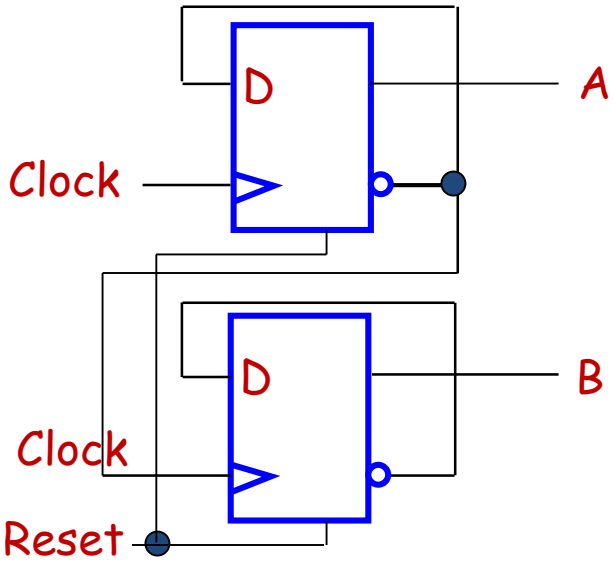
Counters

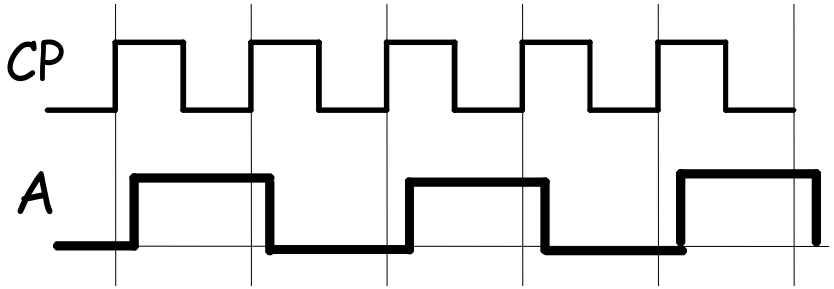
- ✓ Counters are sequential circuits which "count" through a specific state sequence.
  - They can count up, count down, or count through other fixed sequences.
- ✓ Two distinct types are in common usage:
  - Ripple Counters
    - Clock connected to the flip-flop clock input on the LSB bit flip-flop
    - For all other bits, a flip-flop output is connected to the clock input, thus circuit is not truly synchronous!
    - Output change is delayed more for each bit toward the MSB.
    - Resurgent because of low power consumption
  - Synchronous Counters
    - Clock is directly connected to the flip-flop clock inputs
    - Logic is used to implement the desired state sequencing

Ripple Counter

- ✓ How does it work?
  - When there is a positive edge on the clock input of A, A complements
  - The clock input for flip-flop B is the complemented output of flip-flop A

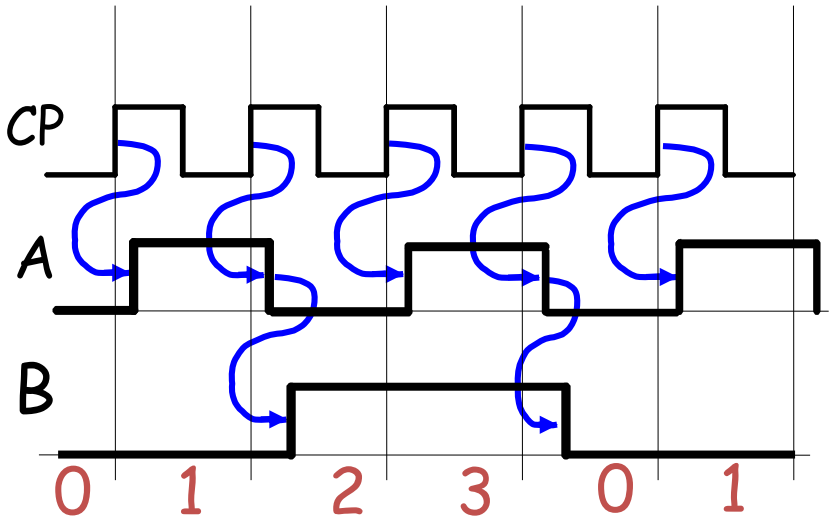


- When flip A changes from 1 to 0, there is a positive edge on the clock input of B causing B to complement



