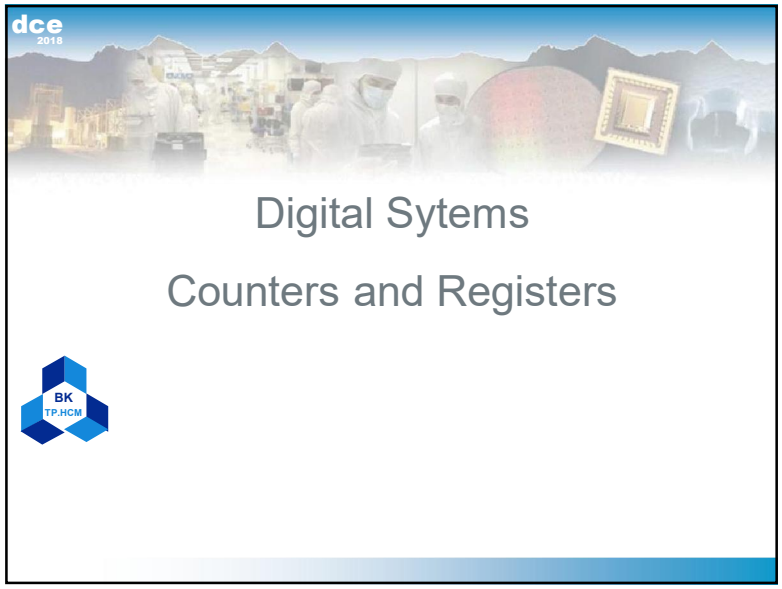


Digital Sytems

Counters and Registers

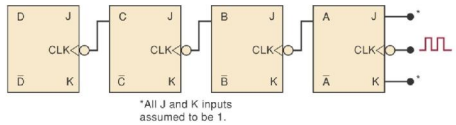
Introduction

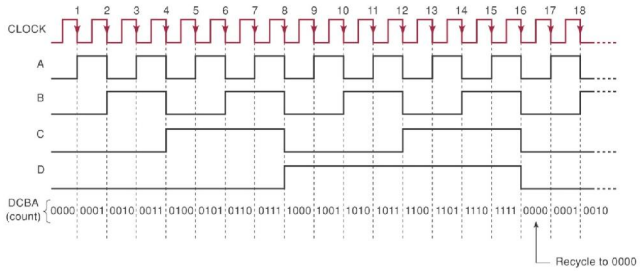
- FFs and logic gates are combined to form various counters and registers.
- Part 1 covers counter principles, various counter circuits, and IC counters.
- Part 2 covers several types of IC registers and shift register counter troubleshooting.

Asynchronous (Ripple) Counters

- Review of four bit counter operation (refer to next slide)
  - Clock is applied only to FF A. J and K are high in all FFs to toggle on every clock pulse.
  - Output of FF A is CLK of FF B and so forth.
  - FF outputs D, C, B, and A are a 4 bit binary number with D as the MSB.
  - After the negative transistion of the 15th clock pulse the counter recycles to 0000.
- This is an asynchronous counter because state is not changed in exact synchronism with the clock.

Four-bit asynchronous (ripple) counter

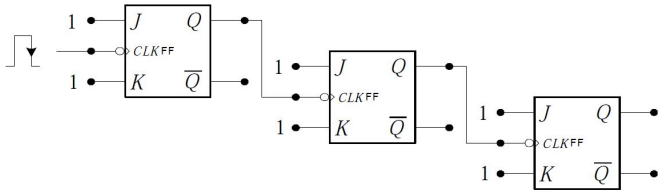




- MOD = the number of states

Frequency division

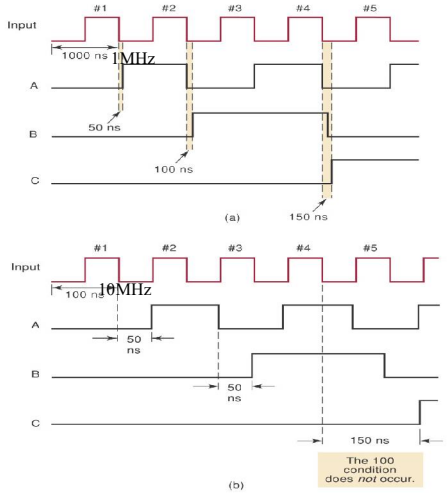
- The output frequency of each FF = the clock frequency of input / 2.
- The output frequency of the last FF = the clock frequency / MOD.



Propagation Delay in Ripple Counters

- Ripple counters are simple, but the cumulative propagation delay can cause problems at high frequencies.
- For proper operation the following apply:
  - $T_{clock} \geq N \times t_{pd}$
  - $F_{max} = 1/(N \times t_{pd})$

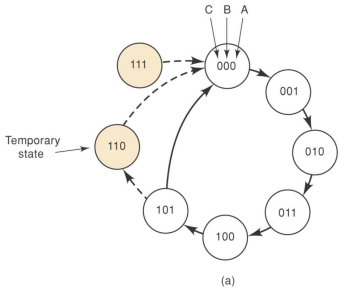
Ripple Counter Propagation Delay



Counters with MOD Number < 2^N

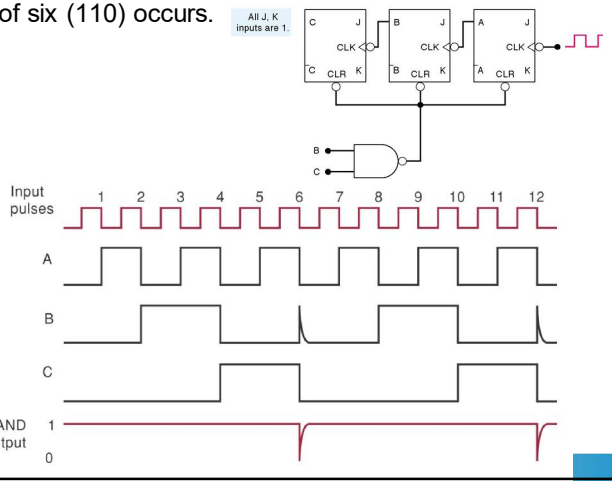
- Find the smallest MOD required so that 2^N is less than or equal to the requirement.
- Connect a NAND gate to the asynchronous CLEAR inputs of all FFs.
- Determine which FFs are HIGH at the desired count and connect the outputs of these FFs to the NAND gate inputs.

