Introduction to Digital Logic

Assist. Prof. Hamza Osman İLHAN

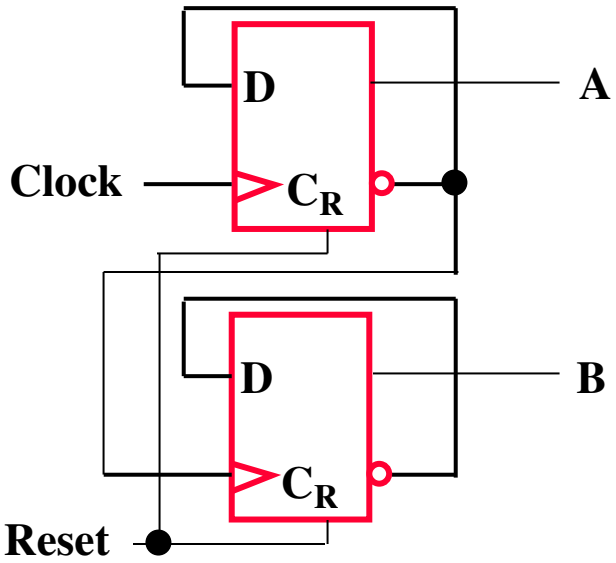
hoilhan@yildiz.edu.tr

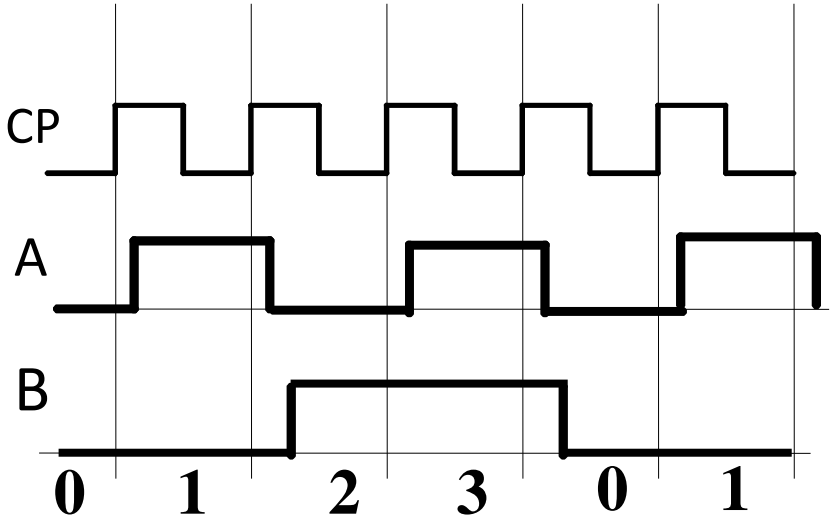
Counters

- 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 is 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

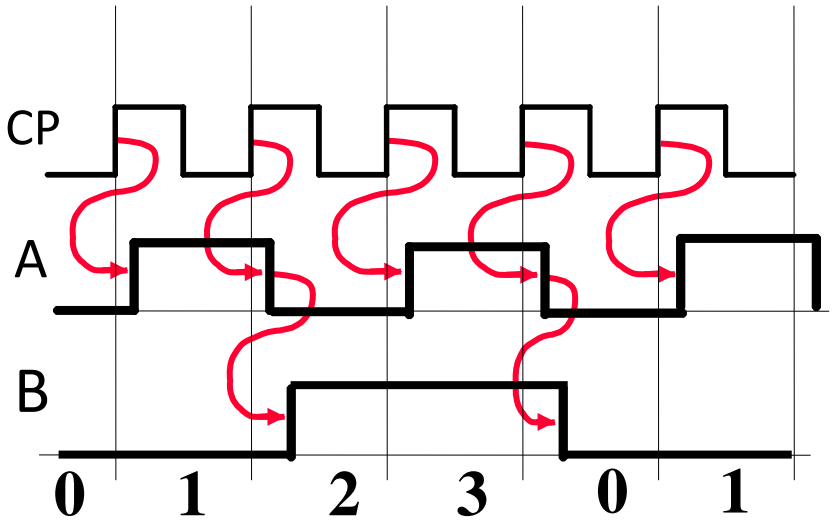
- How does it work?
  - When there is a positive edge on the clock input of A, A complements
  - The clock input for flip-flop B is the complemented output of flip-flop A
  - When flip A changes from 1 to 0, there is a positive edge on the clock input of B causing B to complement





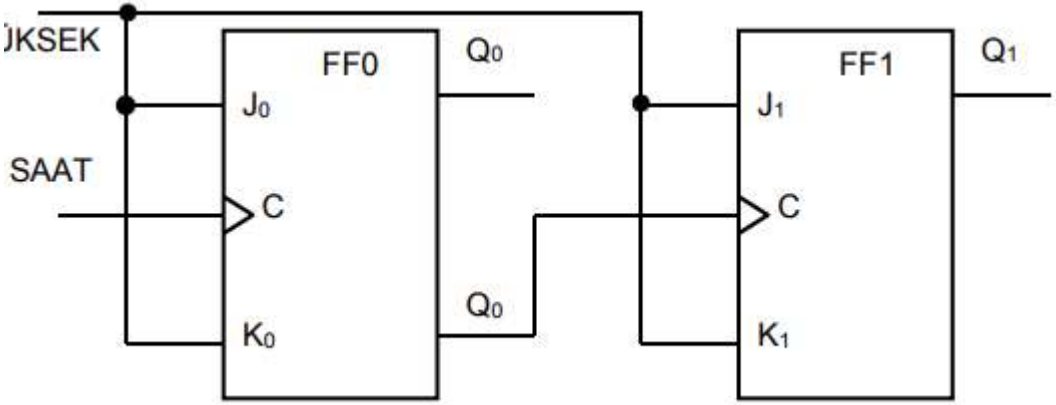
Ripple Counter (continued)

- The arrows show the cause-effect relation-ship from the prior slide =>

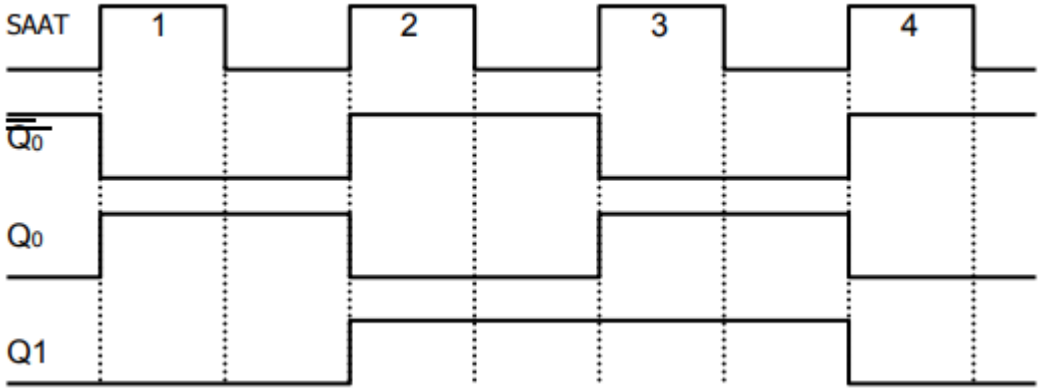


- The corresponding sequence of states => (B,A) = (0,0), **(0,1), (1,0), (1,1), (0,0), (0,1), …**
- Each additional bit, C, D, …behaves like bit B, changing half as frequently as the bit before it.
- For 3 bits: (C,B,A) = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), (0,0,0), …

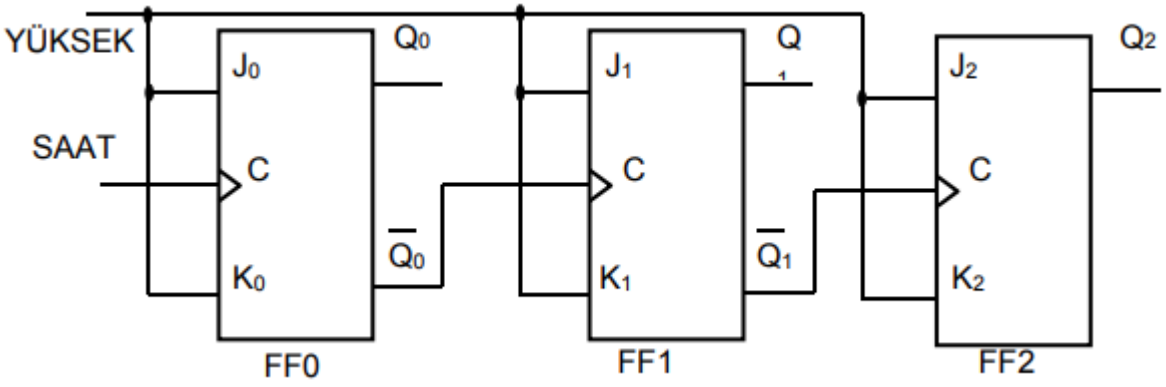
Ripple Counter (continued)



