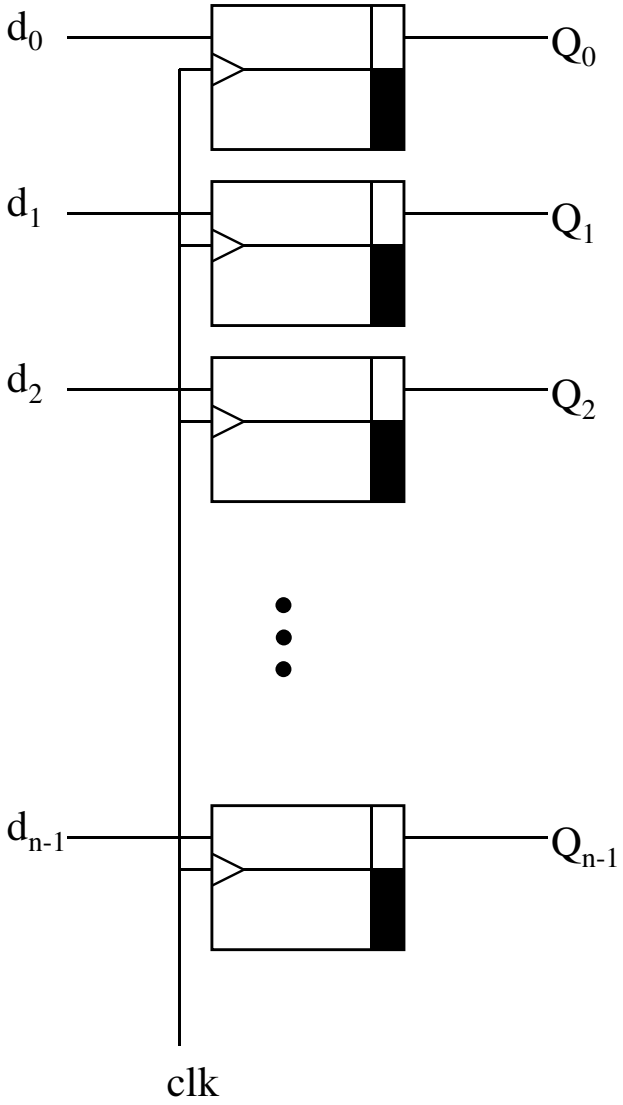
Special Sequential Circuits

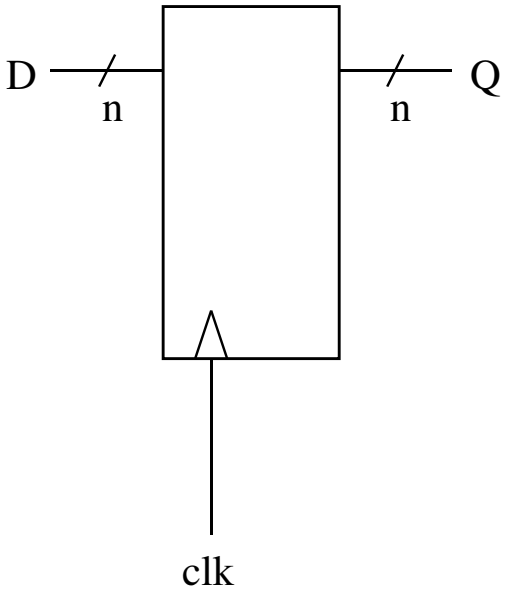
In this chapter, we will learn about some sequential circuits which are used as basic components in several digital circuits.

The Register

The register also known as word memory is a parallel wiring of n flip flops. In a register, a binary word of the length, n can be saved. The flip flops could be latches or system flip flops (e.g. D flip flops). The following sheets show the structure of a register and its circuit diagram.