State transition diagram for the MOD-6 counter



MOD-6 Counter

MOD-6 counter produced by clearing a MOD-8 counter when a count of six (110) occurs.



9

Counters with MOD Number < 2^N

- General Procedures Counter Design
  1. Find the smallest number of FF
  2. Connect a NAND gate to the Asynchronous CLEAR inputs of all the FFs
  3. Determine which FFs will be in the HIGH state at a count = X; then connect the normal outputs of these FFs to the NAND gate inputs

10

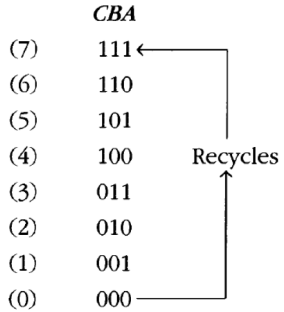
Decade counters/BCD counters

- Decade counters/BCD counters
  - A decade counter is any counter with 10 distinct states, regardless of the sequence. Any MOD-10 counter is a decade counter.
  - A BCD counter is a decade counter that counts from binary 0000 to 1001.
- Decade counters are widely used for counting events and displaying results in decimal form.

11

Asynchronous Down Counter

- All of the counters we have looked were up counters.
- Down counter counts number downward e.g: 111-> 000

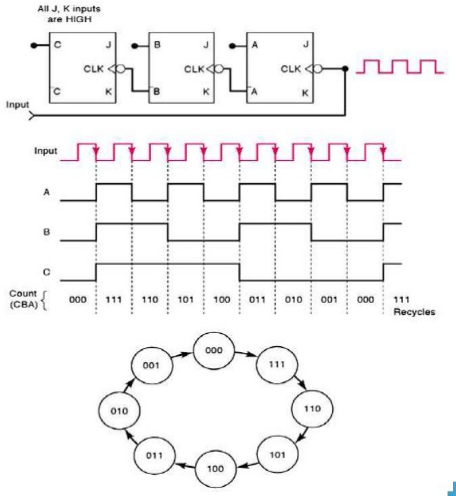


12

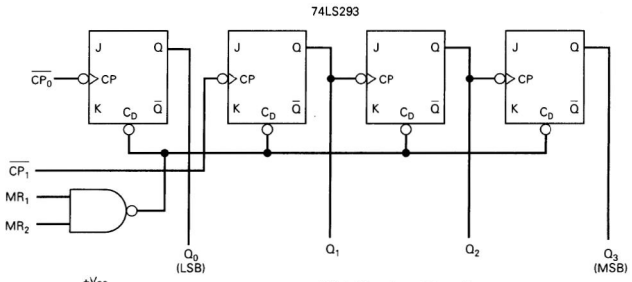
Asynchronous Down Counter

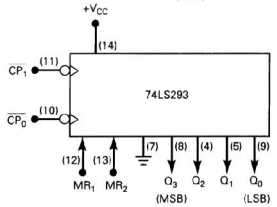
- Each FF, except the first must toggle when the preceding FF goes from LOW to HIGH
- If the FFs have CLK inputs that respond to negative transition (HIGH to LOW), then an inverter can be placed in front of each CLK input; however the same effect can accomplished by driving each FF CLK input from the inverted output of the preceding FF.
- Input pulses are applied to A. The A' output serves as the CLK input for B ; the B' output serves as the CLK input for the C.
- The waveforms at A, B and C show that B toggles whenever A goes LOW to HIGH and C toggles whenever B goes LOW to HIGH.

Asynchronous Down Counter



IC Asynchronous counter





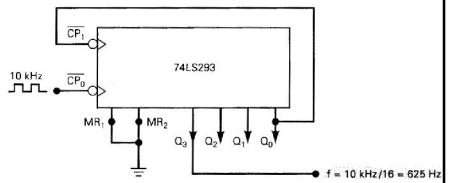
*All J, K inputs are internally connected HIGH.

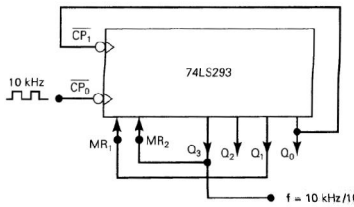
LS293 MODE SELECTION

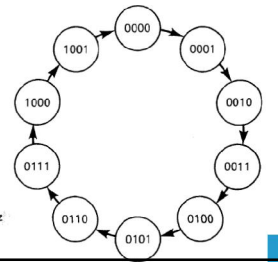
RESET INPUTS		OUTPUTS			
MR_1	MR_2	Q_0	Q_1	Q_2	Q_3
H	H	L	L	L	L
L	H	Count			
H	L	Count			
L	L	Count			

Example

- Show how to wire the 74LS293 as a MOD-16, MOD-10 counter with a 10-kHz clock input. Determine the frequency at Q3.

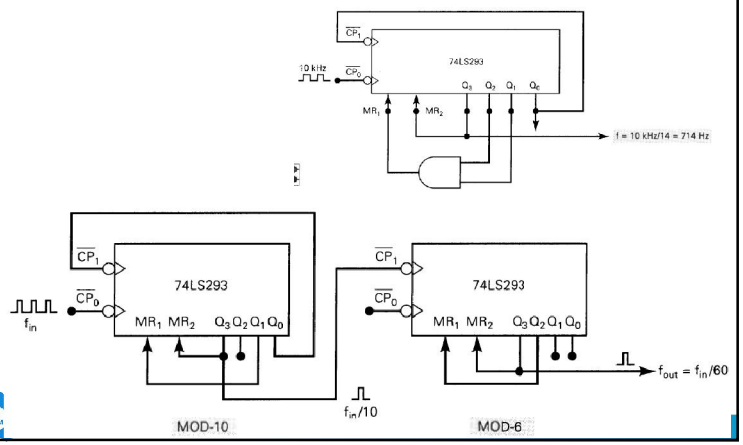






Example

- Show how to wire the 74LS293 as a MOD-14, MOD-60, counter with a 10-kHz clock input.