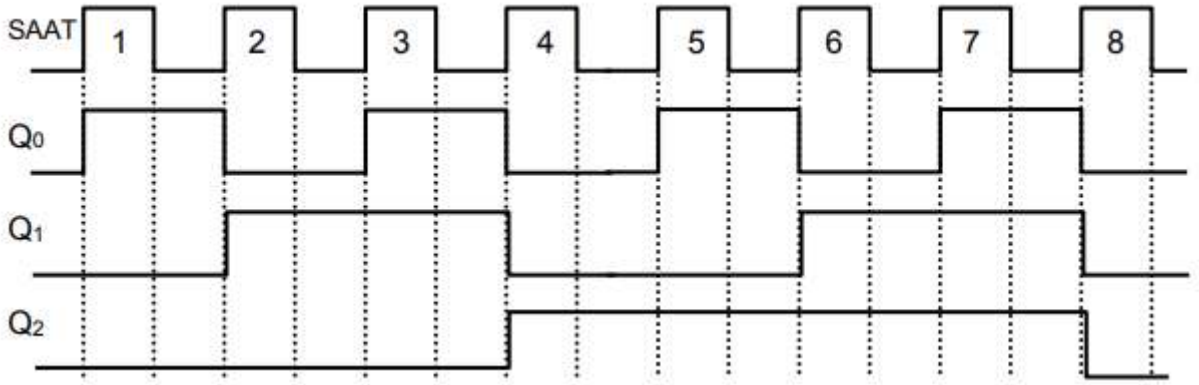
SAAT VURUSU	Q_1	Q_0
Başlangıç	0	0
1	0	1
2	1	0
3	1	1
4	0	0



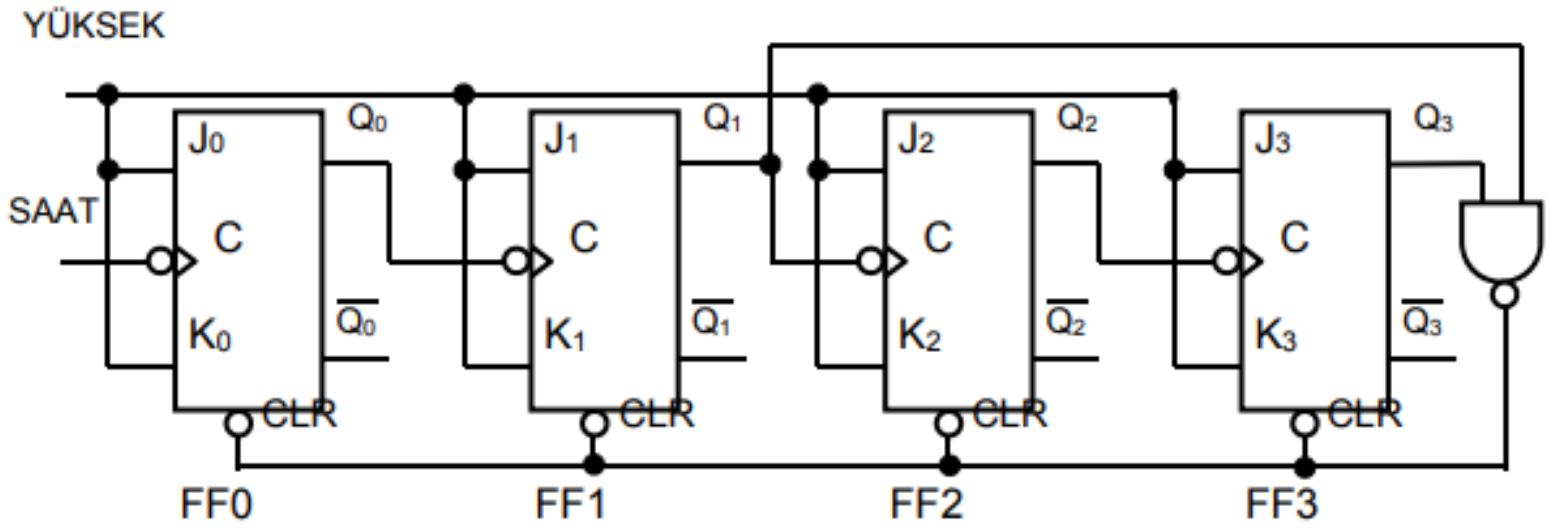
Ripple Counter (continued)



SAAT VURUSU	Q_2	Q_1	Q_0
Başlangıç	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

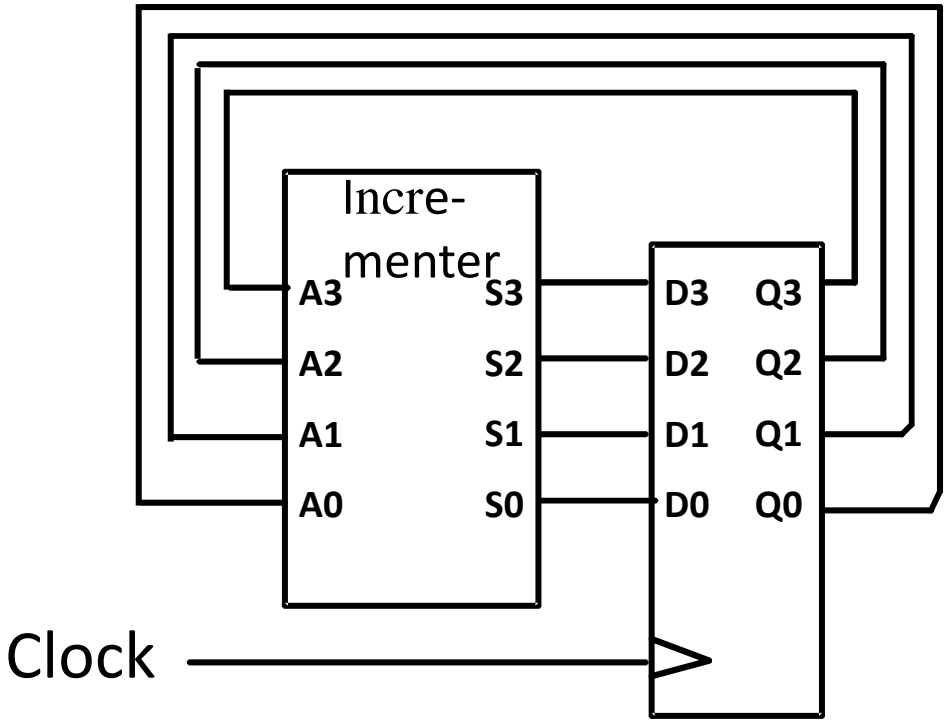


Ripple Counter (continued)

