Computer Architecture and Organisation

Lecture 1

Bert Molenkamp

Learning objectives:

- You can design combinational logic
- You understand the principles of a synchronous sequential system

---

IEEE (EE) / "pearls" (CSC):

° Logic gates

° Boolean algebra

° Truth table

Today:

° Standard form (SOP form)

° Karnaugh-map

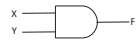
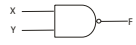
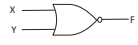
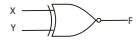
° Example combinational logic

- Truth table, Karnaugh-map, minimal SOP-form

° Asynchronous, Synchronous sequential logic

° Memory elements

Overview of binary functions of two variables

Name	Distinctive shape	Rectangular shape	Algebraic equation	Truth table
AND			$F = X \cdot Y$	X Y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR			$F = X + Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NAND			$F = \overline{A \cdot B}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR			$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)			$F = X \cdot \bar{Y} + \bar{X} \cdot Y$ $F = X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)			$F = X \cdot Y + \bar{X} \cdot \bar{Y}$ $F = \overline{X \oplus Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

3

Basic properties of Boolean algebra (more on page 438)

$A \cdot B = B \cdot A$	$A + B = B + A$	Commutative
$A(B + C) = A \cdot B + A \cdot C$	$A + B \cdot C = (A + B) \cdot (A + C)$	Distributive
$1 \cdot A = A$	$0 + A = A$	
$A \cdot \bar{A} = 0$	$A + \bar{A} = 1$	
$0 \cdot A = 0$	$1 + A = 1$	
$A(B \cdot C) = (A \cdot B)C$	$A + (B + C) = (A + B) + C$	Associative
$\overline{A \cdot B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \cdot \bar{B}$	DeMorgan (often used)

DeMorgan theorem also for more than 2 variables, e.g.

$\overline{A \cdot B \cdot C} = \bar{A} + \bar{B} + \bar{C}$ $\quad$ $\overline{A + B + C} = \bar{A} \cdot \bar{B} \cdot \bar{C}$

In a design we use names, e.g. CLK. Therefore use explicity the '·' symbol: A·B (not AB).

4

Logic symbols:

not a	is also represented as $\overline{a}$, **a'**, **!a**, **#a**, $\neg a$
a and b	is also represented as **a.b**, **a&b**, $a \wedge b$
a or b	is also represented as **a + b**, $a \vee b$
a xor b	is also represented as $a \oplus b$, $a \veebar b$

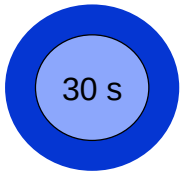
5

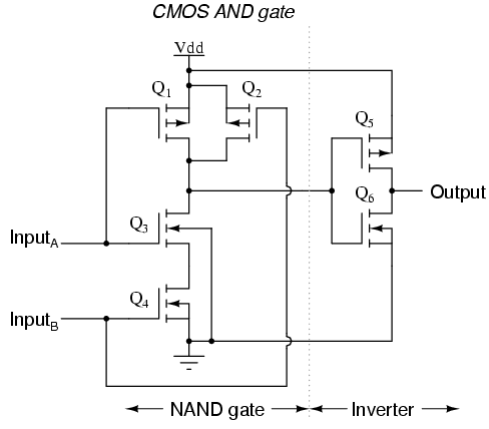
NAND/NOR gate faster/less area than AND/OR gate

$$F = a \bullet \overline{b} + c$$

Give a realization with only:

- **NAND gates**
- **NOR gates**





http://www.allaboutcircuits.com/vol_4/chpt_3/7.html
(not part of course)

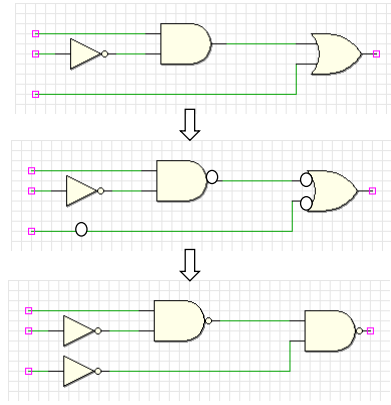
6

NAND (algebraic and schematic approach)

$$F = a.\bar{b} + c$$

$$F = \overline{\overline{a.\bar{b} + c}}$$

$$F = \overline{\overline{a.\bar{b}.\bar{c}}}$$



From output to input: replace with NAND gates using inverters.