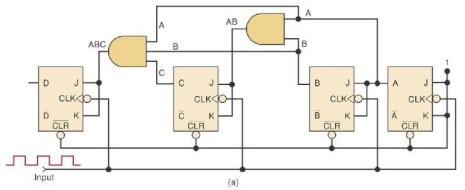


Synchronous (Parallel) Counters

- All FFs are triggered by CLK simultaneously
- Mod-16 counter.
  - Each FF has J and K inputs connected so they are **HIGH only when the outputs of all lower-order FFs are HIGH**.
  - The total propagation delay will be the same for any number of FFs.
- Synchronous counters can operate at much higher frequencies than asynchronous counters.

Count	D	C	B	A
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
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0
.	.	.	.	.
.	.	.	.	.
.	.	etc.	.	.

(b)



Synchronous (Parallel) Counters

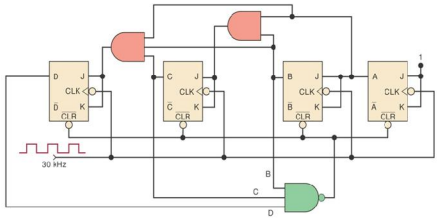
- Circuit Operation
  - On a given NGT of the clock, only those FFs that are supposed to toggle on that NGT should have J=K=1 when that NGT occurs.
  - FF A must change states at each NGT. Its J and K inputs arepermanently HIGH so that it will toggle on each NGT of the CLK input.
  - FF B must change states on each NGT that occurs while A=1.
  - FF C must change states on each NGT that occurs while A=B=1
  - FF D must change states on each NGT that occurs while A=B=C=1

Synchronous (Parallel) Counters

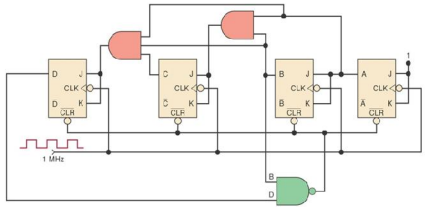
- Each FF should have its J&K inputs connected such that they are HIGH only when the outputs of all lower-order FFs are in the HIGH state.
- Advantages over asynchronous:
  1. FFs will change states simultaneously; synchronized to the NGTs of the input clock pulses.
  2. Propagation delays of the FFs do not add together to produce the overall delay.
  3. he total response time is the time it takes one FF to toggle plus the time for the new logic levels to propagate through a single AND gate to reach the J, K inputs.
- total delay = FF *tpd +AND gate tpd*

Counters for MOD < 2^N

MOD-14 counter resets when count 14 is reached.

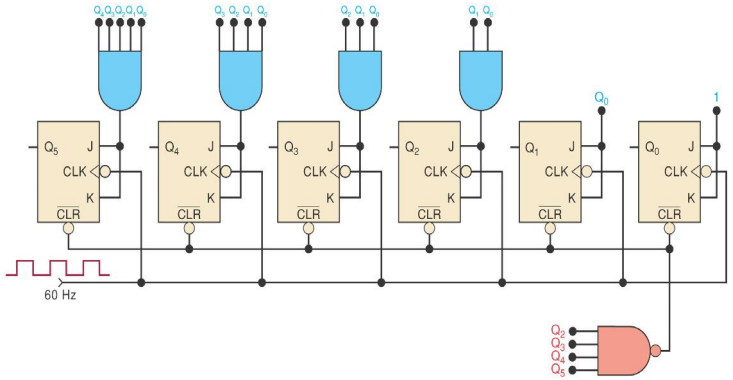


MOD-10 (decade) counter. Resets when count 10 is reached.

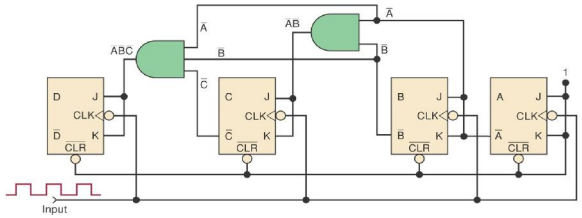


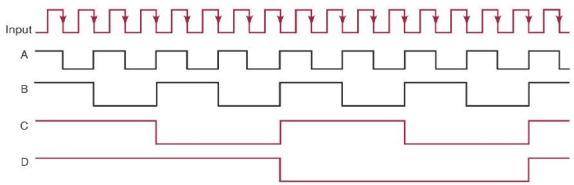
Example: MOD-60 Counter

Resets when count 60 is reached.



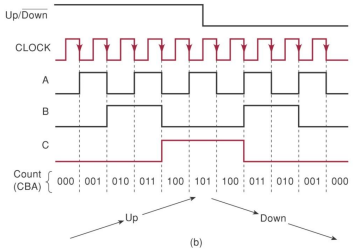
Synchronous, MOD-16, down counter





Synchronous Down and Up/Down Counters

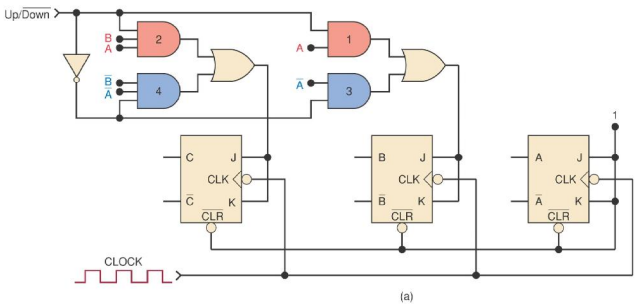
- The synchronous counter can be converted to a down counter by using the inverted FF outputs to drive the JK inputs.