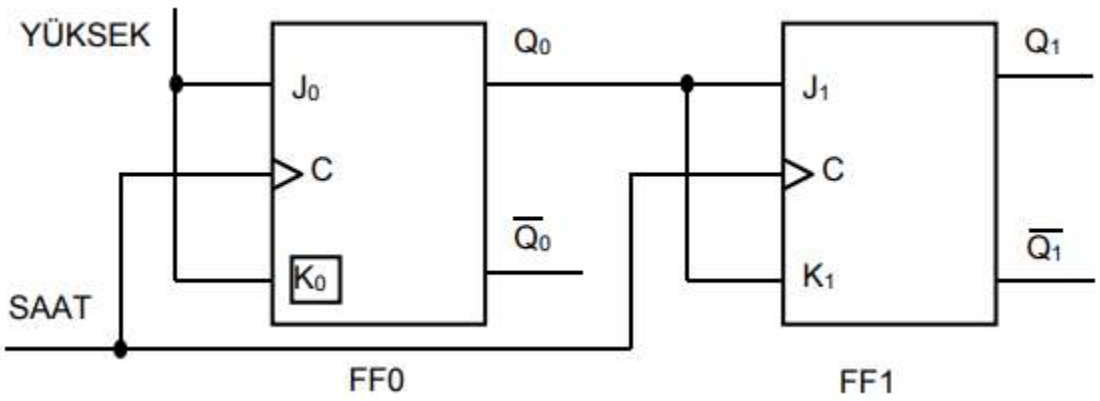


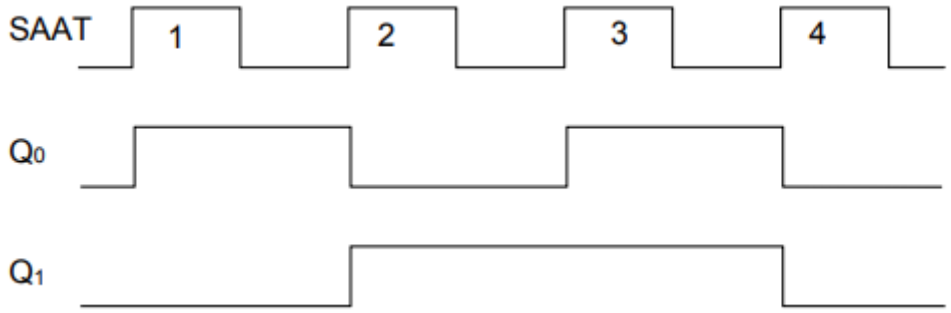
Synchronous Counters

- 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 =>

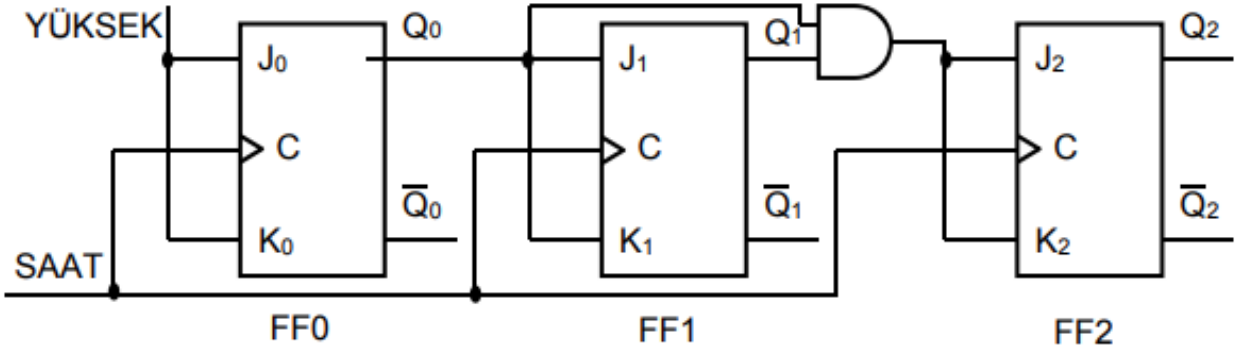


Synchronous Counters (continued)

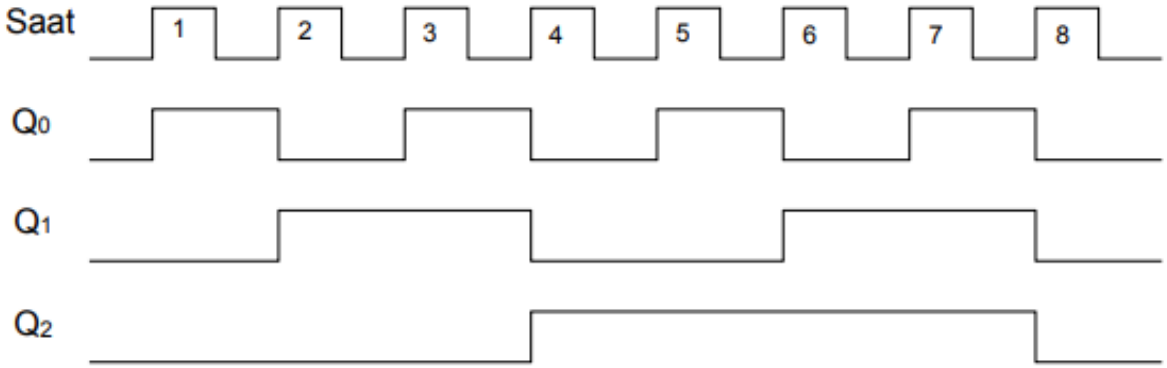




Synchronous Counters (continued)



Şekil-3.14 3 bit senkron sayıcı



Şekil-3.15 3 bit senkron sayıcı zamanlama diyagramı.

Design Example: Synchronous BCD

- Use the sequential logic model to design a synchronous BCD counter with D flip-flops
- State Table =>
- 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)

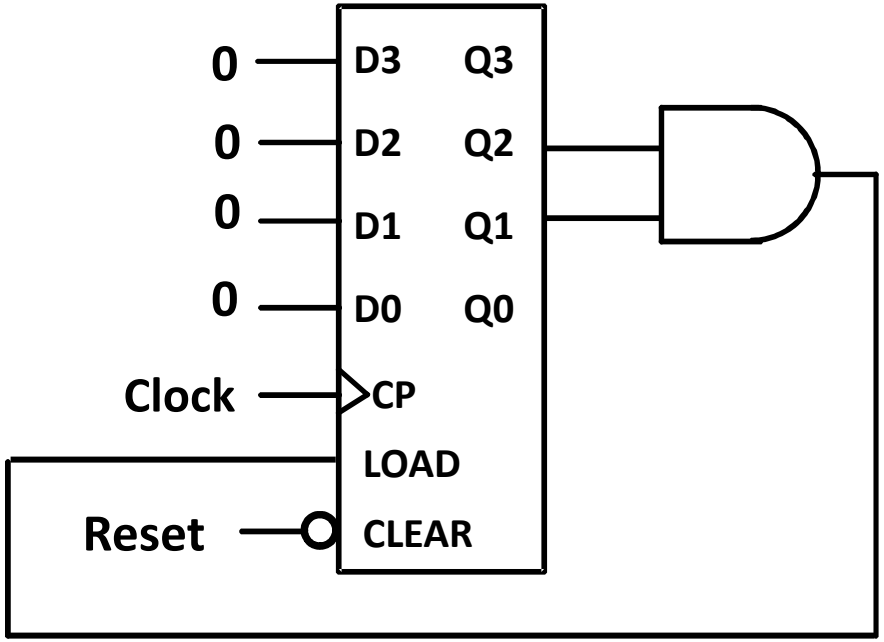
- Use K-Maps to two-level optimize the next state equations and manipulate into forms containing XOR gates:

$$D1 = \overline{Q1}$$
$$D2 = Q2 \oplus Q1\overline{Q8}$$
$$D4 = Q4 \oplus Q1Q2$$
$$D8 = Q8 \oplus (Q1Q8 + Q1Q2Q4)$$

- 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?

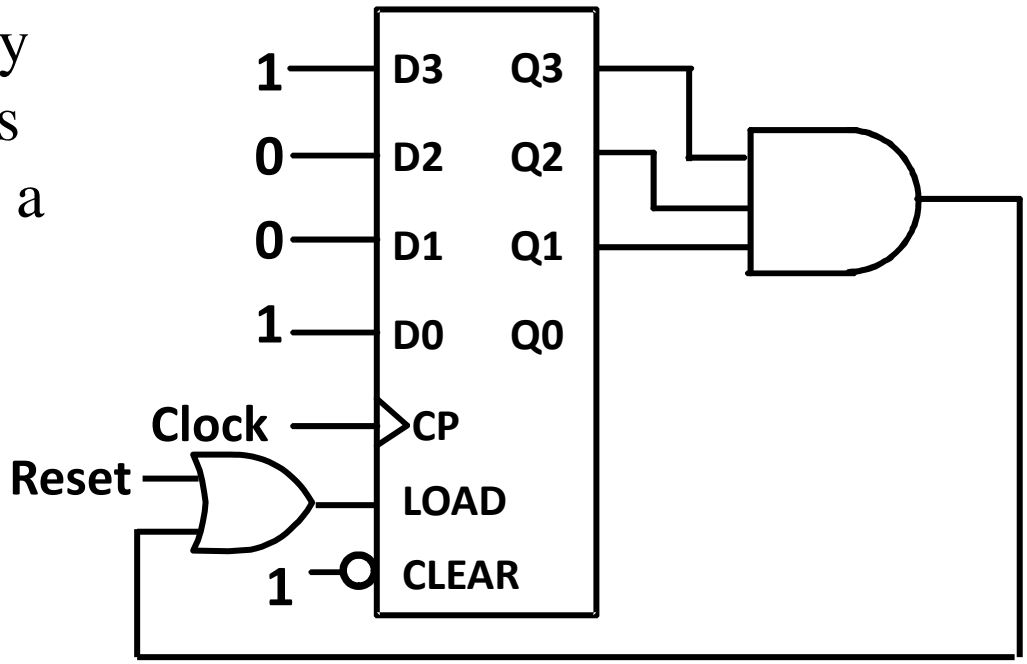
Counting Modulo 7: Synchronously Load on Terminal Count of 6