Ripple Counter (continued)

- ✓ The arrows show the cause-effect relation-ship from the prior slide
- ✓ The corresponding sequence of states ⇨



(B,A) = (0,0), (0,1), (1,0), (1,1), (0,0), (0,1), …

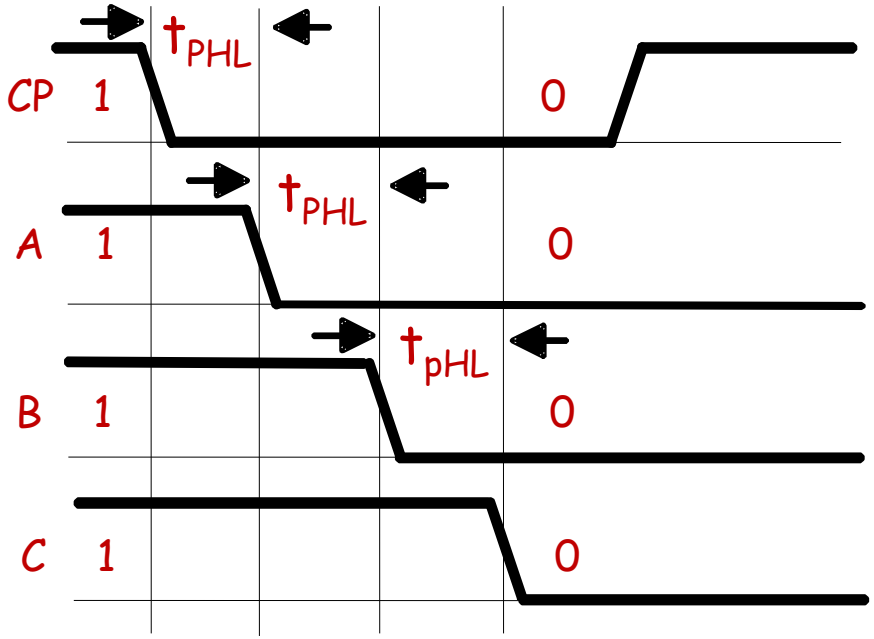
- ✓ Each additional bit, C, D, …behaves like bit B, changing half as frequently as the bit before it.
- ✓ For 3 bits: (C,B,A) = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), (0,0,0), …

Ripple Counter (continued)

- ✓ These circuits are called ripple counters because each edge sensitive transition (positive in the example) causes a change in the next flip-flop's **state**.
- ✓ The changes ripple upward through the chain of flip-flops, i. e., each transition occurs after a clock-to-output **delay** from the stage before.