Register

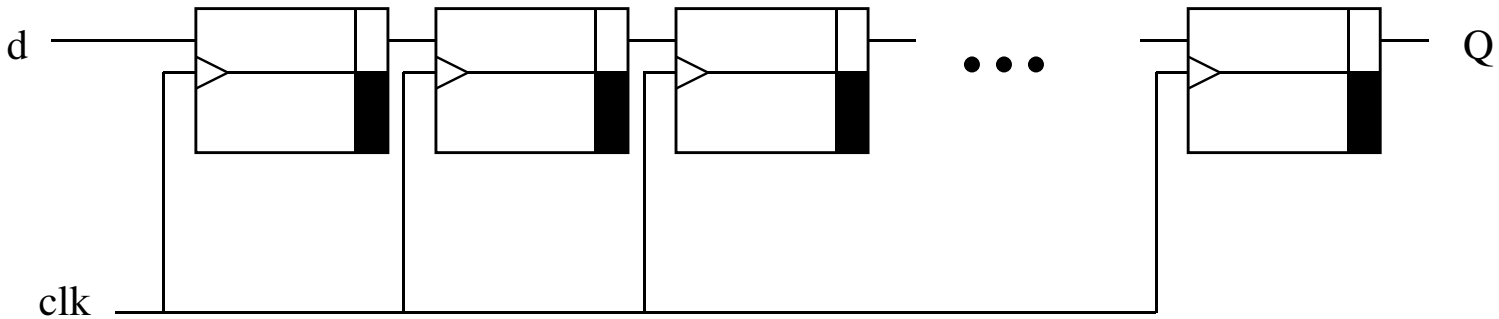




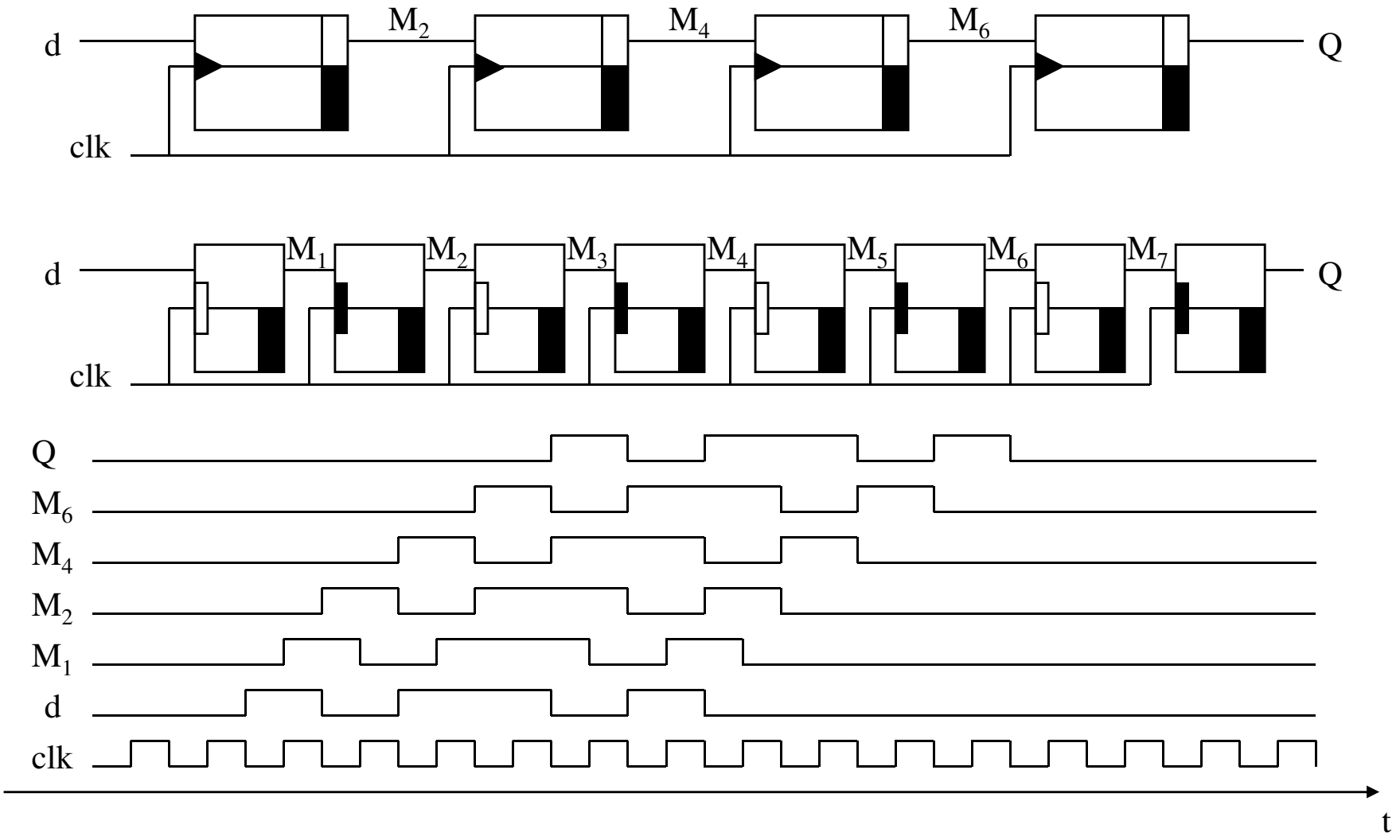
The Shift Register

The shift register is a word memory with bit serial access. It consists of a wiring of n D flip flops in series. In a shift register, a binary word of the length n can be saved because each clock means that there is an input of 1 bit. Correspondingly, the saved word will be sent to the output at the end of the shift register in n following clocks. The flip flops have to be system flip flops. The following sheets show the structure of a shift register. We know already that we can construct a system flip flop as Master slave flip flop using two latches. This structure and the progress of a signal 001011010000 along the shift register can be seen in the slide following the next slide.

Shift Register



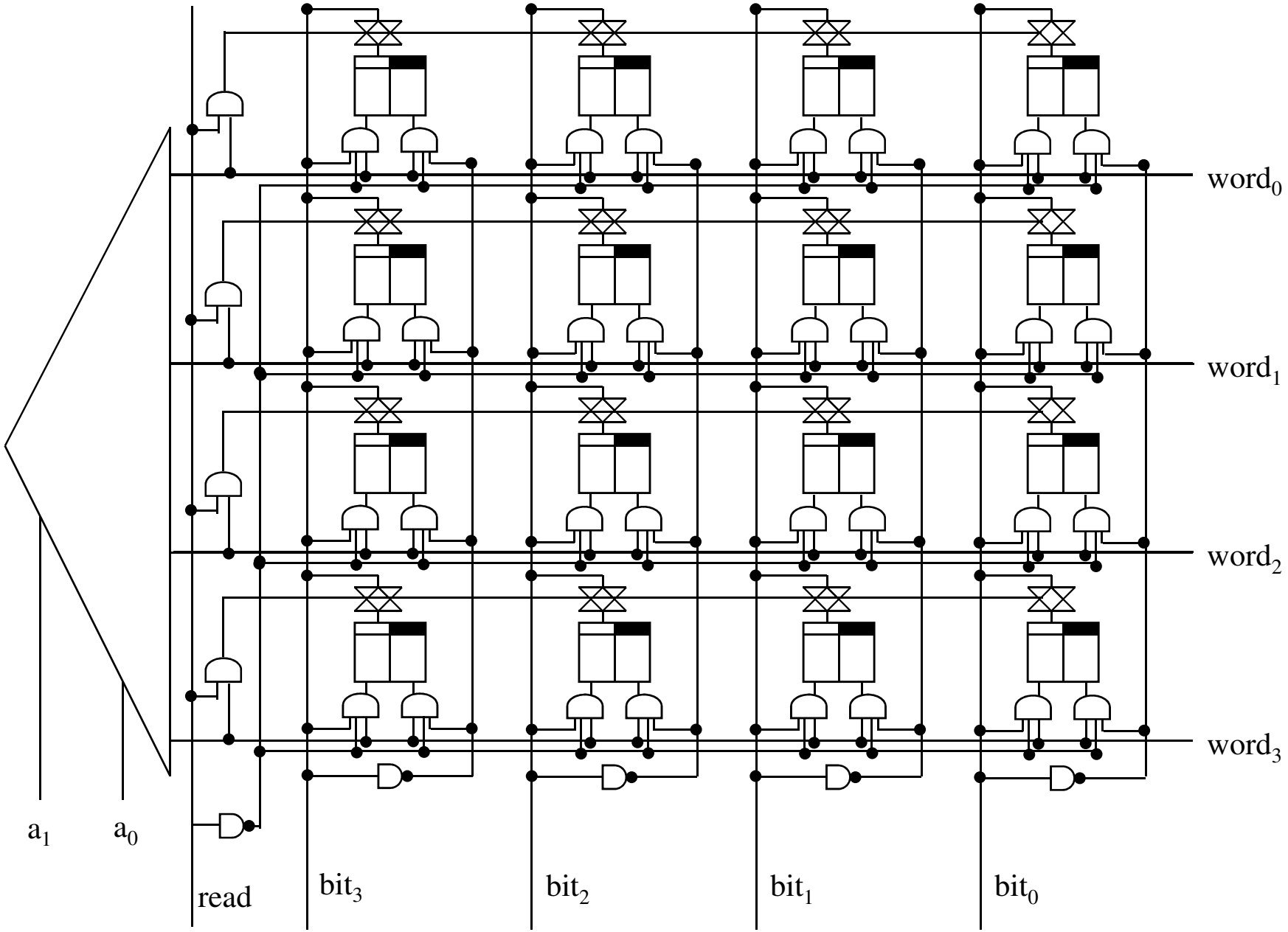
4-bit-Shift Register



The RAM

The Random Access Memory is a memory for N words with width as m bit. Every of this word is identifiable by one address. The address has n = log N bits. When one address $a_{n-1}a_{n-2}...a_1a_0$ is provided, one can access the respective word either by reading or writing. In this intended function, one loads the address to a decoder, which decodes a 1-to-N code. For every N word, it provides a so-called word signal. Based on the decoding process, a "1" will be placed on the selected word signal while the rest a "0". On each of these word signals, one register of the width m Bit is now available. By activating the selected word signal, this register can now be read or written externally.