- 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, detection of 6 can be done with Load = Q2 Q1



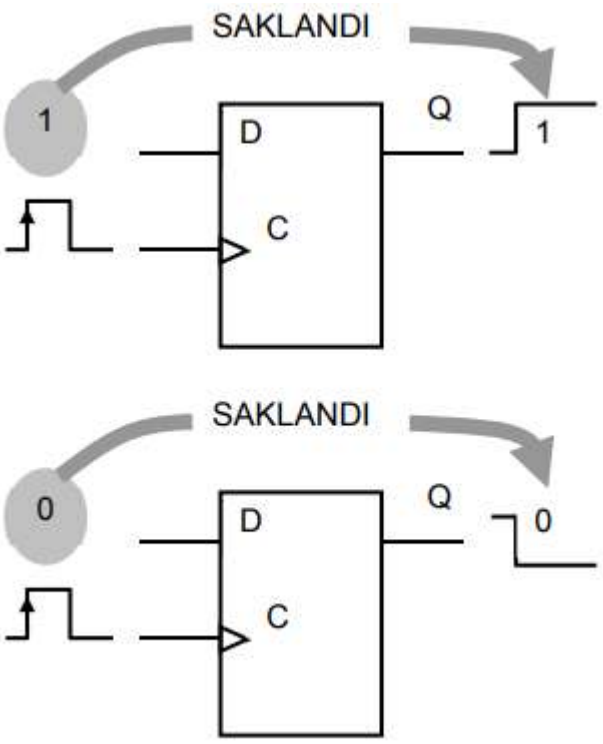
Counting Modulo 6: Synchronously Preset 9 on Reset and Load 9 on Terminal Count 14

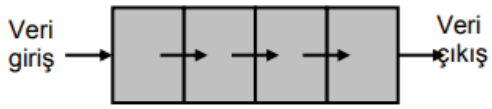
- A synchronous, 4-bit binary counter with a synchronous Load is to be used to make a Modulo 6 counter.
- Use the Load feature to preset the count to 9 on Reset and detection of count 14.



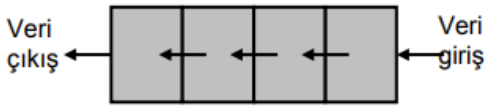
- This gives a count of 9, 10, 11, 12, 13, 14, 9, 10, 11, 12, 13, 14, 9, …
- If the terminal count is 15 detection is usually built in as Carry Out (CO)

Registers

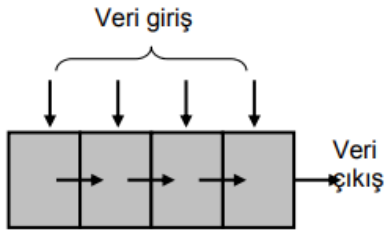




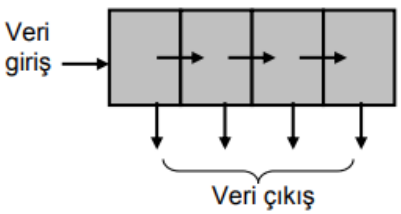
(a) Seri giriş-sağa ötele-seri çıkış.



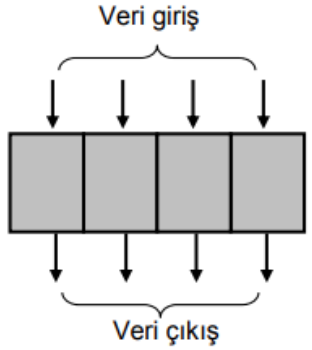
(b) Seri giriş-sola ötele-seri çıkış.



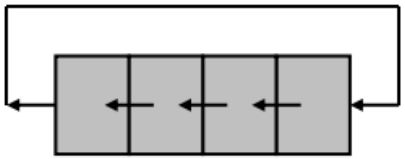
(c) Paralel giriş-sağa ötele-seri çıkış.



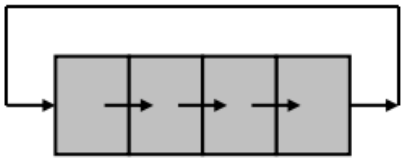
(d) Seri giriş-sağa ötele- Paralel çıkış.



(e) Paralel giriş- Paralel çıkış.

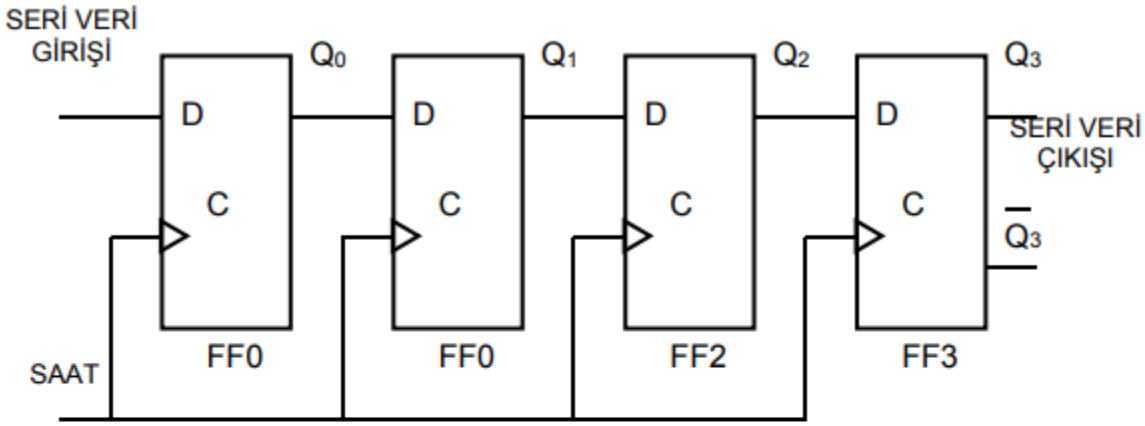


(g) Sola döndür.

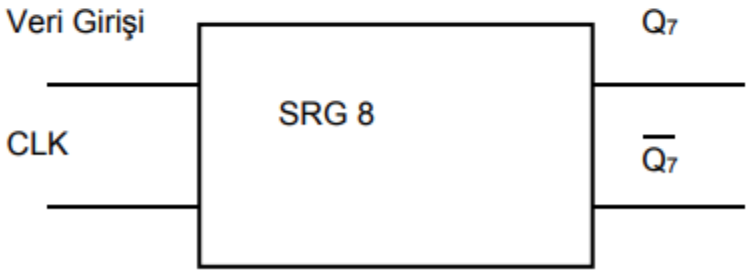


(f) Sağa döndür.

SISO registers

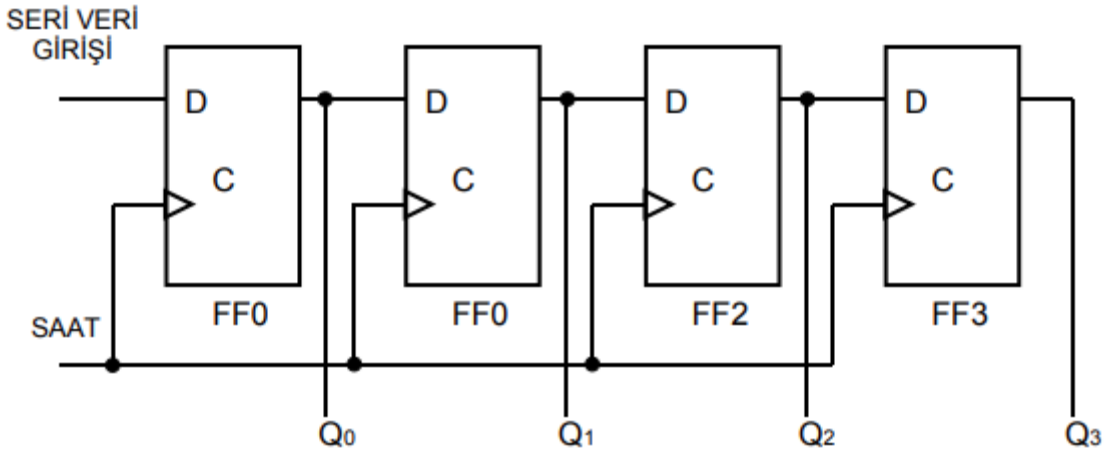


Şekil-4.3 Seri giriş seri çıkış kayar yazaç. (SISO)

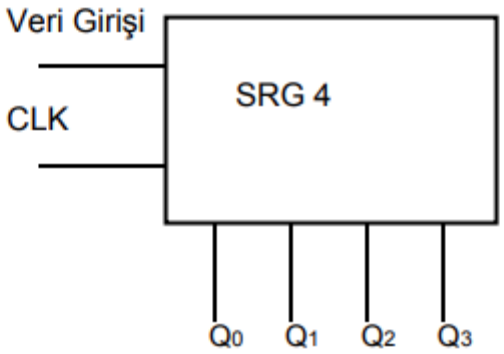


Şekil-4.4 8 bit SISO'nun mantık simgesi.