An inverter is a NAND gate with input connected to eaxh other

Standard forms for combinational logic

° **Two standard forms:**

- **SOP; Sum-of-Products**
- **POS; Product-of-Sums (part of Digital Hardware (EE))**

° **SOP**

- **A sum of product terms; OR of AND terms.**
- **A variable in the AND term is true or its complement (NOT) is true.**
- **A minterm is a product term with exactly one instance of every variable**

$$F(A,B,C) = \bar{A} \cdot \bar{B} + \bar{B} \cdot C + \underbrace{A \cdot B \cdot \bar{C}}_{\text{minterm}}$$

$$F(A,B,C) = \underbrace{\overline{A.B}}_{} + A \cdot B \cdot \bar{C}$$

Not product term → not SOP form

More ‘complex’ components (Medium scale integration)

Often more complex components are used, e.g.:

° Multiplexer

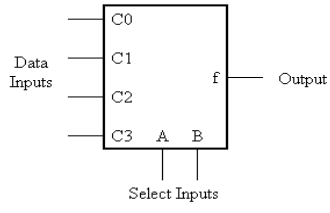
° Demultiplexer

° Decoder

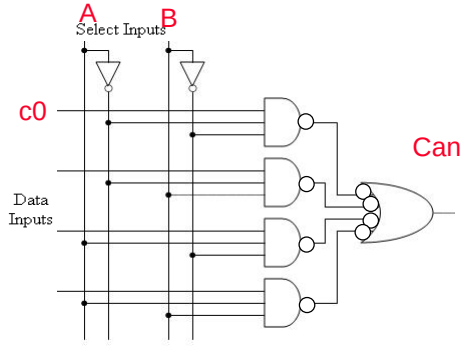
° Priority encoder

° Programmable logic (EE-students; “digital hardware”)

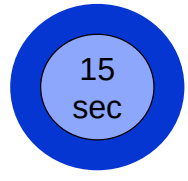
Multiplexer



The "select inputs" selects which input is forwarded to the output.

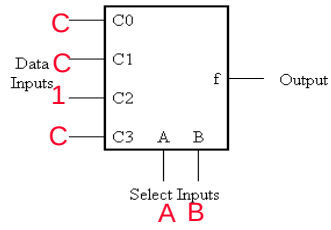


Can you realize this with NAND gates?



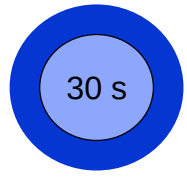
Can you realize this function with a multiplexer?

$$F = a \bullet \bar{b} + c$$



Mux

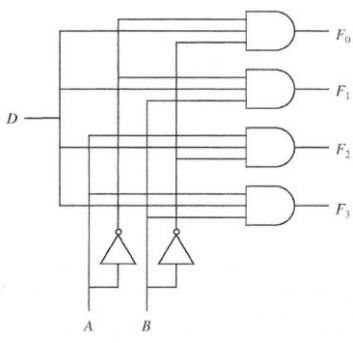
A	B	Output
0	0	C0
0	1	C1
1	0	C2
1	1	C3

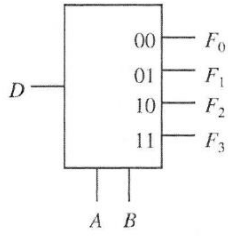


With a=1, b=0 F=1 → C2=1, other inputs c

11

Demultiplexer (inverse of multiplexer)





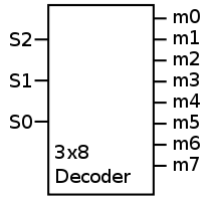
$F_0 = D\bar{A}\bar{B}$ $\quad F_2 = DA\bar{B}$

$F_1 = D\bar{A}B$ $\quad F_3 = DAB$

D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

12

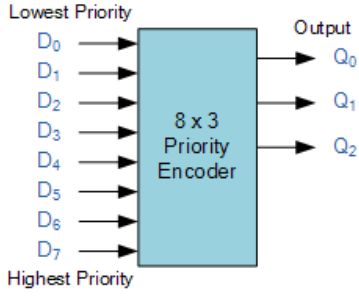
Decoder



- The select inputs (S) determines which output is 1 (other outputs 0).
- m<nr> is a minterm.
  E.g. Output m3 is 1 when s2=0, s1=1 and s2=1 ($011_2=3_{10}$)
- A decoder can also have an enable input (see book pag 451). When disabled all outputs are 0
- An application of a decoder: it can be used to extend the address range of a memory bank (CAO wk 3 & 4)