MOD-8 synchronous up/down counter

The counter counts up when the control input Up/Down = 1; it counts down when the control input Up/Down = 0.

MOD-8 synchronous up/down counter



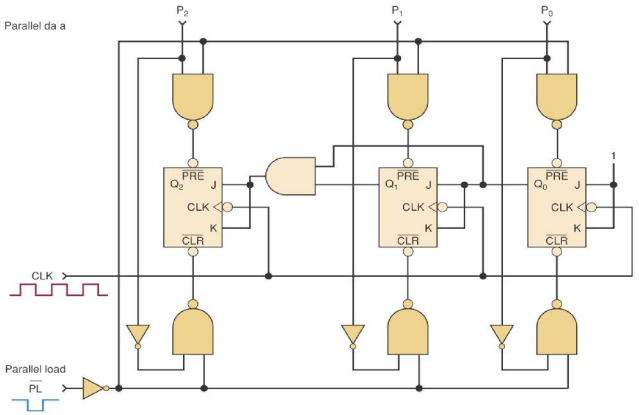
(a)

The counter counts up when the control input Up/Down = 1; A and B signals are passed
it counts down when the control input Up/Down = 0; inverted A and B signals are passed

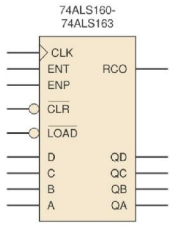
Presettable Counters

- A presettable counter can be set to any desired starting point either asynchronously or synchronously.
- The preset operation is also called parallel loading the counter.

Synchronous counter with asynchronous parallel load



IC Synchronous Counters



(a)

Part Number	Modulus
74ALS160	10
74ALS161	16
74ALS162	10
74ALS163	16

(b)

74ALS160-74ALS163 Function Table

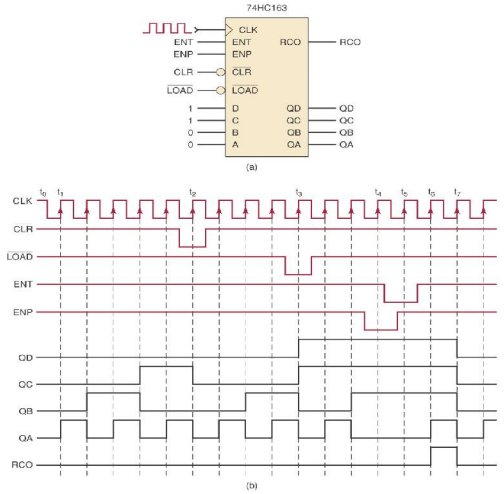
CLR	LOAD	ENP	ENT	CLK	Function	Part Numbers
L	X	X	X	X	Asynch. Clear	74ALS160 & 74ALS161
L	X	X	X	↑	Synchr. Clear	74ALS162 & 74ALS163
H	L	X	X	↑	Synchr. Load	All
H	H	H	H	↑	Count up	All
H	H	L	X	X	No change	All
H	H	X	L	X	No change	All

(c)

- 4 FFs,
- PGT at the CLK input,
- The counter can be preset to any value (applied to the A, B, C, and D inputs) by applying an active-low LOAD input.

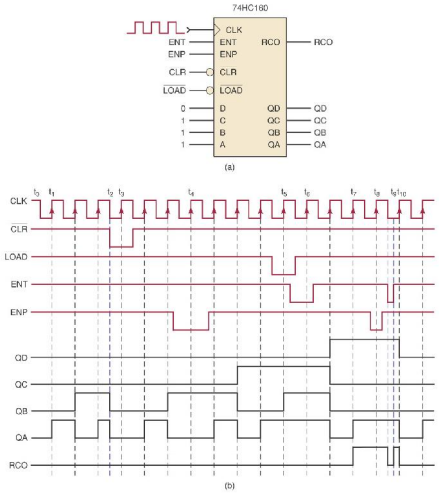
Synchronous Counter Example

- start counting at t1
- synchronous clear at t2
- synchronous load at t3
- stop counting at t4 (ENT low)
- no counting at t5 (ENP low)
- resume counting at t6
- terminal state sets RCO (ripple carry out) high automatic reset at t7

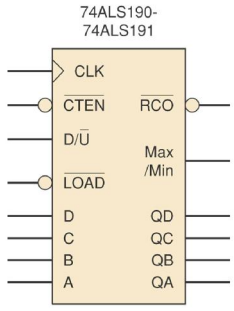


Synchronous Counter Example

- start counting at t1
- asynchronous clear at t2
- asynchronous clear at t3
- stop counting at t4 (ENP low)
- synchronous load at t5
- stop counting at t6 (ENT low)
- continue counting at t7 terminal state of 1001 sets RCO
- stop counting at t8 (ENP)
- RCO goes low at t9 due to low ENT (ENP does not affect RCO)



74ALS190-75ALS191 series synchronous counters (up/down)



Part Number	Modulus
74ALS190	10
74ALS191	16

(b)

74ALS190-74ALS191 Function Table

LOAD	CTEN	D/U	CLK	Function
L	X	X	X	Asynch. Load
H	L	L	↑	Count up
H	L	H	↑	Count down
H	H	X	X	No change

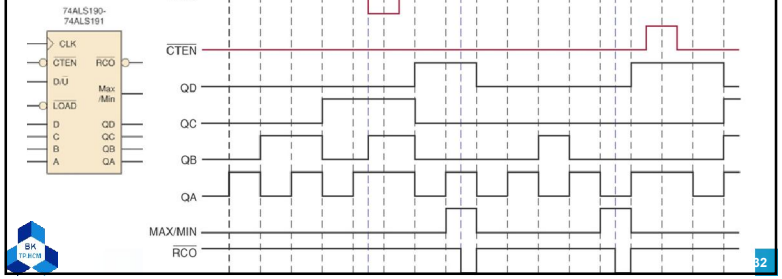
(c)