SIPO registers

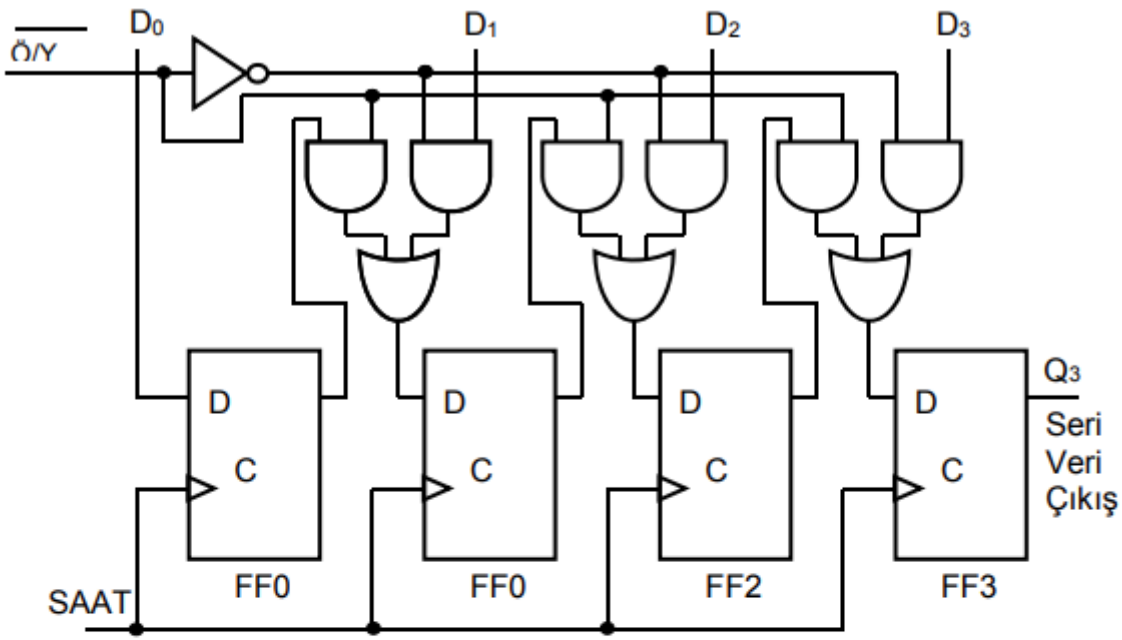


Şekil-4.5 Seri giriş paralel çıkış kayar yazaç. (SIPO)

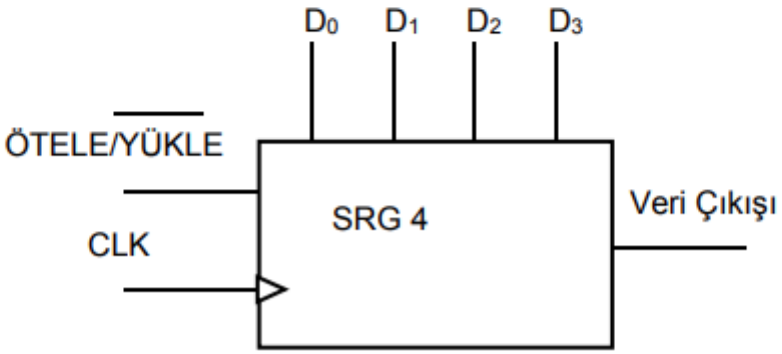


Şekil-4.6 4 bit SIPO'nun mantık simgesi.

PISO registers

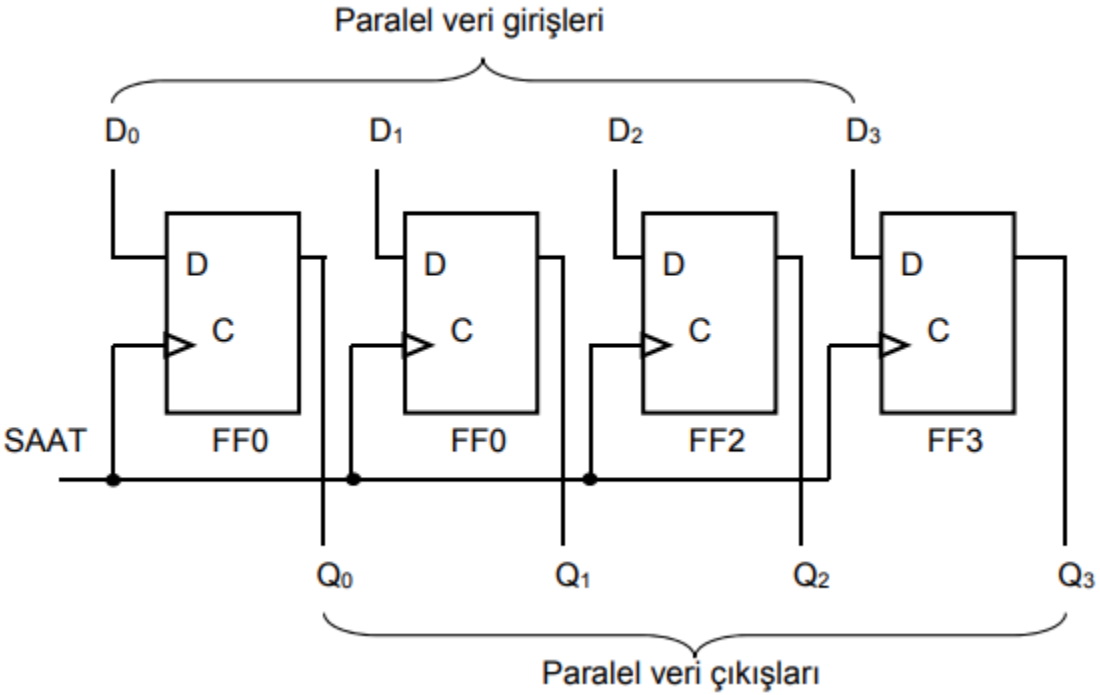


Şekil-4.7 Paralel giriş seri çıkış kayar yazaç. (PISO).



Şekil-4.8 4 bit PISO'nun mantık simgesi.

PIPO registers



Memory Definitions

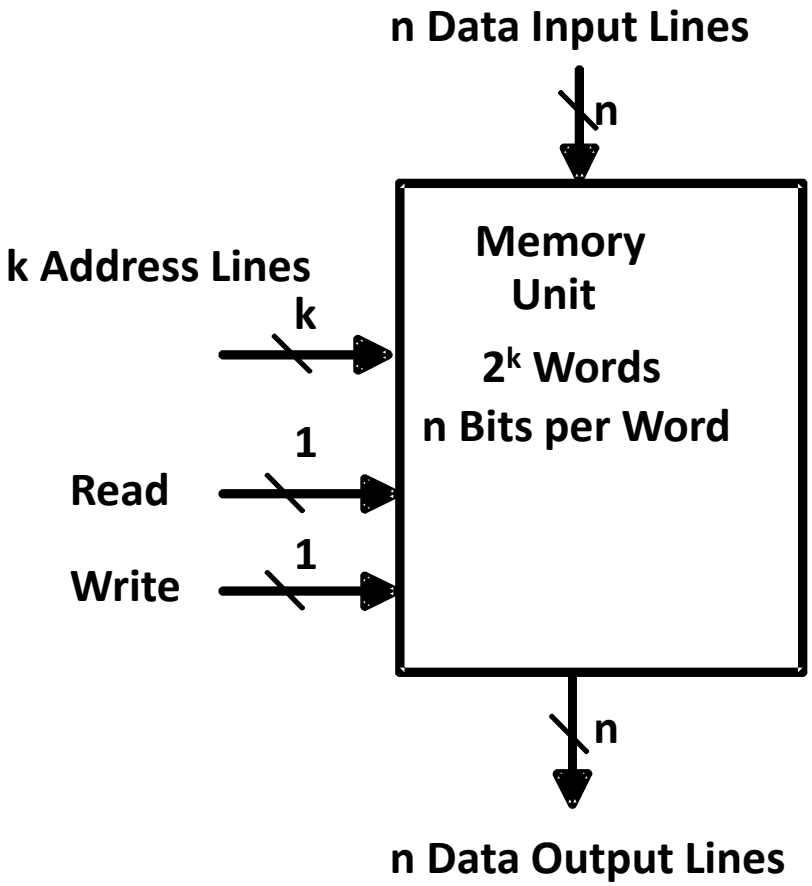
- Typical data elements are:
  - bit – a single binary digit
  - byte – a collection of eight bits accessed together
  - word – a collection of binary bits whose size is a typical unit of access for the memory. It is typically a power of two multiple of bytes (e.g., 1 byte, 2 bytes, 4 bytes, 8 bytes, etc.)
- Memory Data – a bit or a collection of bits to be stored into or accessed from memory cells.
- Memory Operations – operations on memory data supported by the memory unit. Typically, *read* and *write* operations over some data element (bit, byte, word, etc.).