A simple structure of a RAM with N=4 and m=4 is shown on the following slide. The memory building blocks are RS flip flops. In each case, four of them are connected to a parallel word memory.


word0
word1
word2
word3
a1
a0
read
bit3
bit2
bit1
bit0

The Functional Behaviour of the RAM

A m-bit word will be placed on the bit signal during a write process which shall be written in the cell i. The address is sent to the decoder. Due to that, a "1" is applied on the i-th word signal. The AND gate before the flip flops thus allows the bit value at the s-input of every RS flip flops to pass and at the r-input the inverted bit value to pass through. All other word signals are "0". (i.e. the AND gate generates the memory combination r = 0 and s = 0 on all other RS flip flops). Thus, the cell i will "store" the new word. All other cells stores the old values.

During a read process, nothing will be generated externally on the bit signals as they are in the high impedance state Z. Based on the word signals, the transmission gates at the output of the flip flops in the i-th cell are opened. Thus, the stored values reach the bit signals. Therefore, the values of the i-th row will be transmitted to the output of the RAM.

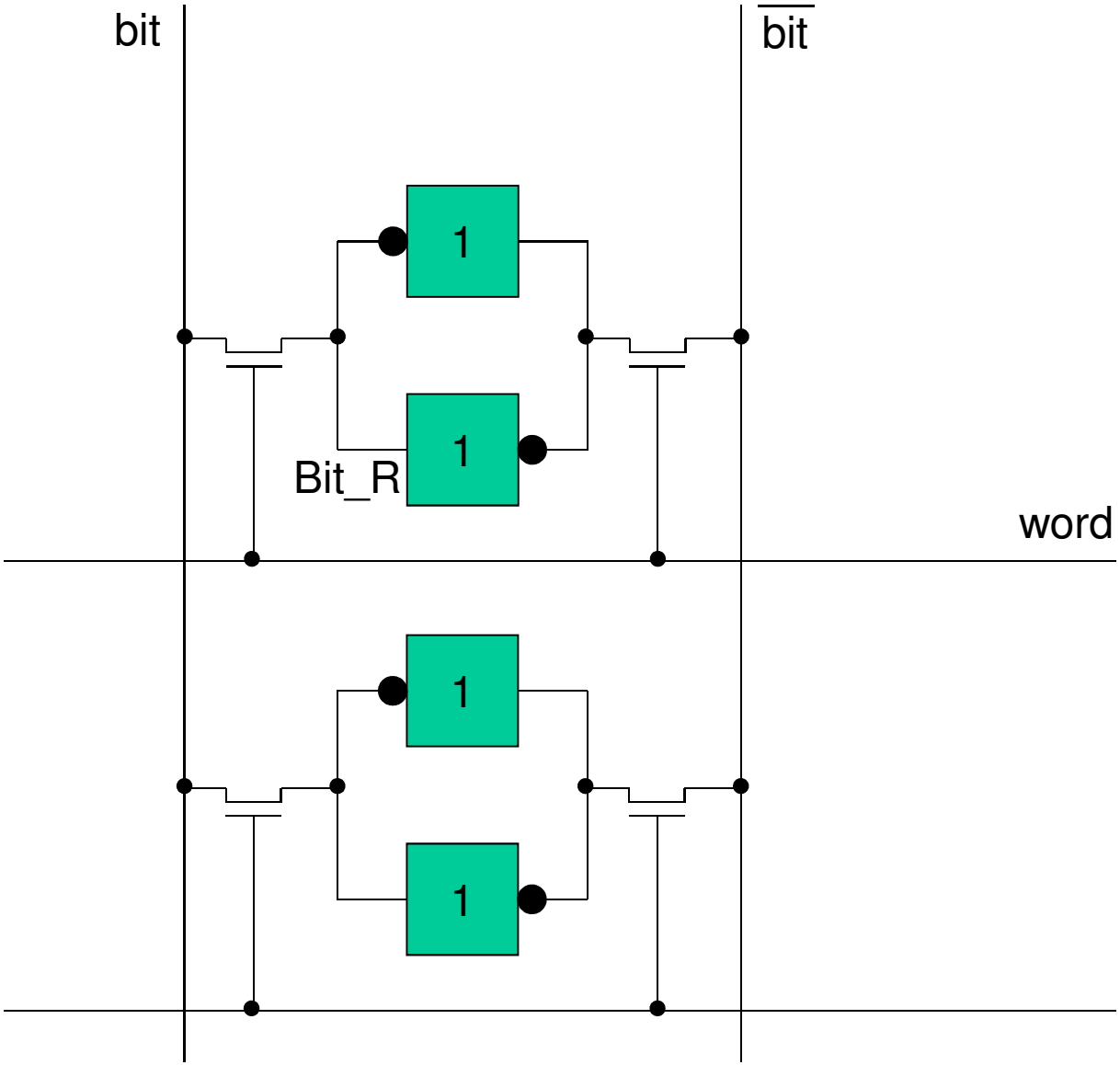
In reality, the RAM is not built from RS flip flops and AND gates and transmission gates as these realizations become too complex and costly. One at first differentiate between SRAM (static RAM) and DRAM (dynamic RAM). The static RAM stores and retains a value as long as the power supply remains on. Our RAM made of RS flip flops is an example of a realization of a static RAM.

The dynamic RAM saves all values only for a short time. And in fact, the capacitance of a transistor gate (polycrystalline silicon) acts as a storage medium, as compared to the exploitation of the Source/Drain (Diffusion). Of course, such a memory is volatile. Hence, every bit in this dynamic RAM must be refreshed from time to time. That occurs automatically at a fixed interval in a cell to cell manner that all the bits in the RAM is read once and written back again without changes. A typical time interval for the refreshing cycle of today's DRAMs is one millisecond.