13

Priority encoder



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

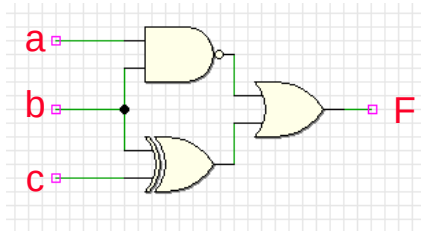
X = dont care

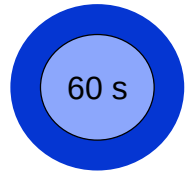
http://www.electronics-tutorials.ws/combination/comb_4.html

- The binary encoded output has the value of the first (D_7 .. D_0) input that is 1.
- A priority encoder can be used for arbitration between devices that compete for the same resource (CAO wk 3 & 4).

14

Give a simplified SOP-form for this circuit.





a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

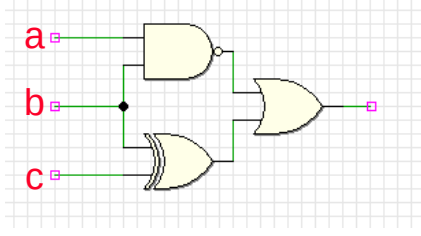
Find simplified SOP-form using:

- Algebraic reduction
- Karnaugh map reduction
- With truth table? Rarely!

$$\bar{F} = a.b.c$$
$$F = \overline{a.b.c}$$
$$F = \bar{a} + \bar{b} + \bar{c}$$

15

Algebraic reduction.



$F=\overline{\bar{A}.\bar{B}} + \bar{B}.C + B.\bar{C}$

$F=\bar{A} + \bar{B} + \bar{B}.C + B.\bar{C}$ DeMorgan

$F=\bar{A} + \bar{B}.1 + \bar{B}.C + B.\bar{C}$

$F=\bar{A} + \bar{B}.(C + \bar{C}) + \bar{B}.C + B.\bar{C}$

$F=\bar{A} + \bar{B}.C + \bar{B}.\bar{C} + \bar{B}.C + B.\bar{C}$

$F=\bar{A} + \bar{B}.C + \bar{B}.\bar{C} + B.\bar{C}$

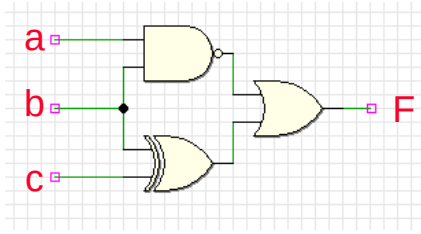
$F=\bar{A} + \bar{B}.C + \bar{B}.\bar{C} + \bar{B}.\bar{C} + B.\bar{C}$

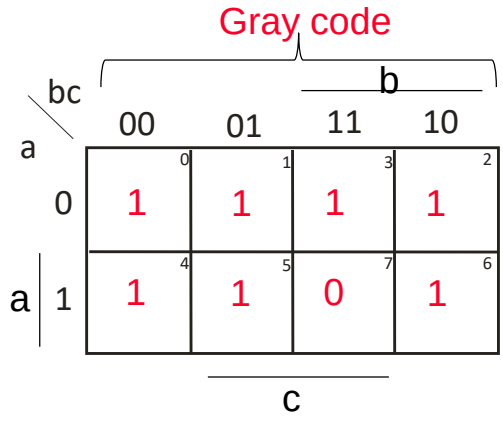
$F=\bar{A} + \bar{B}(C + \bar{C}) + (\bar{B} + B)\bar{C}$

$F=\bar{A} + \bar{B} + \bar{C}$

16

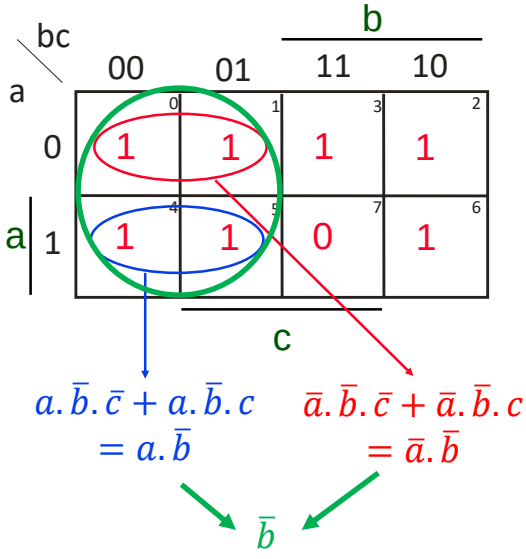
Karnaugh.