Memory Organization

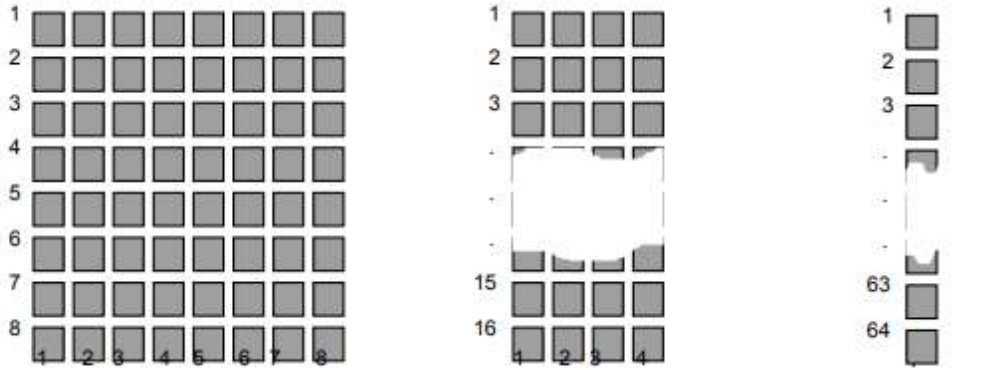
- Organized as an indexed array of words. Value of the index for each word is the memory address.
- Often organized to fit the needs of a particular computer architecture. Some historically significant computer architectures and their associated memory organization:
  - Digital Equipment Corporation PDP-8 – used a 12-bit address to address 4096 12-bit words.
  - IBM 360 – used a 24-bit address to address 16,777,216 8-bit bytes, or 4,194,304 32-bit words.
  - Intel 8080 – (8-bit predecessor to the 8086 and the current Intel processors) used a 16-bit address to address 65,536 8-bit bytes.

Memory Block Diagram

- A basic memory system is shown here:
- k address lines are decoded to address 2^k words of memory.
- Each word is n bits.
- Read and Write are single control lines defining the simplest of memory operations.

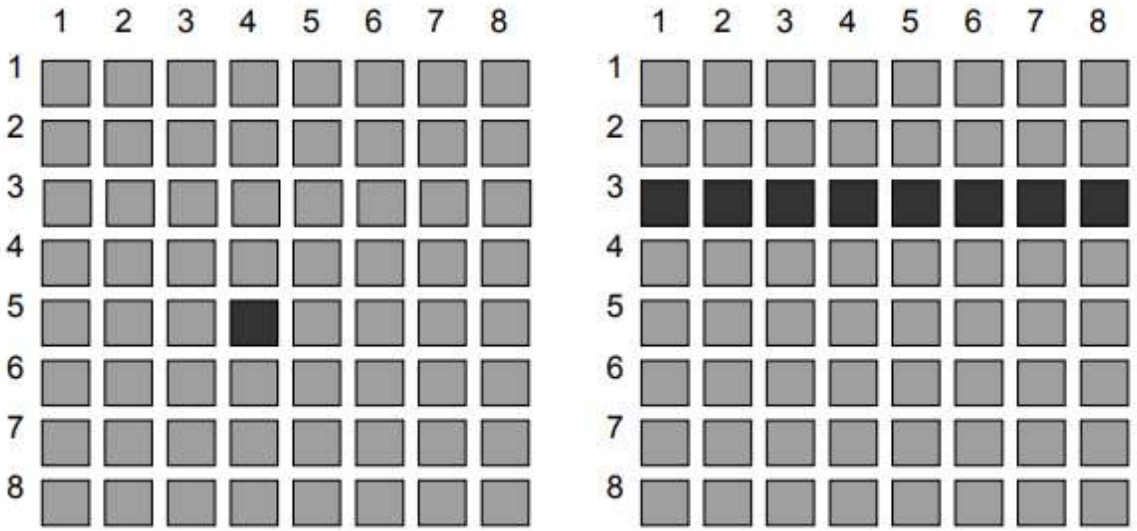


Memory



(a) 8×8 dizim (b) 16×4 dizim (c) 64×1 dizim

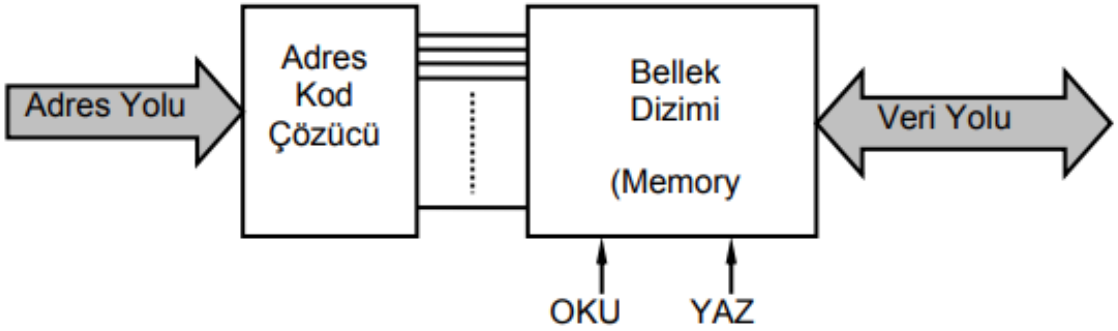
Şekil-5.1 Üç farklı şekilde hücrelerin dizimi



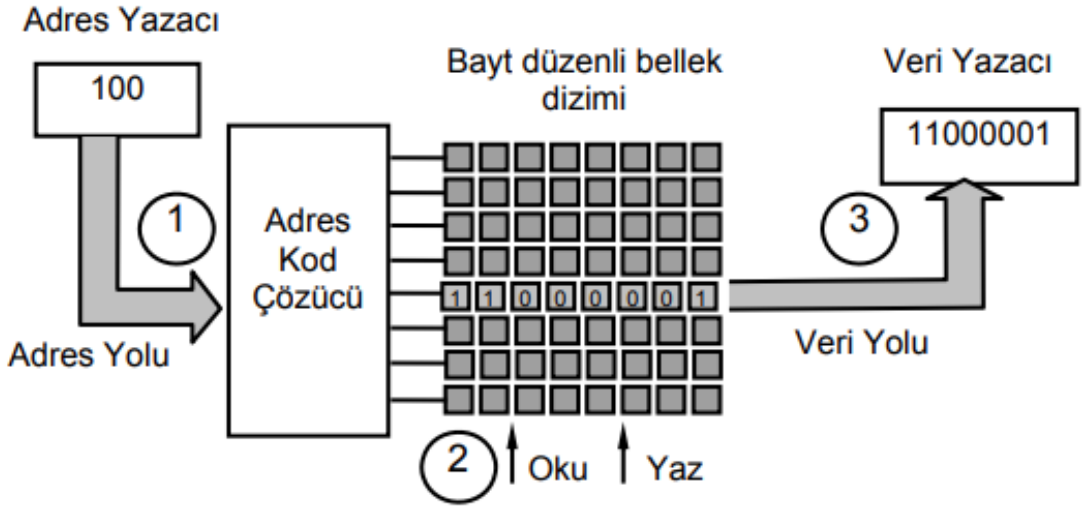
(a) Bitin adresi satır 5, sütun 3 (b) Baytın adresi satır 3

Şekil-5.2 Belleğin adreslenmesi.

