1
Q=1 if J=1, K=0
Q=!Q if J=K=1
Q=Q if J=K=0

Implementation uses D F−F
D= J.!K + J.K. !Q + !J.!K.Q
=Q.!K + J.!Q

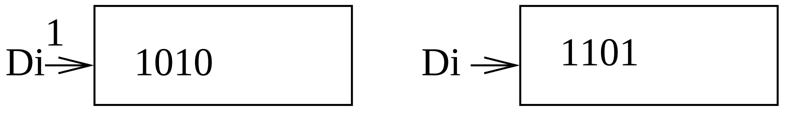
Parallel Registers and Shift Registers



4−Bit Parallel Register



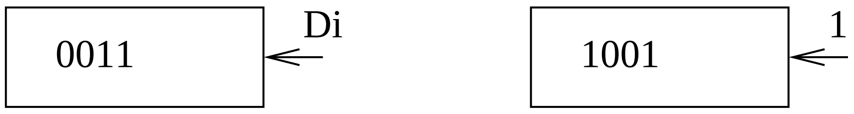
Shift Register



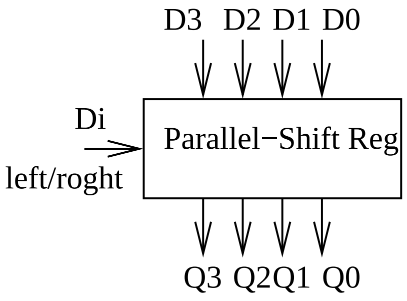
Operation =SR

©N. Mekhiel

Parallel- Shift Registers



Operation= SL

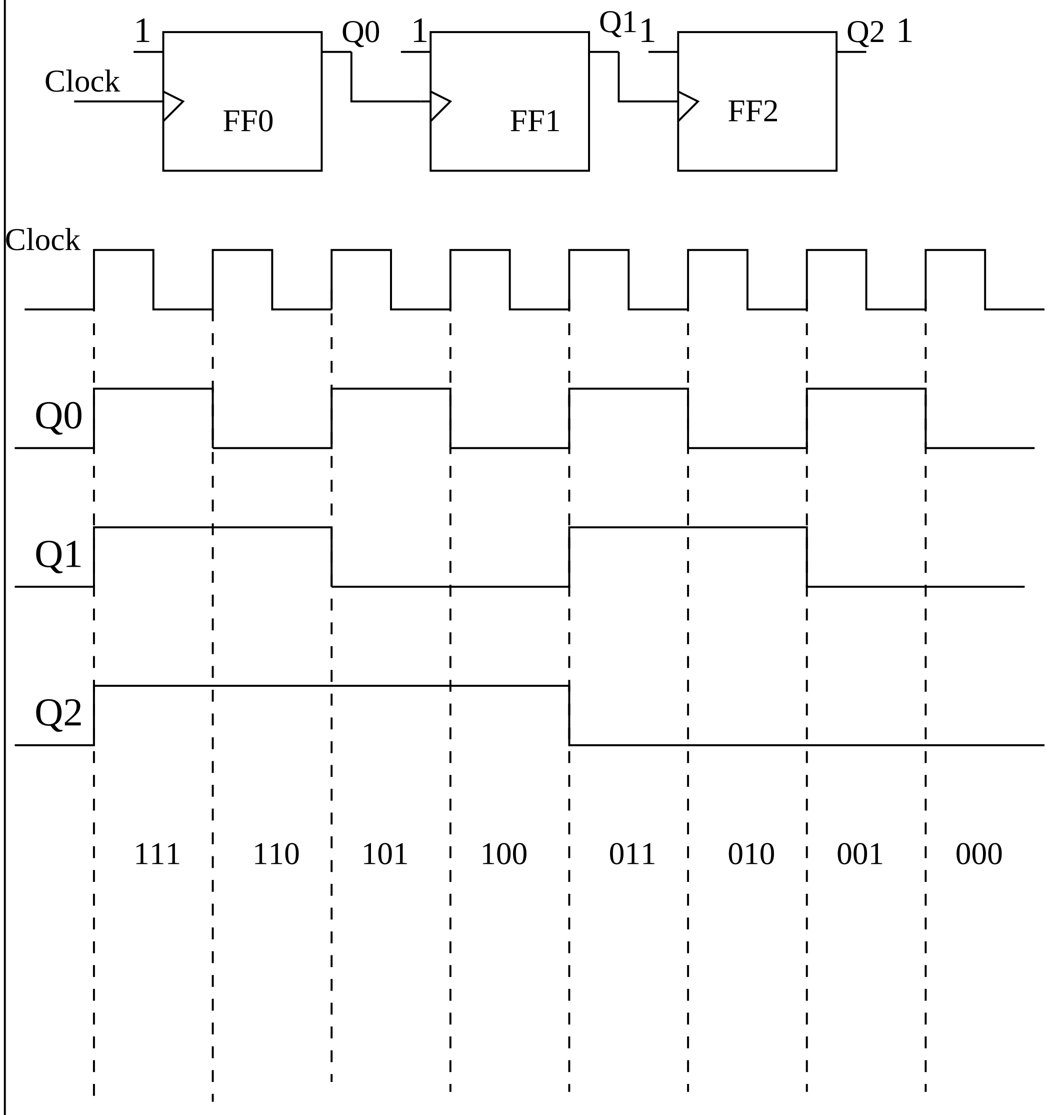


Implementation

Di= Pi.load + Qi+1. !load.SR + Qi−1.!load.SL

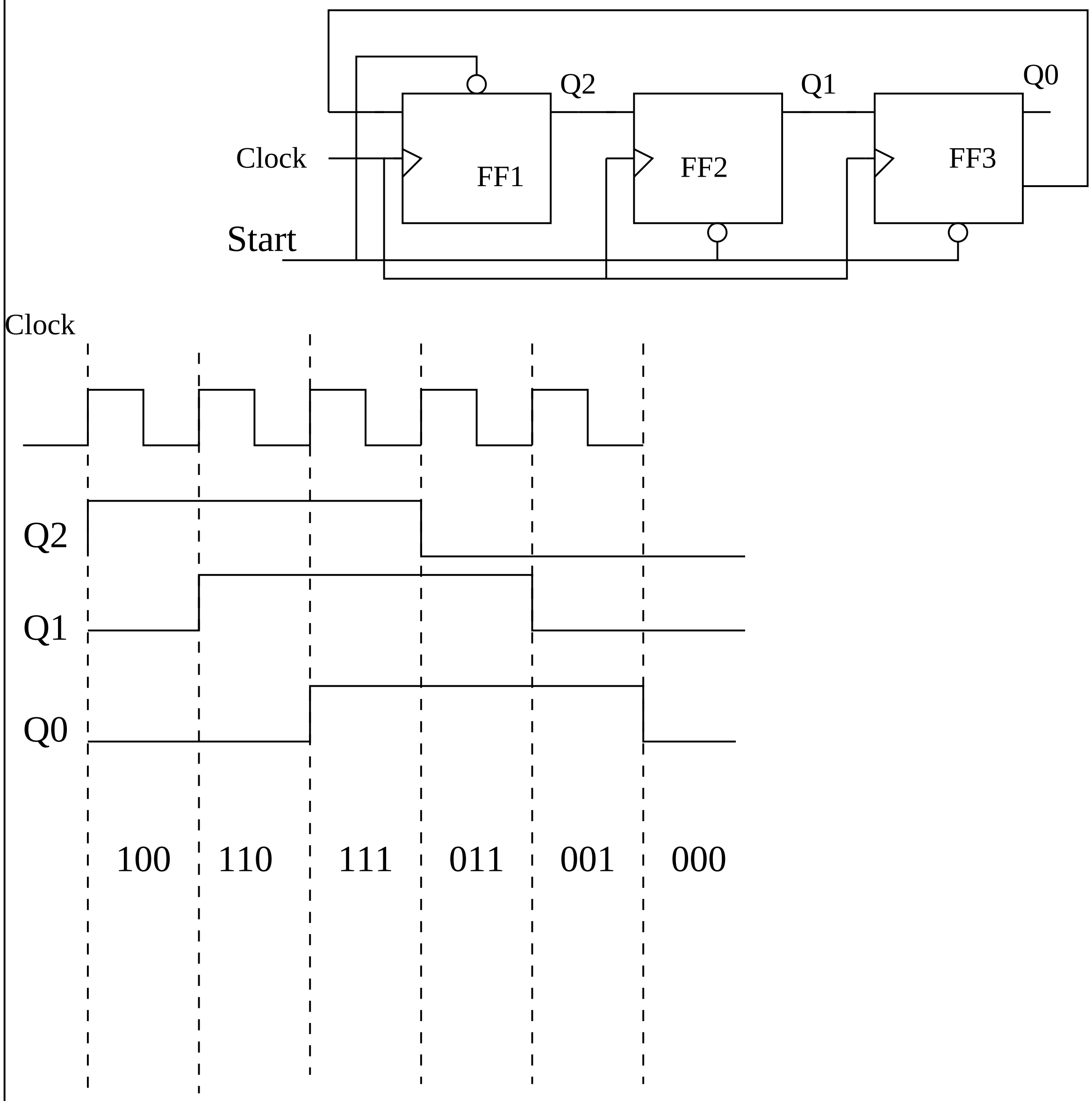
Counters

1-Ripple Counter (problem with delay)



Ripple Counter (count down) modulo 8

2-Ring Counter (Johnson)



Ring Counter (Johnson) modulo 6

2-Synchronous Counters

Connect all Flip-Flop to same clock (Synchronous)

bf Using T F-F

clock	**Q3**	**Q2**	**Q1**	**Q0**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
-	-	-	-	-
-	-	-	-	-
15	1	1	1	1

Design:

T0 =1, T1=Q0

T2=Q1.Q0, T3=Q2.Q1.Q0

Synchronous Counters

2- Using D F-F

Design:

D0= 1 XOR Q0

D1 = Q1 XOR Q0

D2= Q2 XOR (Q1.Q0)

D3=Q3 XOR (Q2.Q1.Q0)

-Clear the counter:

Make D3 .. D0 =0 0 0 0 using Reset as

D0 = (1 XOR Q0).!Reset

D1= (Q1 XOR Q0). !Reset