MOD-10 Counter

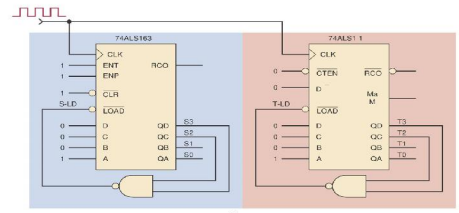
- Maximum state is 1001
- Max/min is high when state is 1001 and up-counting; or 0000 and down-counting
- Max/min low at other times

74ALS190-74ALS191 Function Table

LOAD	CTEN	D/U	CLK	Function
L	X	X	X	Asynch. Load
H	L	L	↑	Count up
H	L	H	↑	Count down
H	H	X	X	No change

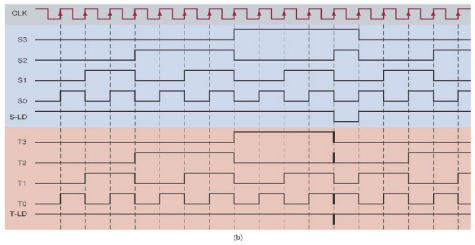


Using 74ALS163 (syn clear) and 74ALS191(async clear) MOD-16 counters for other MODs

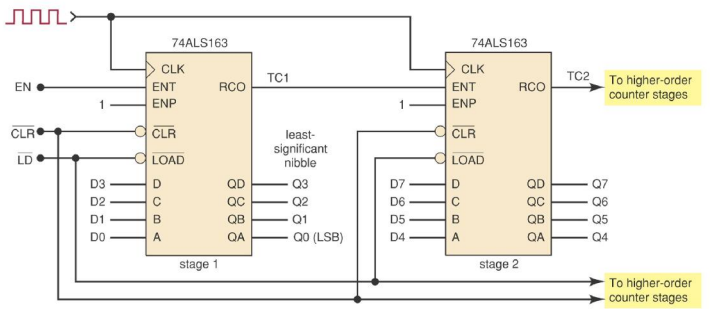


Synchronous load
0001-1100
mod-12 counter

asynchronous load
0001-1011
mod-11 counter (in 1100
state for a short period of time



Extending Maximum Counting Range

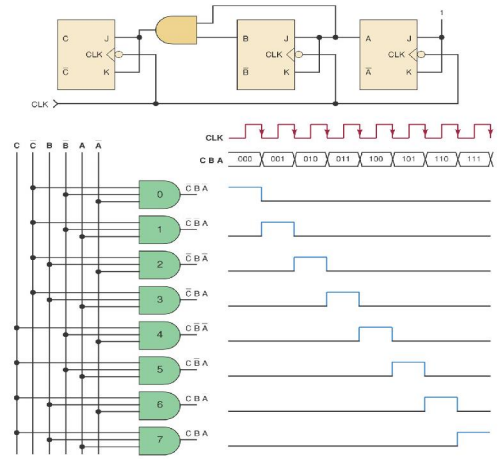


Decoding a Counter

- Decoding is the conversion of a binary output to a decimal value.
- The active high decoder could be used to light an LED representing each decimal number 0 to 7.
- Active low decoding is obtained by replacing the AND gates with NAND gates.

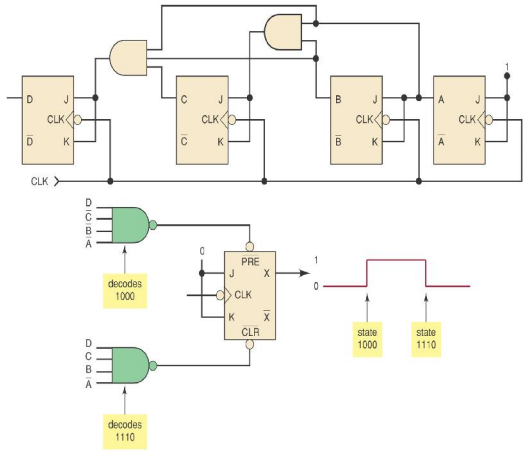
Decoding a Counter

Using AND Gates to Decode a MOD-8 Counter (produce pulse at specific count)



Decoding a Counter

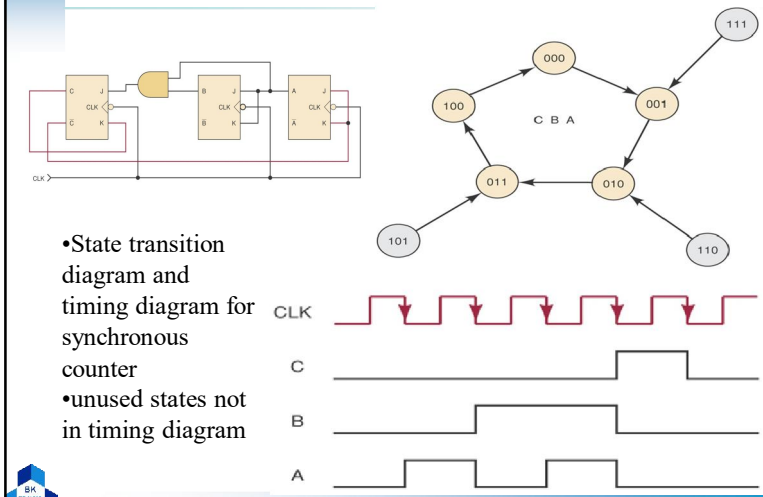
Circuit to Make X High Between Counts of 8 and 14 (sets FF at count 8, then clears at count 14)



Analyzing Synchronous Counters

- Example of a synchronous up counter.
  - The control inputs are as follows: $J_C = A \cdot B$, $K_C = C$, $J_B = K_B = A$, $J_A = K_A = \overline{C}$

Analyzing Synchronous Counters



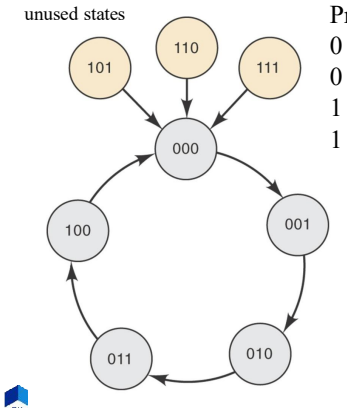
- State transition diagram and timing diagram for synchronous counter
- unused states not in timing diagram

Synchronous Counter Design

- Determine desired number of bits and desired counting sequence
- Draw the state transition diagram showing all possible states
- Use the diagram to create a table listing all PRESENT states and their NEXT states
- Add a column for each JK input (or other inputs). Indicate the level required at each J and K in order to produce transition to the NEXT state.
- Design the logic circuits to generate levels required at each JK input.
- Implement the final expressions.