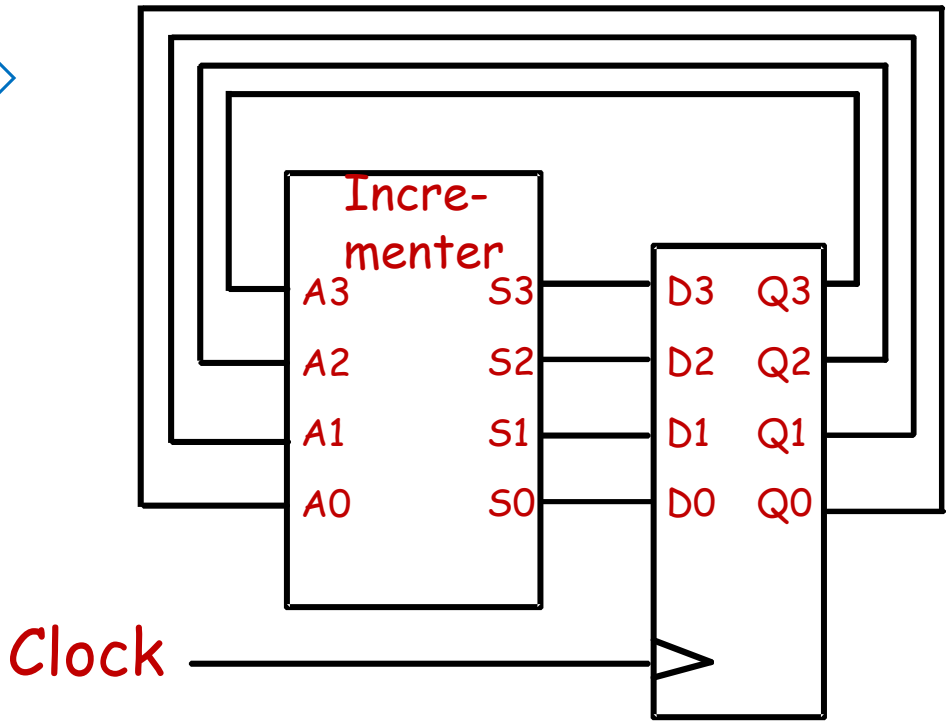
Ripple Counter (continued)

- ✓ Starting with C = B = A = 1, equivalent to (C,B,A) = 7 base 10, the next clock increments the count to (C,B,A) = 0 base 10. In fine timing detail:
  - The clock to output delay t_{PHL} causes an increasing delay from clock edge for each stage transition.
  - Thus, the count "ripples" from least to most significant bit.
  - For *n* bits, total worst case delay is *n* t_{PHL}.



Synchronous Counters

- ✓ To eliminate the "ripple" effects, use a common clock for each flip-flop and a combinational circuit to generate the next state.
- ✓ For an up-counter, use an incrementer ⇨



Other Counters

- ✓ Counters:
  - Down Counter - counts downward instead of upward
  - Up-Down Counter - counts up or down depending on value a control input such as Up/$\overline{\text{Down}}$
  - Parallel Load Counter - Has parallel load of values available depending on control input such as Load
- ✓ Divide-by-n (Modulo n) Counter
  - Count is remainder of division by *n*; *n* may not be a power of 2
  - Count is arbitrary sequence of *n* states specifically designed state-by-state
  - Includes modulo 10 which is the BCD counter

Design Example: Synchronous BCD

- ✓ Use the sequential logic model to design a synchronous BCD counter with D flip-flops
- ✓ Input combinations 1010 through 1111 are don't cares

Current State				Next State			
Q8	Q4	Q2	Q1	Q8	Q4	Q2	Q1
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0

Synchronous BCD (continued)

- ✓ Use K-Maps to two-level optimize the next state equations:

$$D1 = \overline{Q1}$$
$$D2 = \overline{Q8}\,\overline{Q2}\,Q1 + Q2\,\overline{Q1}$$
$$D4 = \overline{Q4}\,Q2\,Q1 + Q4\,\overline{Q2} + Q4\,\overline{Q1}$$
$$D8 = Q8\,\overline{Q1} + Q4\,Q2\,Q1$$

- ✓ The logic diagram can be draw from these equations
  - An asynchronous or synchronous reset should be added
- ✓ What happens if the counter is perturbed by a power disturbance or other interference and it enters a state other than 0000 through 1001?

Synchronous BCD (continued)

- ✓ Find the actual values of the six next states for the don't care combinations from the equations
- ✓ Find the overall state diagram to assess behavior for the don't care states (states in decimal)