17

Karnaugh.



Group adjacent cells!
2 cells → 1 variable less
4 cells → 2 variables less
8 cells → 3 variables less

Adjacent cells may not include 0

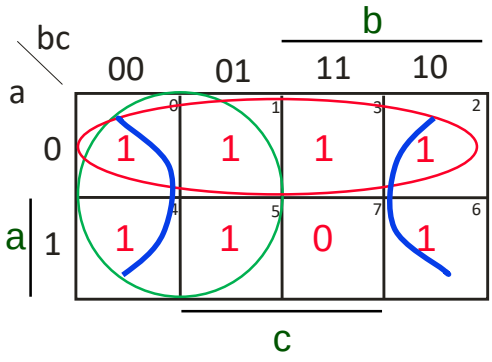
Adjacent cells are horizontal/vertical, not diagonal

Each group of adjacent cells should be as large as possible

Groups may overlap

18

Karnaugh.



$$F = \bar{a} + \bar{b} + \bar{c}$$

19

Example

° **F = 1 for the prime numbers in the range 1 to 9**

° **F = 0 for non-prime numbers in the range 1 to 9**

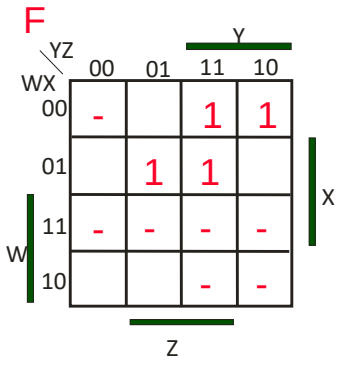
YZ / WX	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

YZ / WX	00	01	11	10
00	d	0	1	1
01	0	1	1	0
11	d	d	d	d
10	0	0	d	d

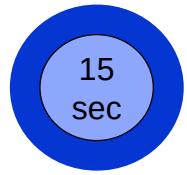
$F(W,X,Y,Z) = \sum m(2,3,5,7)$ $d = \sum m(0,10,11,12,13,14,15)$

20

What is the SOP-solution?



$F = \overline{X} \cdot Y + X \cdot Z$



For fields that are don't care often 'd', 'x' or '-' is used

21

Combinatorial <-> Sequential

Combinational logic

° **Has no memory**

° **The outputs are functions of the current inputs.**

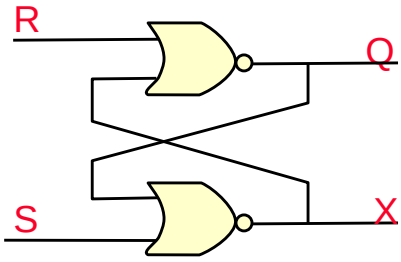
Sequential logic

° **Can store data; has memory**

° **The outputs are functions of the current inputs and the past.**

22

SR-latch with NORs



$Q = \overline{R + \overline{(S+Q)}}$

$Q = \overline{R} \bullet (S+Q)$ (Reset dominant)

$X = \overline{S + Q}$

$X = \overline{S} \bullet \overline{Q}$

- Output Q is dominated by the Reset
- Output X is the inverse of Q, except when S=1, R=1, because the Set dominates X, hence X=0 (S=1, R=1 is often indicated as 'disallowed').
- X is often indicated with $\overline{Q}$, although this is not correct
- An SR-latch with cross coupled NAND gates is also possible

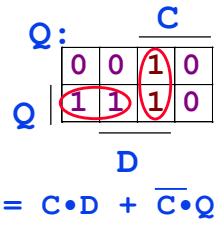
23

D-latch 1

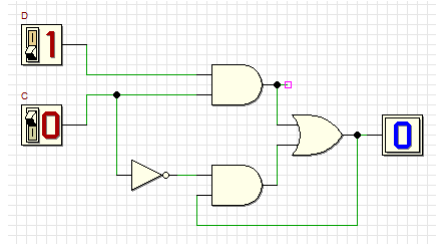
- D-latch has two inputs, C (Control) and D (Data) and output Q
- When C=0 then Q=Q (remember)
- When C=1 then Q=D (follow input D)