Choose a type of FF – JK in this example

State transition diagram for the synchronous counter design



Present State	Next State	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

State table of counter example

JK Flip-Flop excitation table

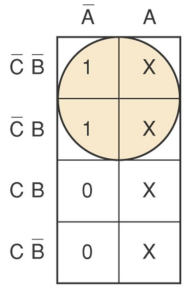
Present State	Next State	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

	PRESENT State C	B	A	NEXT State C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
Line 1	0	0	0	0	0	1	0	x	0	x	1	x
2	0	0	1	0	1	0	0	x	1	x	x	1
3	0	1	0	0	1	1	0	x	x	0	1	x
4	0	1	1	1	0	0	1	x	x	1	x	1
5	1	0	0	0	0	0	x	1	0	x	0	x
6	1	0	1	0	0	0	x	1	0	x	x	1
7	1	1	0	0	0	0	x	1	x	1	0	x
8	1	1	1	0	0	0	x	1	x	1	x	1

K maps for the *J* and *K* logic circuits

K map used to obtain the simplified expression for J_A; from the state table

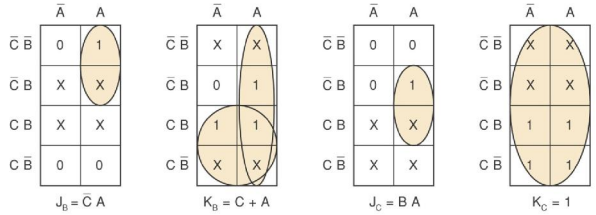
	PRESENT State C	B	A	NEXT State C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
Line 1	0	0	0	0	0	1	0	x	0	x	1	x
2	0	0	1	0	1	0	0	x	1	x	x	1
3	0	1	0	0	1	1	0	x	x	0	1	x
4	0	1	1	1	0	0	1	x	x	1	x	1
5	1	0	0	0	0	0	x	1	0	x	0	x
6	1	0	1	0	0	0	x	1	0	x	x	1
7	1	1	0	0	0	0	x	1	x	1	0	x
8	1	1	1	0	0	0	x	1	x	1	x	1



$J_A = \bar{C}$

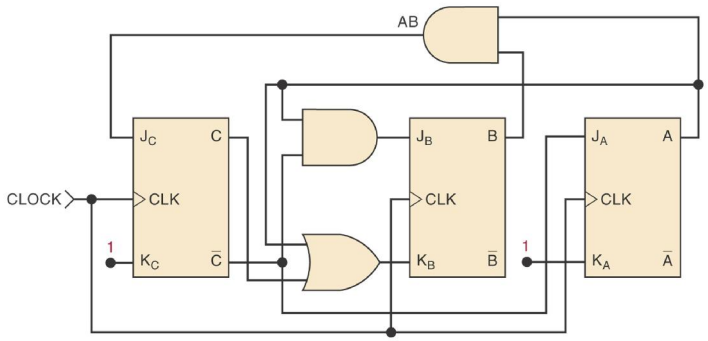
(b)

K maps for the *J* and *K* logic circuits



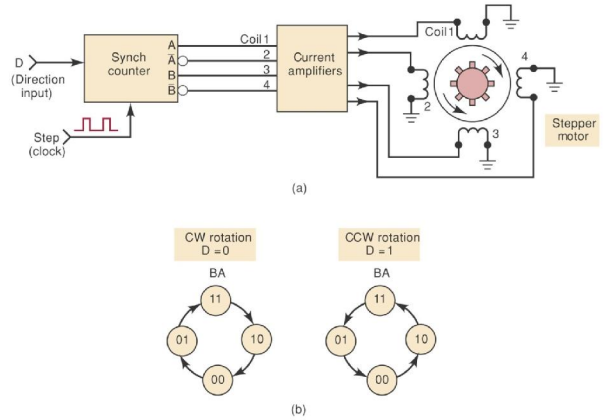
	PRESENT State C	B	A	NEXT State C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
Line 1	0	0	0	0	0	1	0	x	0	x	1	x
2	0	0	1	0	1	0	0	x	1	x	x	1
3	0	1	0	0	1	1	0	x	x	0	1	x
4	0	1	1	1	0	0	1	x	x	1	x	1
5	1	0	0	0	0	0	x	1	0	x	0	x
6	1	0	1	0	0	0	x	1	0	x	x	1
7	1	1	0	0	0	0	x	1	x	1	0	x
8	1	1	1	0	0	0	x	1	x	1	x	1

Final implementation of the synchronous counter design example

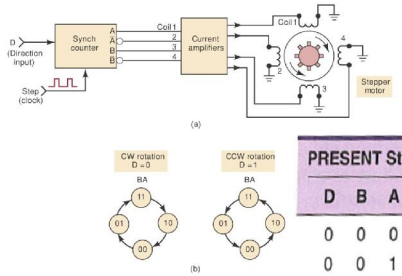


Design example

(a) A synchronous supplies the appropriate sequential outputs to drive a stepper motor; (b) state transition diagrams for both states of Direction input, *D*.