Memory



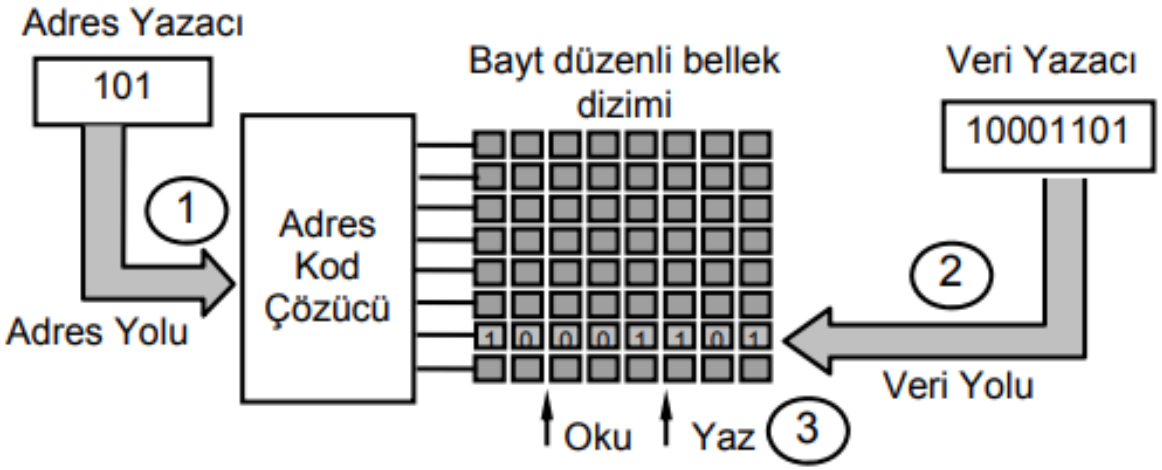
Şekil-5.3 Bellek blok diyagramı.



1. Adres kodu adres yoluna verilerek adres 4 seçilir.
2. Oku komutu belleğe uygulanır.
3. 4 nolu bellek satırın içeriği veri yolu kullanılarak veri yazacına alınır.

Şekil-5.4 Okuma işlemi.

Memory



1. Adres kodu adres yoluna verilerek adres6 seçilir.
2. Veri baytı veri yoluna yazılır.
3. Yaz komutu ile önceki veri silinerek yeni veri adres6 ya yazılır.

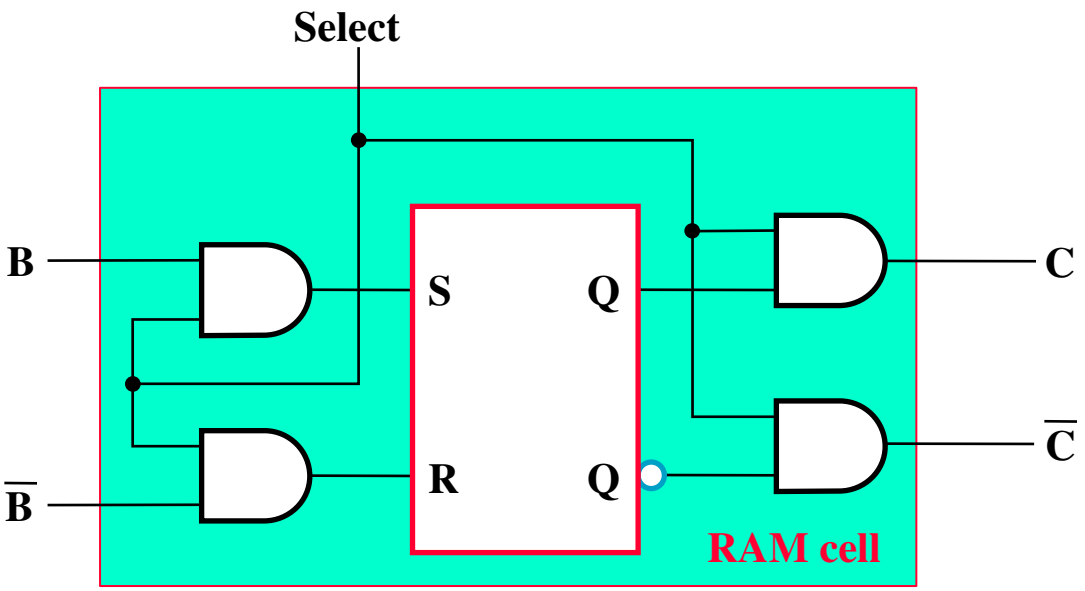
Şekil-5.5 Yazma işlemi.

RAM Integrated Circuits

- Types of random access memory
  - *Static* – information stored in latches
  - *Dynamic* – information stored as electrical charges on capacitors
    - Charge "leaks" off
    - Periodic *refresh* of charge required
- Dependence on Power Supply
  - *Volatile* – loses stored information when power turned off
  - *Non-volatile* – retains information when power turned off

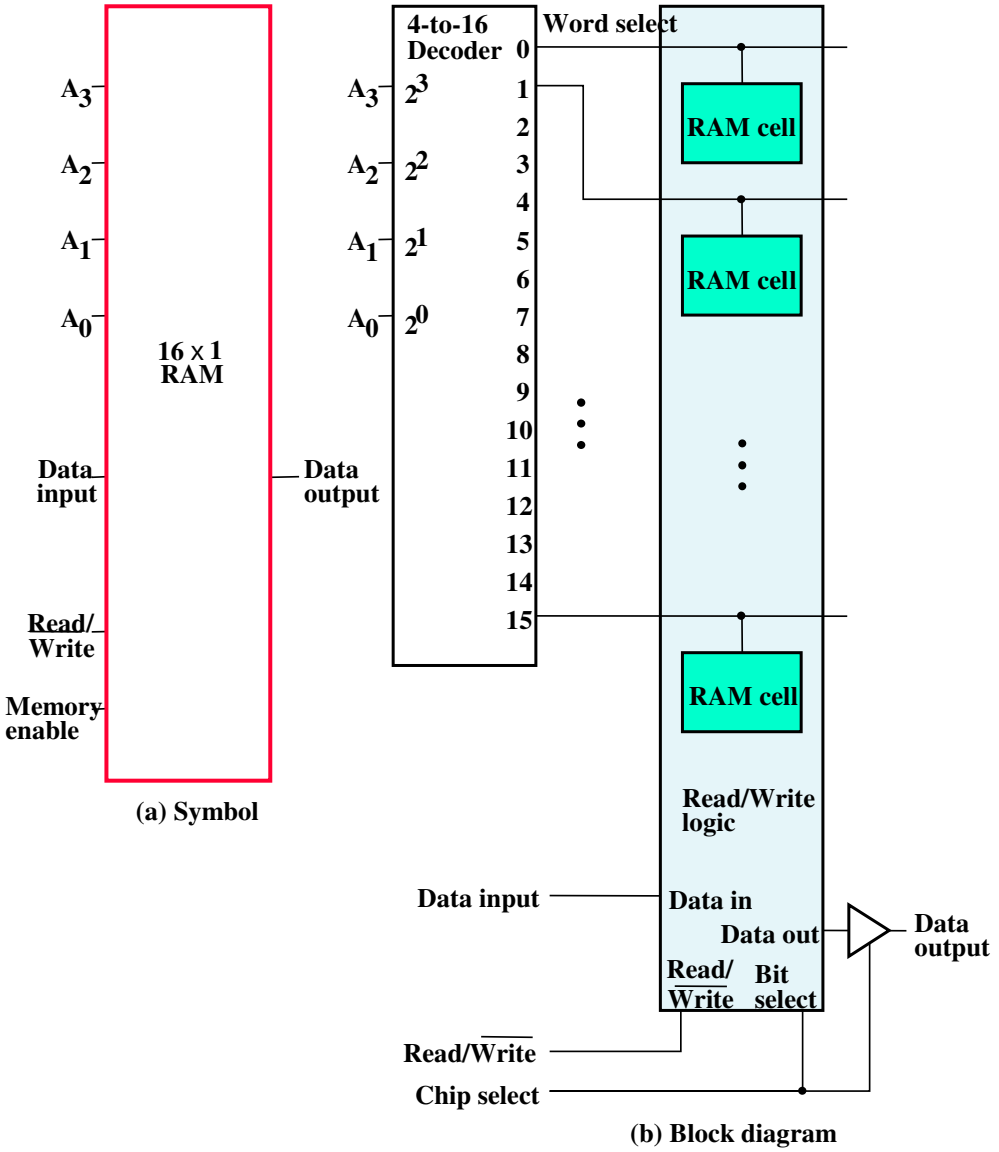
Static RAM □ Cell

- Array of storage cells used to implement static RAM
- Storage Cell
  - SR Latch
  - Select input for control
  - Dual Rail Data Inputs B and $\overline{B}$
  - Dual Rail Data Outputs C and $\overline{C}$