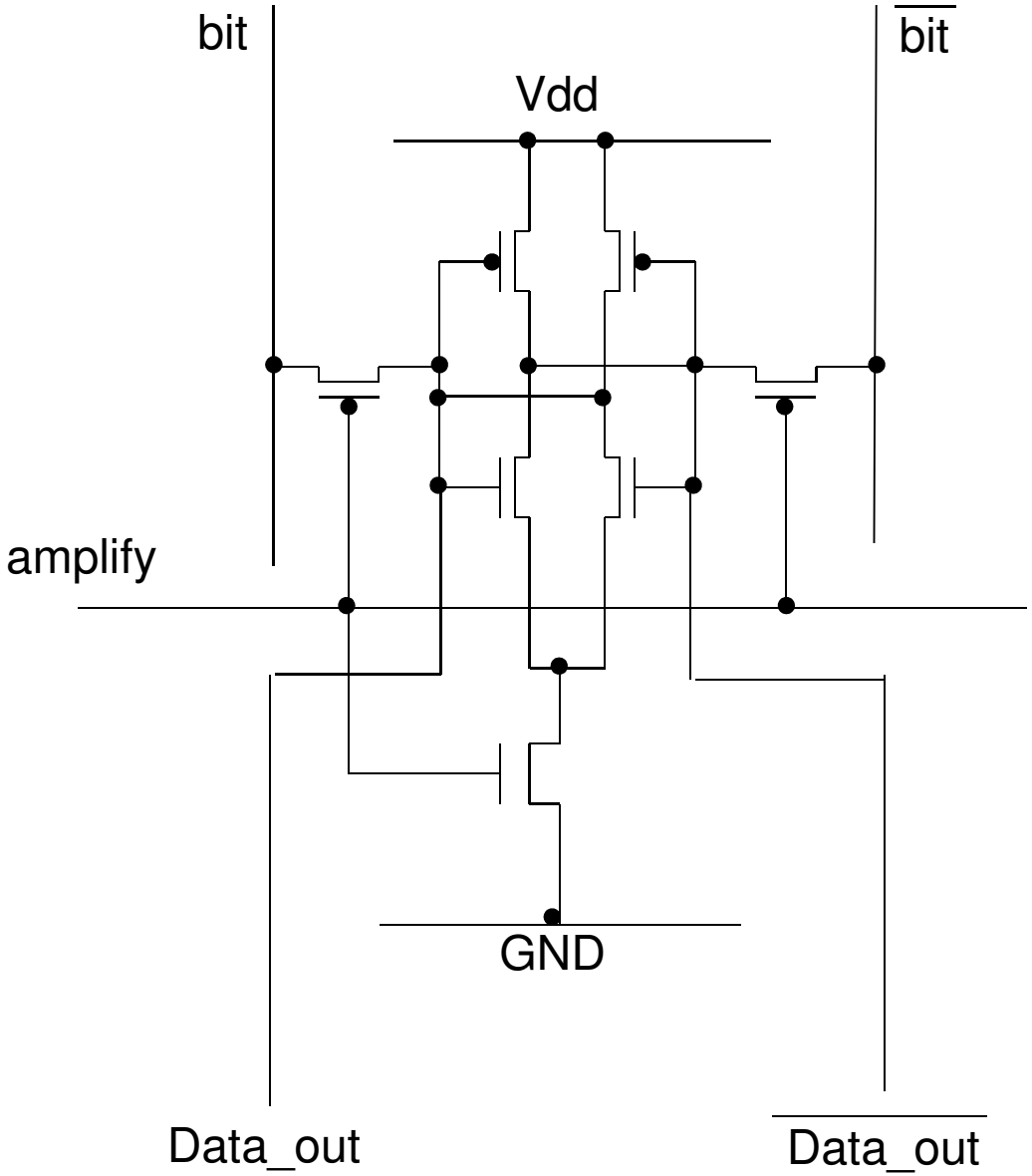
SRAM is faster (shorter access time) and more expensive. However, SRAM requires six transistors for each saved bit, whereas DRAM manages only with one transistor per bit. Therefore, DRAM has a higher memory capacitance per chip area.

Today's memory chips are arranged quadratically. It uses two decoders, one for the rows and the other for the columns. Row and column addresses are thereby of the same length. In this manner, one can manage half of the address pins and uses the address signals in a time multiplexed manner. The row addresses are always first transmitted and the column addresses are then transmitted over the same pins.

Assembly of a 6-Transistor SRAM-Cell



Differential Sense Amplifier (Differenzverstärker)



Austin Semiconductor, Inc.

SRAM
MT5C2564

64K x 4 SRAM
SRAM MEMORY ARRAY

AVAILABLE AS MILITARY SPECIFICATIONS
- SMD 5962-88681
- MIL-STD-883

FEATURES
- High Speed: 15, 20, 25, 35, 45, 55, and 70
- Battery Backup: 2V data retention
- Low power standby
- High-performance, low-power, CMOS double-metal process
- Single +5V (±10%) Power Supply
- Easy memory expansion with CE\
- All inputs and outputs are TTL compatible

OPTIONS — MARKING

- **Timing**

15ns access	-15
20ns access	-20
25ns access	-25
35ns access	-35
45ns access	-45
55ns access	-55*
70ns access	-70*

- **Package(s)**

Ceramic DIP (300 mil)	C	No. 106
Ceramic LCC	EC	No. 204

- **Operating Temperature Ranges**

Industrial (-40°C to +85°C)	IT
Military (-55°C to +125°C)	XT

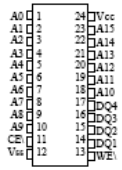
- 2V data retention/low power — L

*Electrical characteristics identical to those provided for the 45ns access devices.

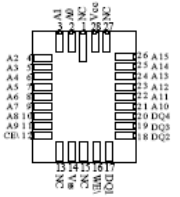
For more products and information please visit our web site at *www.austinsemiconductor.com*

PIN ASSIGNMENT
(Top View)

24-Pin DIP (C)
(300 MIL)



28-Pin LCC (EC)



GENERAL DESCRIPTION

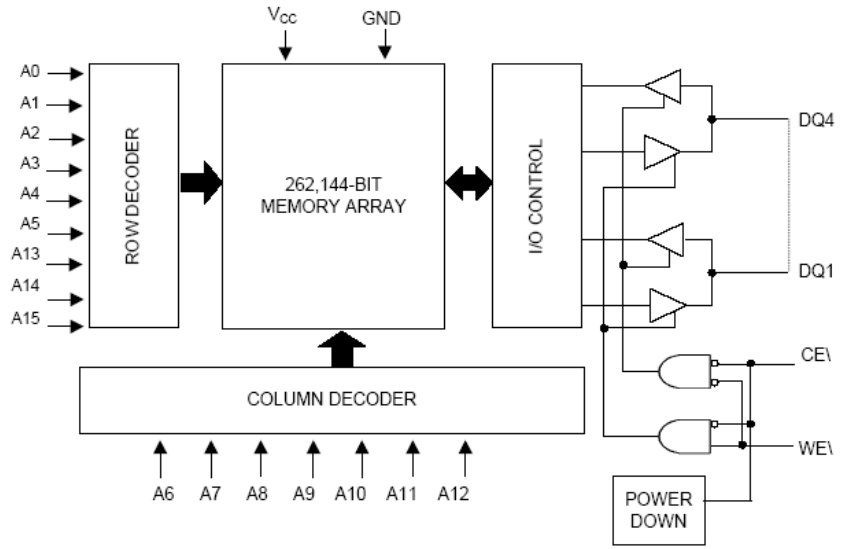
The Austin Semiconductor SRAM family employs high-speed, low-power CMOS and are fabricated using double-layer metal, double-layer polysilicon technology.