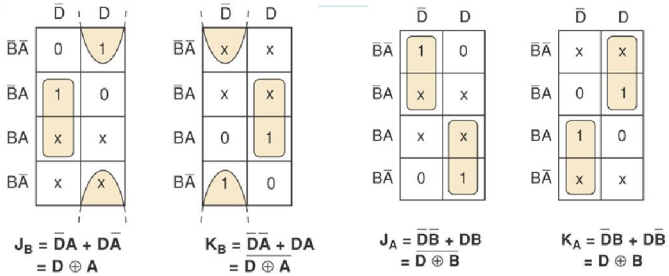


State table for Design Example



PRESENT State			NEXT State		Control Inputs			
D	B	A	B	A	J_B	K_B	J_A	K_A
0	0	0	0	1	0	x	1	x
0	0	1	1	1	1	x	x	0
0	1	0	0	0	x	1	0	x
0	1	1	1	0	x	0	x	1
1	0	0	1	0	1	x	0	x
1	0	1	0	0	0	x	x	1
1	1	0	1	1	x	0	1	x
1	1	1	0	1	x	1	x	0

K–maps for four outputs

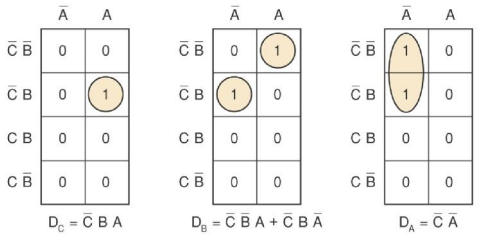


PRESENT State			NEXT State		Control Inputs			
D	B	A	B	A	J_B	K_B	J_A	K_A
0	0	0	0	1	0	x	1	x
0	0	1	1	1	1	x	x	0
0	1	0	0	0	x	1	0	x
0	1	1	1	0	x	0	x	1
1	0	0	1	0	1	x	0	x
1	0	1	0	0	0	x	x	1
1	1	0	1	1	x	0	1	x
1	1	1	0	1	x	1	x	0

State Table for Example: MOD-5 Counter Using D-type Flip-Flops

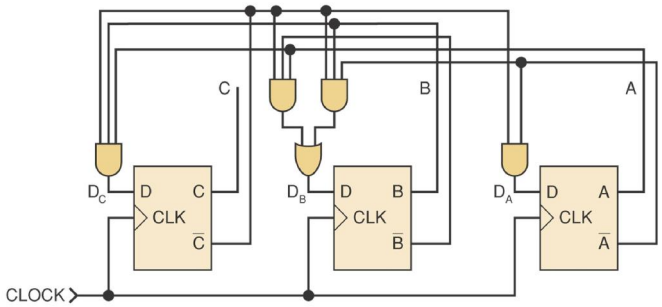
PRESENT State			NEXT State			Control Inputs		
C	B	A	C	B	A	D_C	D_B	D_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

K maps for Outputs - MOD-5 D-flip-flop counter



PRESENT State			NEXT State			Control Inputs		
C	B	A	C	B	A	D_C	D_B	D_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

Implementation of MOD-5, D flip-flop design

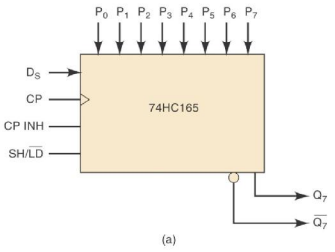


Integrated-Circuit Registers

- Registers can be classified by the way data is entered for storage, and by the way data is outputted from the register.
  - Parallel in/parallel out (PIPO)
  - Serial in/serial out (SISO)
  - Parallel in/serial out (PISO)
  - Serial in/parallel out (SIPO)

PISO – The 74ALS165/74HC165

- 8 bit register
  - Serial data entry via D_S
  - Asynchronous parallel data entry P_0 through P_7
  - Only the outputs of Q_7 are accessible
- CP is clock input for shifting
- Clock inhibit input
- Shift load input



Function Table

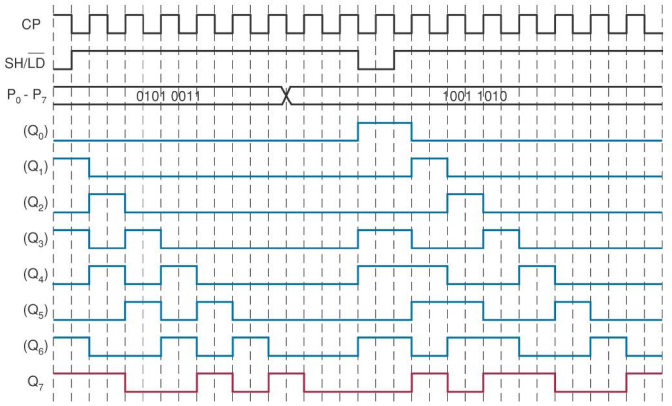
Inputs			Operation
SH/$\overline{LD}$	CP	CP INH	
L	X	X	Parallel load
H	H	X	No change
H	X	H	No change
H	↓	L	Shifting
H	L	↓	Shifting

H = high level
L = low level
X = immaterial
↓ = PGT

(b)

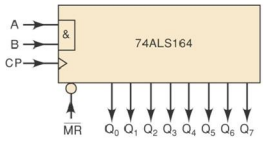
74HC165 PISO Waveforms

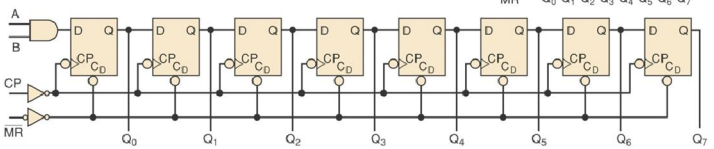
Ds = 0, CP INH = 0, Output values for given inputs (P0=P7)



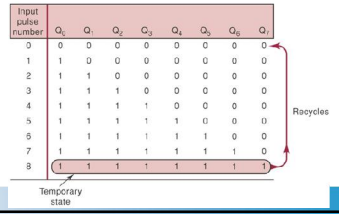
SIPO – The 74ALS164/74HC164

- 8 bit shift register
- Each FF output is externally accessible
- A and B inputs are combined in an AND gate for serial input.
- Shift occurs on NGT of the clock input.