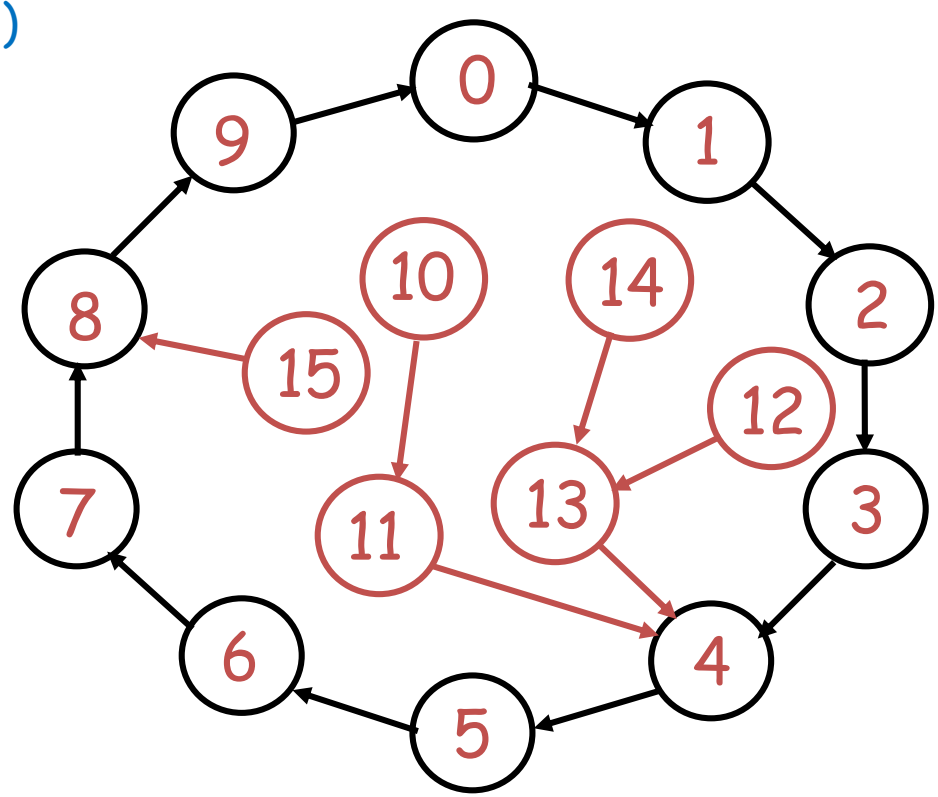
$D1 = \overline{Q1}$

$D2 = \overline{Q8}\,\overline{Q2}Q1 + Q2\overline{Q1}$

$D4 = \overline{Q4}\,\overline{Q2}Q1 + Q4\overline{Q2} + Q4\overline{Q1}$

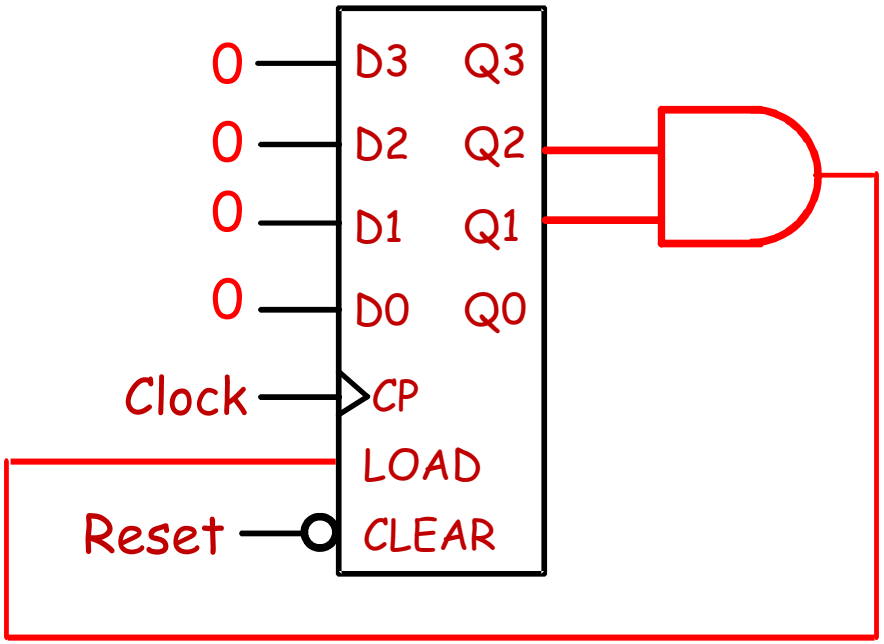
$D8 = Q8\overline{Q1} + Q4Q2Q1$

Present State	Next State
Q8 Q4 Q2 Q1	Q8 Q4 Q2 Q1
1 0 1 0	1 0 1 1
1 0 1 1	0 1 0 0
1 1 0 0	1 1 0 1
1 1 0 1	0 1 0 0
1 1 1 0	1 1 0 1
1 1 1 1	1 0 0 0



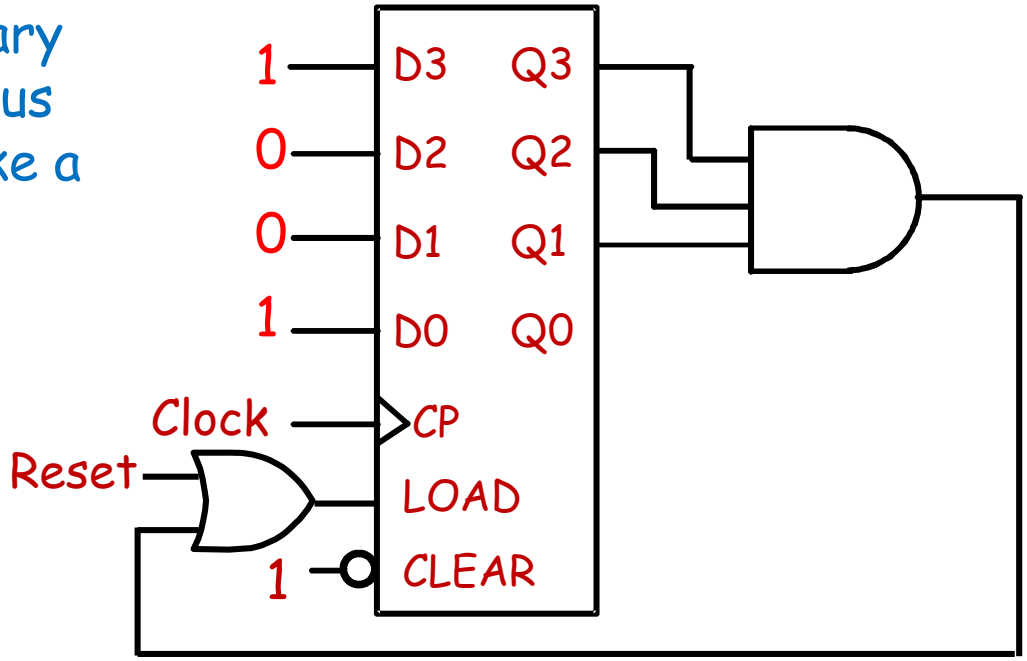
Counting Modulo 7: Synchronously Load on Terminal Count of 6

- A synchronous 4-bit binary counter with a synchronous load and an asynchronous clear is used to make a Modulo 7 counter
- Use the Load feature to detect the count "6" and load in "zero". This gives a count of 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, 3, 4, 5, 6, 0, ...
- Using don't cares for states above 0110



Counting Modulo 6: Synchronously Preset 9 on Reset and Load 9 on Terminal Count 14

- A synchronous, 4-bit binary counter with a synchronous Load is to be used to make a Modulo 6 counter.
- Use the Load feature to preset the count to 9 on Reset and detection of count 14.