For flexibility in high-speed memory applications, Austin Semiconductor offers chip enable (CE\) on all organizations. This enhancement can place the outputs in High-Z for additional flexibility in system design. The x4 configuration features common data input and output.

Writing to these devices is accomplished when write enable (WE\) and CE\ inputs are both LOW. Reading is accomplished when WE\ remains HIGH and CE\ goes LOW. The device offers a reduced power standby mode when disabled. This allows system designs to achieve low standby power requirements.

These devices operate from a single +5V power supply and all inputs and outputs are fully TTL compatible.

FUNCTIONAL BLOCK DIAGRAM



TRUTH TABLE

MODE	CE\	WE\	DQ	POWER
STANDBY	H	X	HIGH-Z	STANDBY
READ	L	H	Q	ACTIVE
WRITE	L	L	D	ACTIVE

A technique whereby the DRAMs can be made faster are the so-called SDRAMs (synchronous DRAMs). It is based on DRAM, whereby it is synchronized to the external clocking and forced to the maximum speed of the processor memory bus.

RAMBUS is pre-dominantly used for Intel processor asynchronous memory technique, with which by matching exactly the capacitance, inductance, resistance would optimized the performance runtime. This technology is however, at this stage again on the fallback.

CDRAM (Cache DRAM) is a combination of SRAM and DRAM. It concerns the principles of DRAM, with which however, the last row that has been read out at the end, is stored in a small separate SRAM (Cache). Since the same row in the memory is often accessed frequently and repeatedly in a successive manner, the fast SRAM is utilized for that while on the other hand, the high storage density of the DRAM is used.

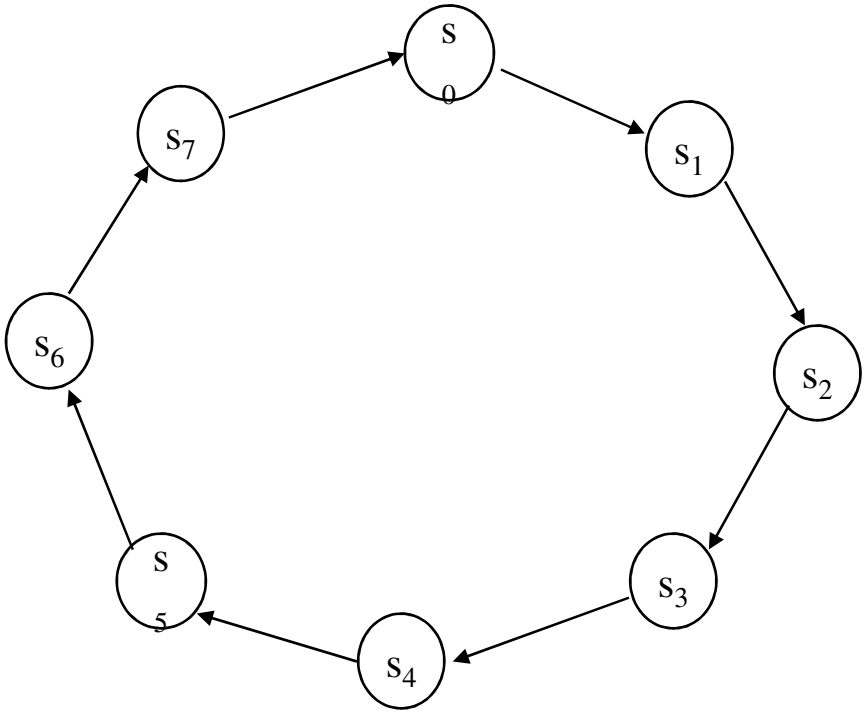
Counters

Counters are special sequential circuits which normally have only a few (sometimes no) inputs. Often, their outputs are identical with their states. The state is the current position of the counter. Very often, the counter value is a binary coded number. The clock makes the counter count, hence changing its state. Due to that, we are actually counting the number of rising clock edges.

From the viewpoint of the automaten theory, so far we have created all our sequential circuits as Mealy automaten graphs. But if the output does not depending on the current input but only from the current state, then we have a so-called “Moore automaten”. If the output is identical with the current state, then we have the so-called “Medwedew-Automaten”.

In this chapter, we will learn about several different counters. As introduction, we choose a synchronous modulo eight counter. On the following slides, the automaten graph for this counter is shown: The modulo eight counter goes through its eight states in a cyclic manner whereby it makes a step at every clock.

Modulo-8-Counter



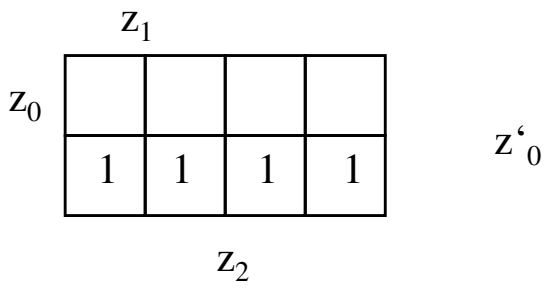
Coding of the States:

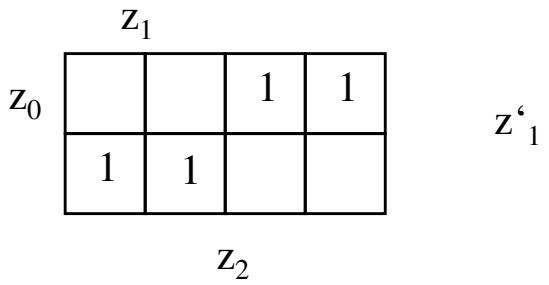
	z_2	z_1	z_0
s_0:	0	0	0
s_1:	0	0	1
s_2:	0	1	0
s_3:	0	1	1
s_4:	1	0	0
s_5:	1	0	1
s_6:	1	1	0
s_7:	1	1	1

Modulo-8-Counter

Truth Table:

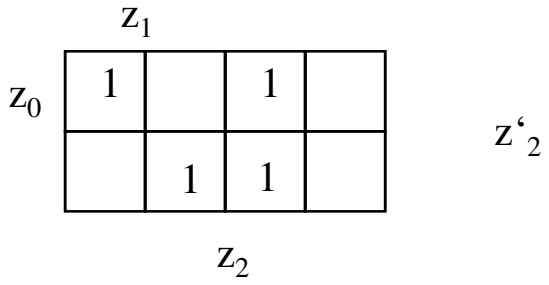
z_2	z_1	z_0	z'_2	z'_1	z'_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0





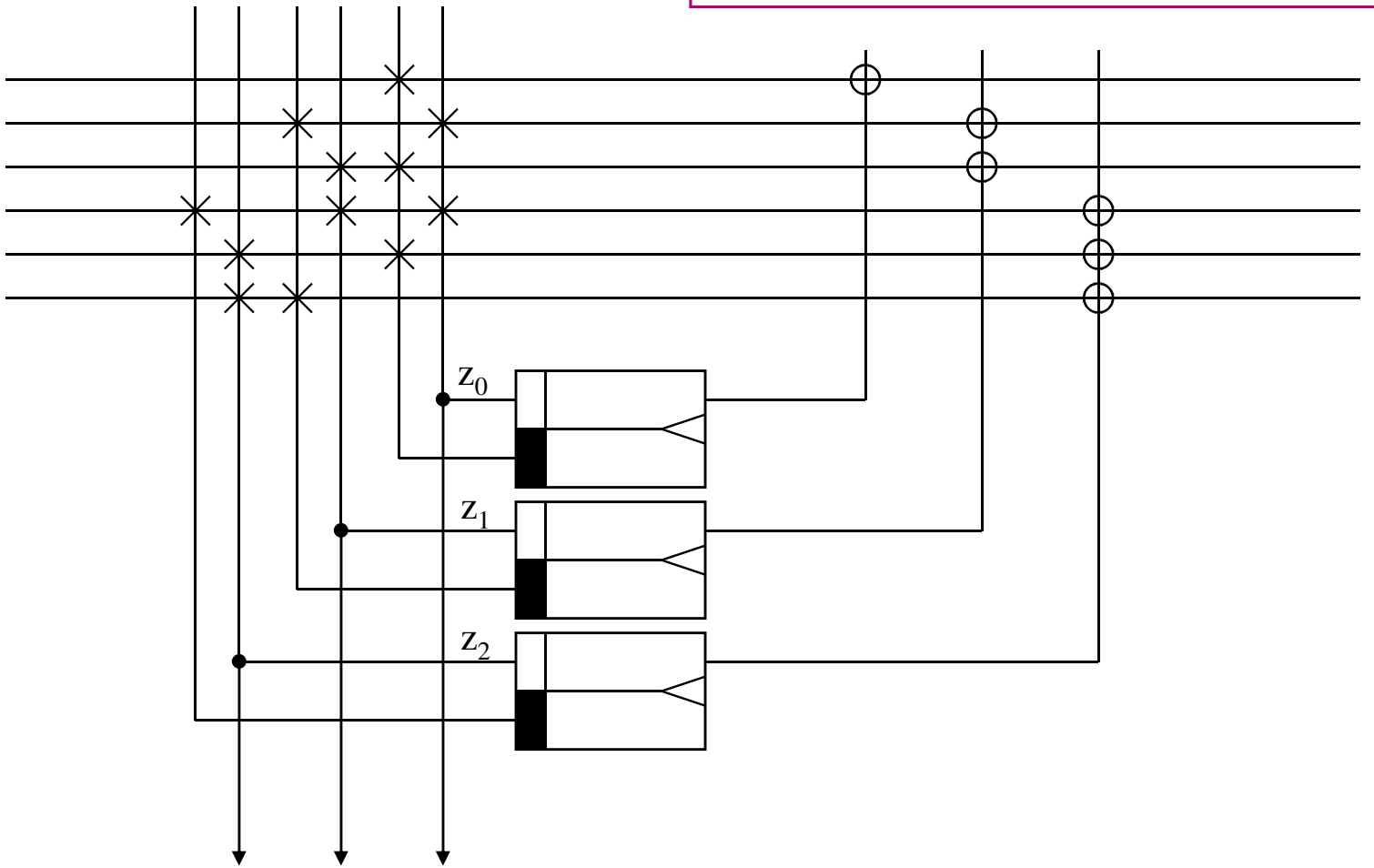
Results:

$$z'_0 = \overline{z_0}$$
$$z'_1 = z_0\,\overline{z_1} + \overline{z_0}\,z_1$$
$$z'_2 = z_0\,z_1\,\overline{z_2} + \overline{z_0}\,z_2 + \overline{z_1}\,z_2$$

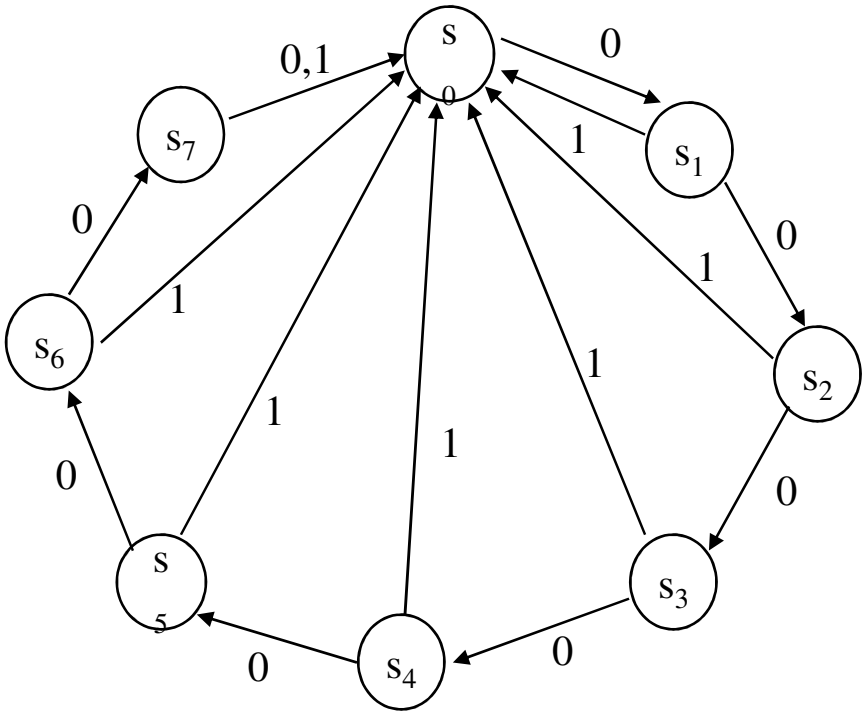


Modulo-8-Counter

Realization as FPLA:



So far, our modulo 8 counter has a small mistake: We cannot set it to a defined starting state. If we want to be able to do that, the sequential circuit needs an additional input, e.g. a reset signal. If this signal is 1, the counter shall go to the state s_0. The corresponding modified state graph is as follows:



Truth Table:

reset	z_2	z_1	z_0	z'_2	z'_1	z'_0
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
1	X	X	X	0	0	0

The reader might think of the realization himself.