D2=................

Parallel Load any Count

D0= ....

D2=(Q2 XOR Q1.Q0)./Load + PD2.Load

Changing the modulo of Counter

Decode the last count and use to load 0000

Example: modulo 6 counter

load=Q2.Q0 then PD=0000, When count=5 load is active

0000

0001

0010

0011

0100

0101 load=1

0000

For BCD counter decode count 9, load =Q3.Q0

Code for 8 BIT Parallel Register with asynchronous Reset

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg8 IS
  PORT( D   : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        Resetn, Clock   :IN  STD_LOGIC ;
        Q   : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END reg8;

ARCHITECTURE Behavior OF reg8 IS
BEGIN
  PROCESS(Resetn, Clock)
  BEGIN
     IF Resetn = '0' THEN
              Q <="00000000";
     ELSEIF Clock'EVENT AND Clock = '1' THEN
              Q<= D;
     END IF;
     END PROCESS;
END Behavior;
```

Code for four BIT UP Counter

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY upcount IS
  PORT( Resetn, Clock, E   :IN  STD_LOGIC ;
        Q   : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END upcount;

ARCHITECTURE Behavior OF upcount IS
   SIGNAL Count: STD_LOGIC_VECTOR( 3 DOWNTO 0);
BEGIN
  PROCESS(Resetn, Clock)
  BEGIN
     IF Resetn = '0' THEN
              Count <="0000";
     ELSEIF (Clock'EVENT AND Clock = '1') THEN
           IF E='1' THEN
              Count<= Count+1;
           ELSE
              Count=Count;
     END IF;
   END PROCESS;
   Q<=Count;
END Behavior;
```

Ch 7 problems

7.5 From 100 MHz clock, generate 50 MHz, 25 MHz, 12.5 MHz

First stage F-F D0=Q0 XOR 1 This is divide by 2 =50 MHz

D1 = Q1 XOR Q0 This is divide by 4 = 25 MHz

D2 = Q2 XOR (Q1.Q0) this is divide by 8 = 12.5 MHz

7-18 Find the sequence of given circuit

T0= 1, T1=Q0, T2=Q1

Q0	Q1	Q2
0	0	0
1	0	0
0	1	0
1	1	1
0	0	0

Code for problem 7-28

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY circuit IS
  PORT( D   : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        Resetn, Clock   :IN  STD_LOGIC ;
        Q   : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END circuit;

ARCHITECTURE Behavior OF reg8 IS
BEGIN
  PROCESS(Resetn, Clock)
  BEGIN
     IF Resetn = ’0’ THEN
             Q <="0000";
     ELSEIF Clock’EVENT AND Clock = ’1’ THEN
             Q<= D+Q;
     END IF;
     END PROCESS;
END Behavior;
```