- This gives a count of 9, 10, 11, 12, 13, 14, 9, 10, 11, 12, 13, 14, 9, …
- If the terminal count is 15 detection is usually built in as Carry Out (CO)

Example 4: Design a modulo-8 up-counter which counts in the way specified below, use J-K FF

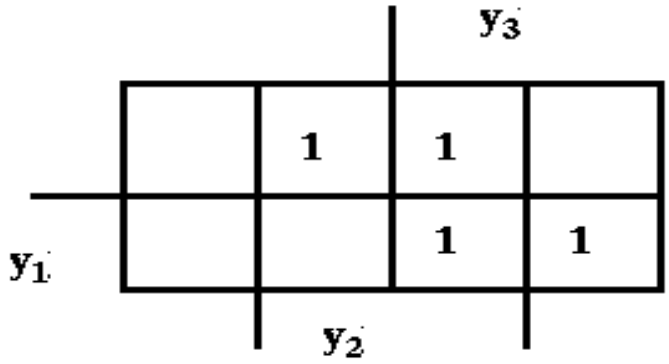
Decimal	Gray
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

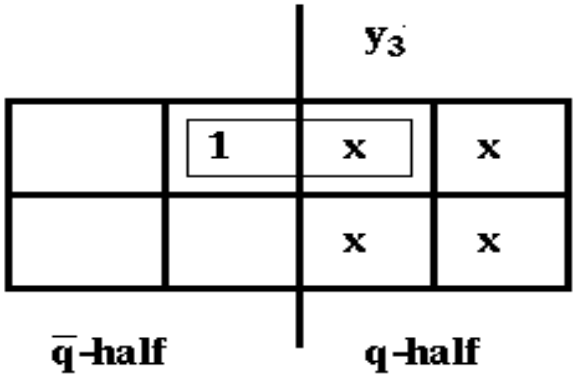
Example 4: TRUTH TABLE

present state			next state		
y_3	y_2	y_1	y_{3+}	y_{2+}	y_{1+}
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	1