Other Counters

Normally D flip flops are the adequate components to save states of sequential circuits, especially for counters but other types of flip flops are also often used, due to the fact that the circuits of the combinatorial circuits can be simplified. The following example is a modulo 8 counter which shall only use RS flip flops. In the truth table, "don’t cares" can be inserted in many places, more exactly, always if the wanted value can be achieved by “setting” or “saving”:

z_2	z_1	z_0	s_2	r_2	s_1	r_1	s_0	r_0
0	0	0	0	X	0	X	1	0
0	0	1	0	X	1	0	0	1
0	1	0	0	X	X	0	1	0
0	1	1	1	0	0	1	0	1
1	0	0	X	0	0	X	1	0
1	0	1	X	0	1	0	0	1
1	1	0	X	0	X	0	1	0
1	1	1	0	1	0	1	0	1

z_0 \ z_1				
	1		X	
		X	X	

z_2

$s_2 = z_0 z_1 \overline{z_2}$

z_0 \ z_1				
		1		X
	X			X

z_2

$r_2 = z_0 z_1 z_2$

z_0 \ z_1				
			1	1
	X	X		

z_2

$s_1 = z_0 \overline{z_1}$

z_0 \ z_1				
	1	1		
			X	X

z_2

$r_1 = z_0 z_1$

z_0 \ z_1				
	1	1	1	1

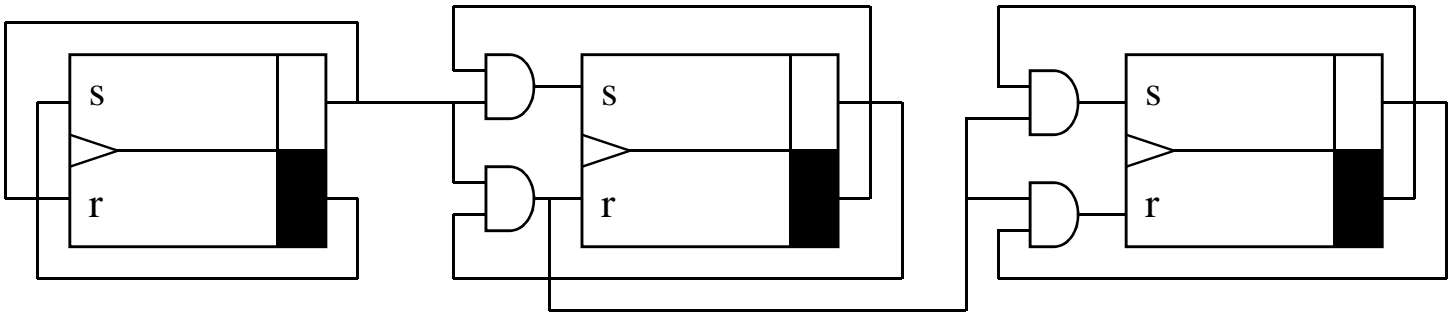
z_2

$s_0 = \overline{z_0}$

z_0 \ z_1				
	1	1	1	1

z_2

$r_0 = z_0$



Usage of JK flip flops for designing counters

We can simplify the circuit even more by using JK flip flops instead of RS flip flops. It is useful to us since the input J = K = 1 can be used to toggle the state.

z_2	z_1	z_0	j_2	k_2	j_1	k_1	j_0	k_0
0	0	0	0	X	0	X	1	X
0	0	1	0	X	1	X	X	1
0	1	0	0	X	X	0	1	X
0	1	1	1	X	X	1	X	1
1	0	0	X	0	0	X	1	X
1	0	1	X	0	1	X	X	1
1	1	0	X	0	X	0	1	X
1	1	1	X	1	X	1	X	1

z_0 \ z_1				
	1	X	X	
		X	X	

z_2

$j_2 = z_0 z_1$

z_0 \ z_1				
	X	1		X
	X			X

z_2

$k_2 = z_0 z_1$

z_0 \ z_1				
	X	X	1	1
	X	X		

z_2

$j_1 = z_0$

z_0 \ z_1				
	1	1	X	X
			X	X

z_2

$k_1 = z_0$

z_0 \ z_1				
	X	X	X	X
	1	1	1	1

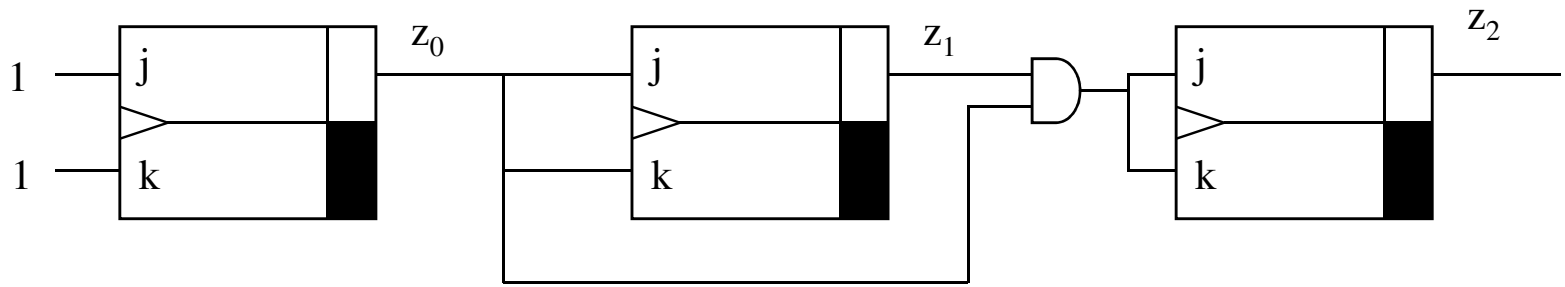
z_2

$j_0 = 1$

z_0 \ z_1				
	1	1	1	1
	X	X	X	X

z_2

$k_0 = 1$



Structure of a modulo 6 forward/backward counter using T flip flops

This counter shall be controlled by an input signal, r (for backwards) that counts up (r = 0) or counts down (r = 1). For its realization, T flip flops shall be used since for such a flip flop, it saves its old state if the input is 0 and toggles the state if the input is 1. The truth table can be found in the next page:

r	z_2	z_1	z_0	T_2	T_1	T_0
0	0	0	0	0	0	1
0	0	0	1	0	1	1
0	0	1	0	0	0	1
0	0	1	1	1	1	1
0	1	0	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	X	X	X
0	1	1	1	X	X	X
1	0	0	0	1	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	1
1	0	1	1	0	0	1
1	1	0	0	1	1	1
1	1	0	1	0	0	1
1	1	1	0	X	X	X
1	1	1	1	X	X	X

z_0

r				
			1	1
		X	X	
1		X	X	
		1		

z_1

z_2

$$T_2 = \bar{z}_0 \bar{z}_1 r + z_0 z_1 \bar{r} + z_0 z_2 \bar{r}$$

z_0

r				
			1	
		X	X	1
1		X	X	
1				

z_1

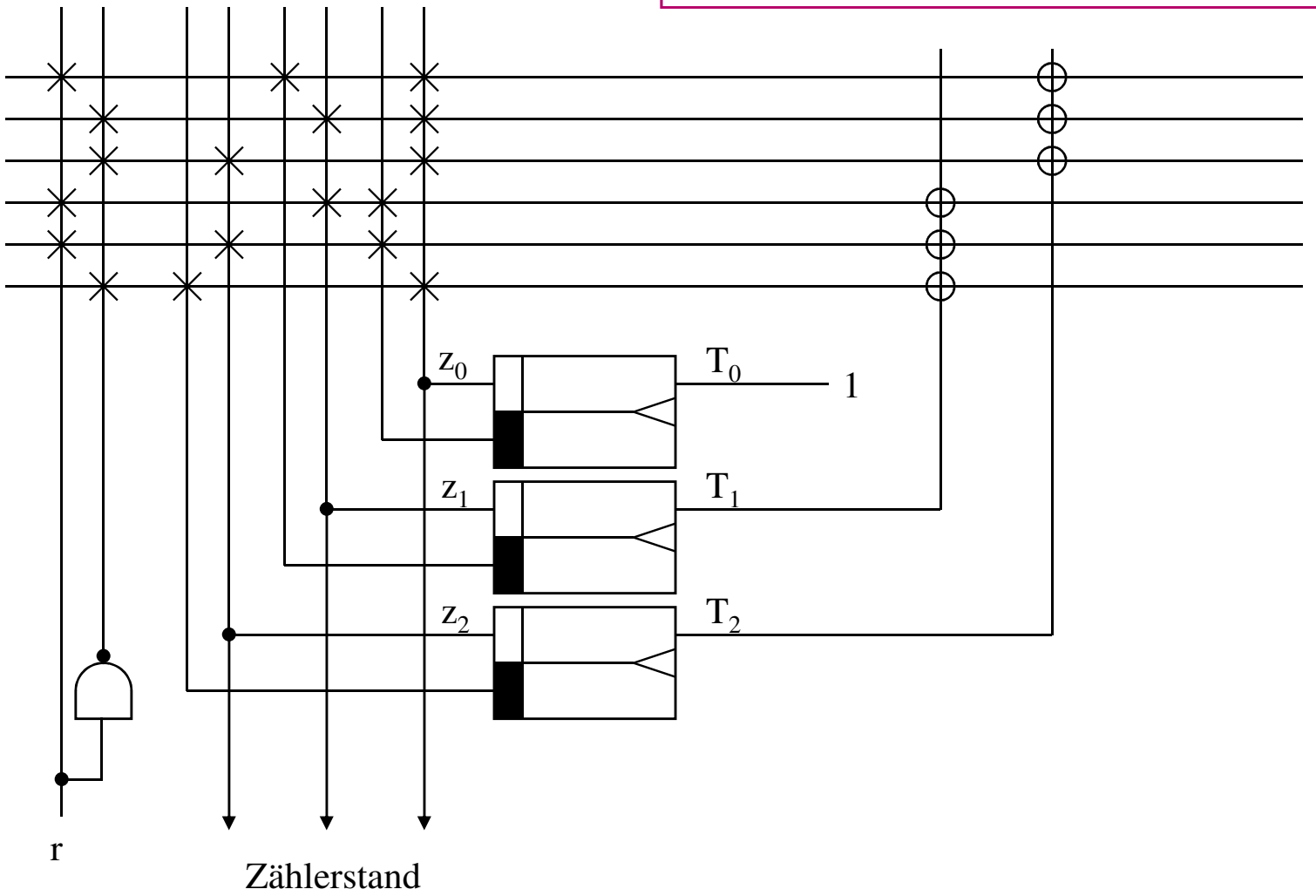
z_2

$$T_1 = \bar{z}_0 z_1 r + \bar{z}_0 z_2 r + z_0 \bar{z}_2 \bar{r}$$

$$T_0 = 1$$

Modulo-6-Counter

Realisation of FPLA:

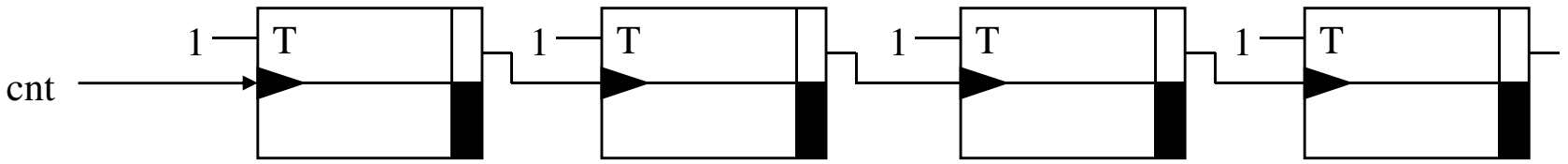


Asynchronous Counters

All sequential circuits that we have looked at so far were synchronous sequential circuits, especially if they were counters.

Definition: A sequential circuit is named a **synchronous** sequential circuit, if all clock inputs uses a single clock source.

There are many good reasons to drive a sequential circuit synchronously. Especially with counters, there are also many commonly used asynchronous types. It makes them special since they have a very simple structure. Let's take a look at the following modulo 16 counter using T flip flops:



This sequential circuit counts the number of impulses at the input cnt. Let the initial state of the counter be 0000 (read from left to right, just like the order of the flip flops shown in the drawing). The four flip flops are negatively edge triggered. After the first impulse of cnt, the first flip flop will be toggled. The counter value is 1000. All other flip flops did not see a negative edge at their clock input yet, hence they retain their states. At the next cnt impulse, the first flip flop changes its value again (this time from 1 to 0). Due to that, the second flip flop receives its first negative edge, it toggles and the counter value is now 0100. At the next cnt impulse, the counter changes to 1100 and so on.

We see that the binary numbers from 0 to 15 can be counted (represented in a mirrored manner). At the next impulse, all flip flops, one after another would change its states and the counter starts again at 0000.

Of course, such an asynchronous modulo N counter works the same for all to the power of 2 N with n = log N flip flops.