2^n-Word × 1-Bit RAM IC

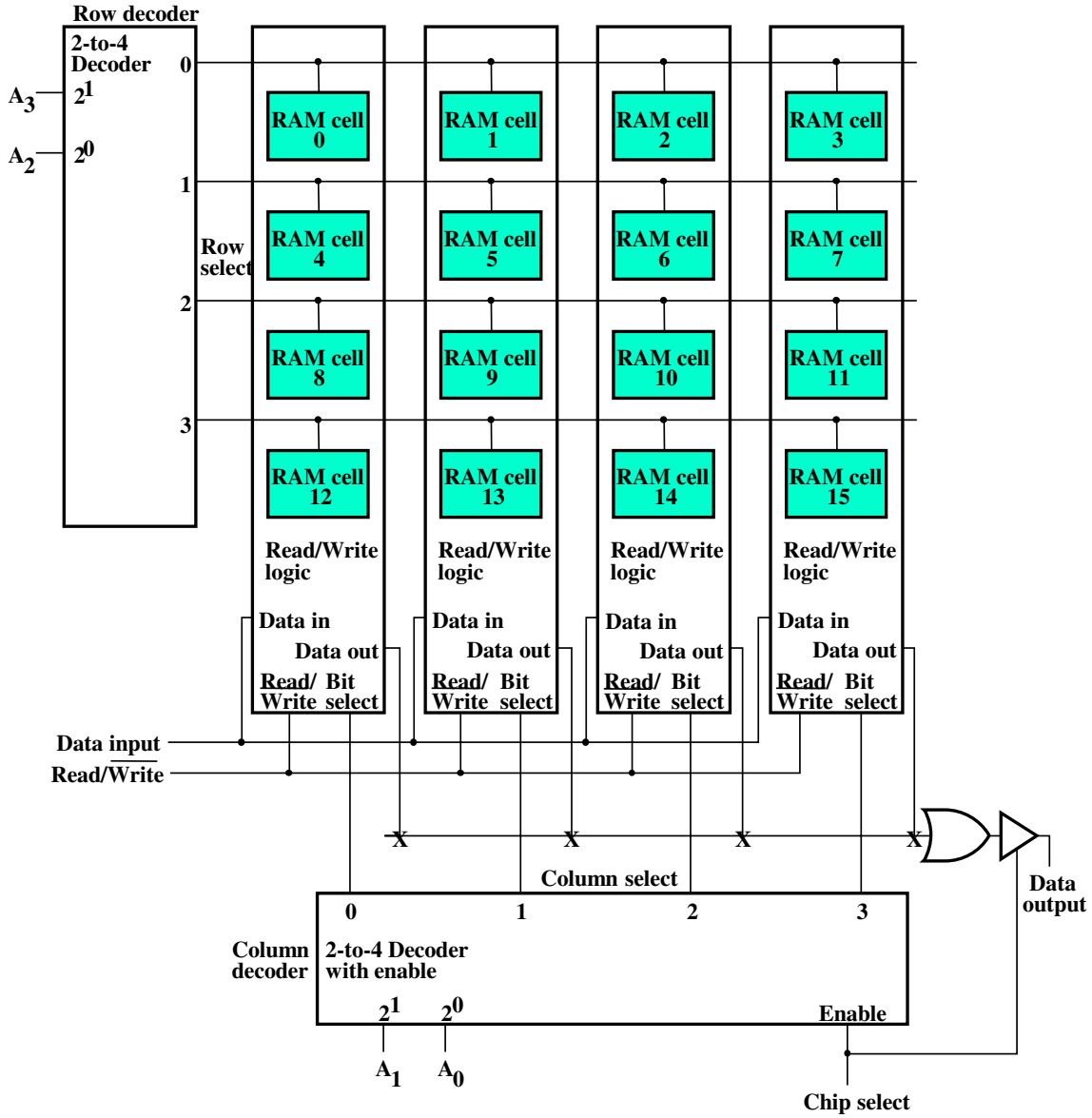
- To build a RAM IC from a RAM slice, we need:
  - Decoder decodes the n address lines to 2^n word select lines
  - A 3-state buffer
  - on the data output permits RAM ICs to be combined into a RAM with c × 2^n words



Cell Arrays and Coincident Selection

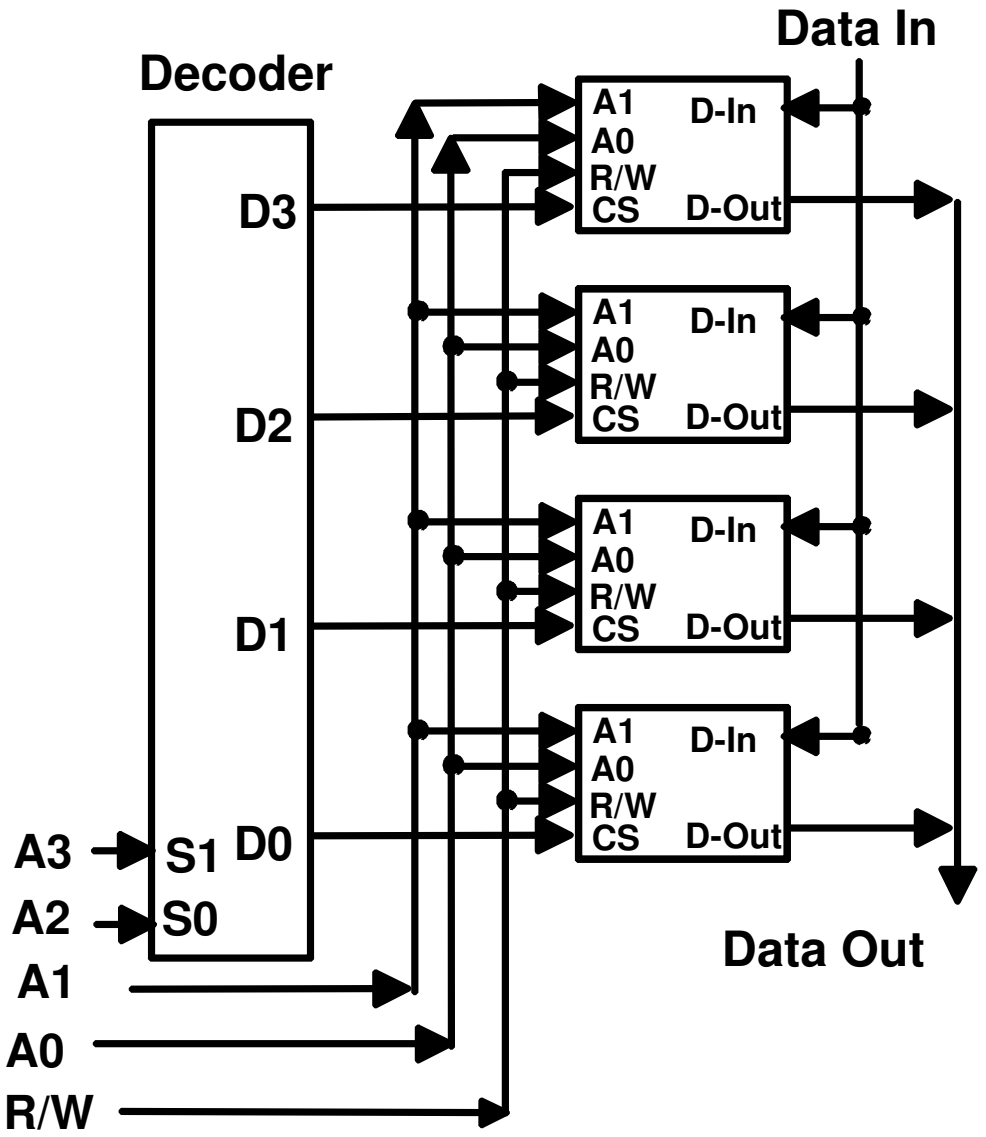
- Memory arrays can be very large =>
  - Large decoders
  - Large fanouts for the bit lines
  - The decoder size and fanouts can be reduced by approximately $\sqrt{\mathbf{n}}$ by using a coincident selection in a 2-dimensional array
  - Uses two decoders, one for words and one for bits
  - Word select becomes Row select
  - Bit select becomes Column select
- See next slide for example
  - A_3 and A_2 used for Row select
  - A_1 and A_0 for Column select

Cell Arrays and Coincident Selection (continued)



Making Larger Memories

- Using the CS lines, we can make larger memories from smaller ones by tying all address, data, and R/W lines in parallel, and using the decoded higher order address bits to control CS.
- Using the 4-Word by 1-Bit memory from before, we construct a 16-Word by 1-Bit memory. $\Rightarrow$



Making Wider Memories

- To construct wider memories from narrow ones, we tie the address and control lines in parallel and keep the data lines separate.
- For example, to make a 4-word by 4-bit memory from 4, 4-word by 1-bit memories
  $\Rightarrow$
- Note: Both 16x1 and 4x4 memories take 4-chips and hold 16 bits of data.