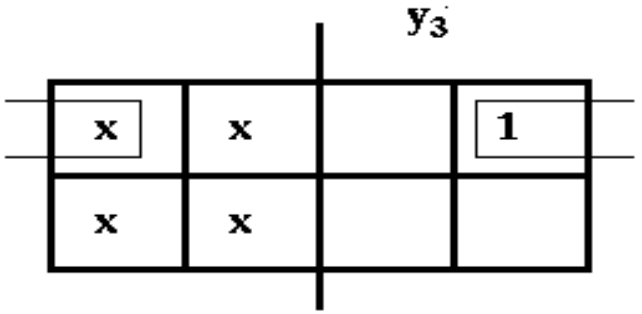
Example 4: Gray code counter

Y3





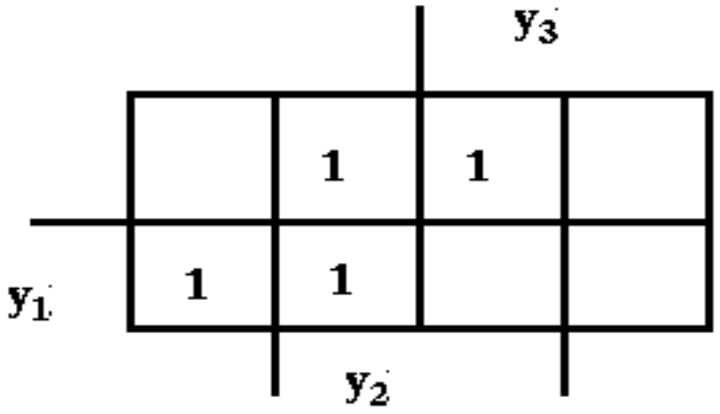
$J_{y3} = y_2\bar{y}_1$

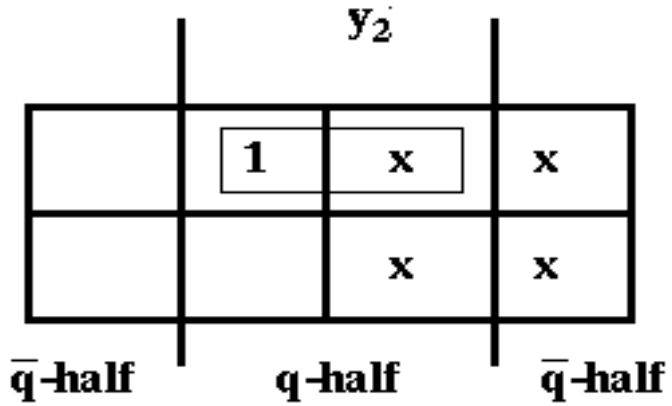


$K_{y3} = \bar{y}_2 y_1$

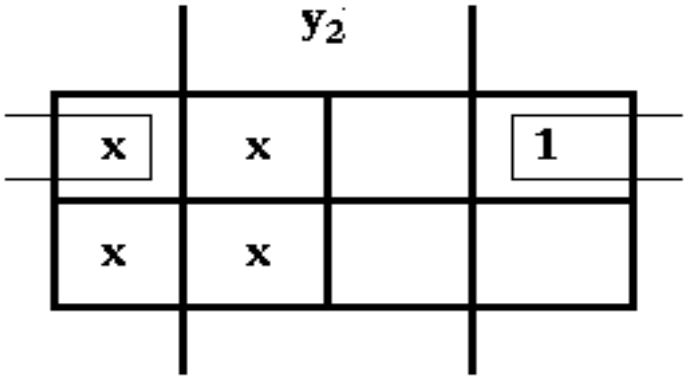
Example 4: Gray code counter

Y2





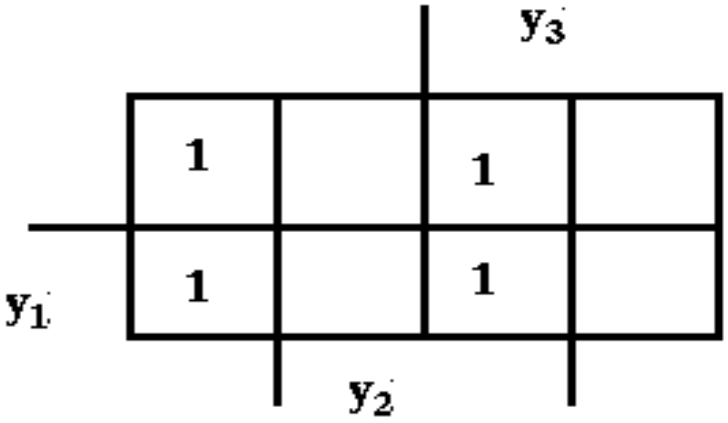
$J_{y2} = \bar{y}_3 y_1$

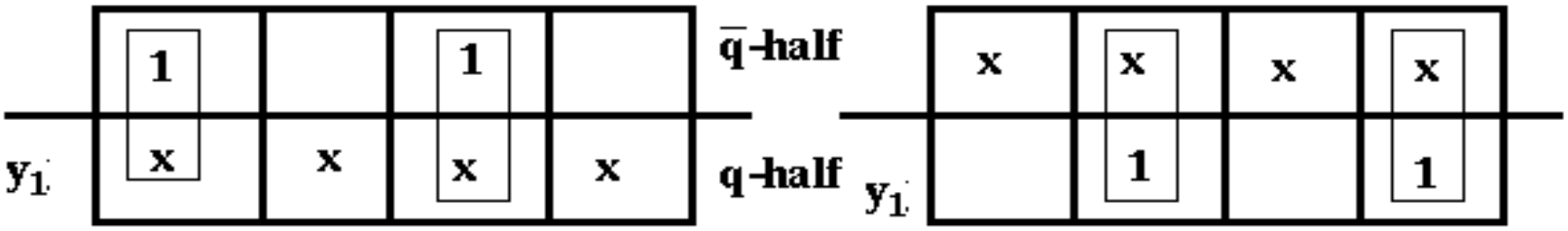


$K_{y2} = y_3 y_1$

Example 4: Gray code counter

Y1





$$J_{y1} = \bar{y}_3\bar{y}_2 + y_3y_2$$

$$K_{y1} = \bar{y}_2y_3 + y_2\bar{y}_3$$