(a)



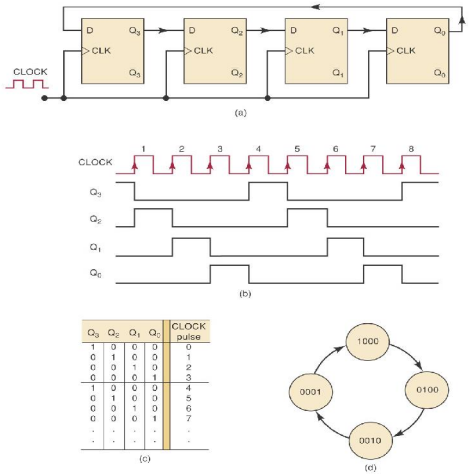
Other similar devices

- 74194/ASL194/HC194
  - 4 bit bi-directional universal shift register
  - Performs shift left, shift right, parallel in and parallel out.
- 74373/ALS373/HC373/HCT373
  - 8 bit PIPO with 8 D latches
  - Tristate outputs
- 74374/ALS374/HC374
  - 8 bit PIPO with 8 edge triggered D FFs, Tristate outputs

Shift Register Counters

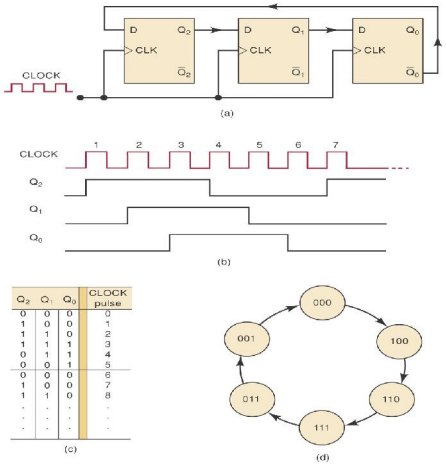
- Ring Counter
  - Last FF shifts its value to first FF
  - Uses D-type FFs (JK FFs can also be used)
  - Must start with only one FF in the 1 state and all others in the 0 state.

Four-bit Ring Counter



MOD-6 Johnson counter

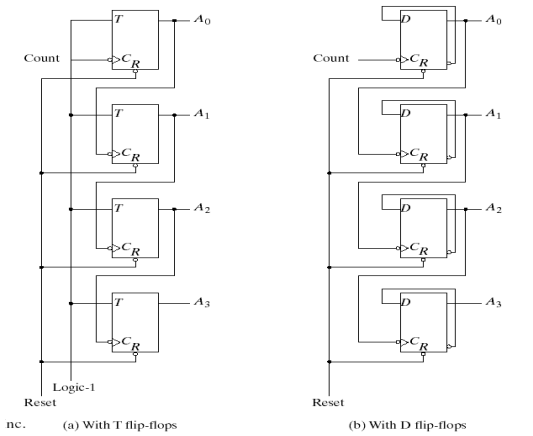
- Johnson counter
  - Also called a twisted ring counter
  - Same as ring counter but the inverted output of the last FF is connected to input of the first FF



HDL for Registers and Counters

- Registers and counters can be described in HDL at either the behavioral or the structural level.
- A structural level description shows the circuit in terms of a collection of components such as gates, flip-flops and multiplexers.
- In the behavioral, the register is specified by a description of the various operations that it performs similar to a function table.
- The various components are instantiated to form a hierarchical description of the design similar to a representation of a logic diagram.

HDL for Registers and Counters (1)



nc. (a) With T flip-flops (b) With D flip-flops

4-bit Binary Ripple Counter

HDL for Registers and Counters (2)

```
//Ripple counter
module ripplecounter(A0,A1,A2,A3,Count,Reset);
  output A0,A1,A2,A3;
  input Count,Reset;
//Instantiate complementing flip-flop
  CF F0 (A0,Count,Reset);
  CF F1 (A1,A0,Reset);
  CF F2 (A2,A1,Reset);
  CF F3 (A3,A2,Reset);
endmodule
```

```
//Complementing flip-flop
//Input to D flip-flop = Q'
module CF (Q,CLK,Reset);
  output Q;
  input CLK,Reset;
  reg Q;
always@(negedge CLK or posedge Reset)
 if(Reset) Q<=1'b0;
 else  Q<= (~Q);
endmodule
```


HDL for Registers and Counters (3)

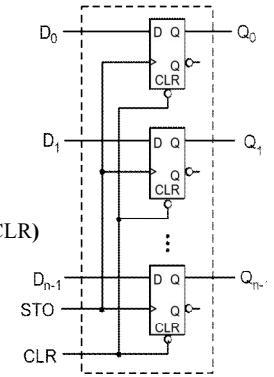
```
//Behavioral description of mod-n up counters
module upmodn (Ck, Q);
    parameter n = 6;
    input Ck;
    output [3:0] Q;
    reg [3:0] Q;

    always @(posedge Ck)
        if (Q == n)
            Q <= 0;
        else
            Q <= Q + 1;
endmodule
```

```
//Behavioral description of mod-n downcounters
module dwnmodn (Ck, Q);
    parameter n = 5;
    input Ck;
    output [3:0] Q;
    reg [3:0] Q;
    always @(posedge Ck)
        if (Q == 0)
            Q <= n;
        else
            Q <= Q -1;
endmodule
```


HDL for Registers and Counters (4)

```
//Parallel Registers
module reg1 (STO, CLR, D, Q);
    parameter n = 16;
    input STO, CLR;
    input [ n-1:0] D;
    output [ n-1:0] Q;
    reg [n-1:0] Q;
    always @(posedge STO or negedge CLR)
        if ( CLR ==0) Q <= 0;
        else Q <= D;
endmodule
```



HDL for Registers and Counters (5)

```
//Shift Registers
module shift4 (D, LD, LI, Ck, Q);
    input [3:0] D;
    input LD, LI, Ck;
    output [3:0] Q;
    reg [3:0] Q;
    always @(posedge Ck)
        if (LD)
          Q <= D;
        else
        begin
            Q[0] <= Q[1];
            Q[1] <= Q[2];
            Q[2] <= Q[3];
            Q[3] <= LI;
        end
endmodule
```