C	D	Q
0	0	Q
0	1	Q
1	0	0
1	1	1

Q	C	D	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



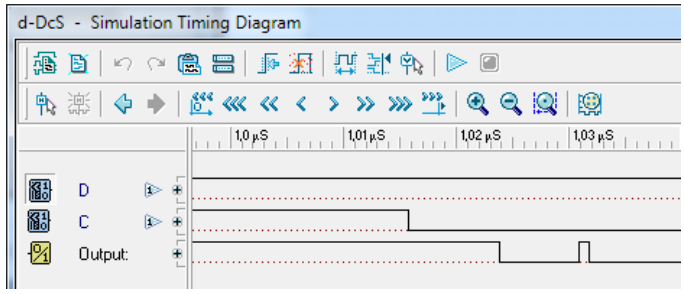
$Q = C \bullet D + \overline{C} \bullet Q$



24

D-latch 2

- The D-latch has a problem when **D=1** and a there is a transition from **C=1** to **C=0**
- The output **Q** should remain **1**, but due to the delays of the components the output is not what is expected.
- Beneath a simulation result with DEEDS (software on Canvas).
- Solution: add an extra, seemingly unnecessary, product term (**Q•D**) in the solution



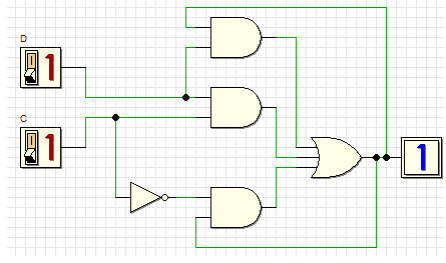
25

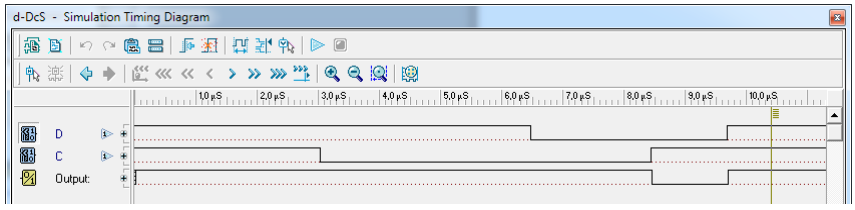
D-latch 3

Q:

Q \ C,D	00	01	11	10
0	0	0	1	0
1	1	1	1	0

$$Q = C \bullet D + \overline{C} \bullet Q + D \bullet Q$$





26

Asynchronous sequential circuits

° Memory is created with feedback

° This is called an asynchronous sequential circuit

° The signals fed back represent the state of the system.

° In principle any sequential circuit can be made this way. (No explicit memory elements)

Problems of an asynchronous circuit

° It is very difficult to design reliable asynchronous circuits.

° Some causes are:

- Differences in delay times. Not all bits of the state change at the same time. As a consequence arbitrary states can occur.
- No synchronization: when is data ”arrived”?

° Solution

- Use explicit memory elements with a clock input
- The memory devices are synchronized: they are all connected to a common clock, and the outputs can only change after a rising edge or falling edge of the clock (not on both edges of the clock)
- The clock frequency is chosen sufficient low, such that all signals in the combinational logic are stable

° These circuits are called synchronous sequential circuits (abbreviated as synchronous circuits)

Synchronous sequential circuits

° Almost all the larger digital circuits are synchronous sequential circuits; asynchronous techniques are rarely used.

° Synchronously sequentially actually means that the operation of parts of the circuit is delayed in such a way that even the slowest path into the circuit under the most unfavorable conditions of time, has sufficient time to stabilize before the next active edge of the clock.

- This path is called “Critical path”

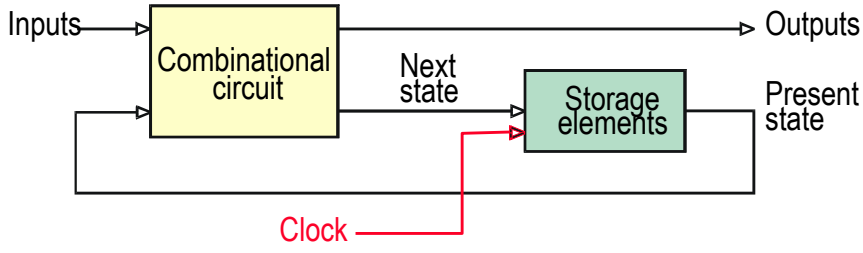
° All synchronous sequential circuits can be described using the following model.

Model of a synchronous sequential system

° The current Outputs (O) are a function of the current Inputs (I) and the Present State (S): $I \times S \rightarrow O$

° The Next State is a function of the current Inputs and the Present State: $I \times S \rightarrow S$

° This model is called: Mealy machine (output also depends on current inputs).



Latches? flip-flops!

° Are D-latches suitable as memory element in this model?

° No!

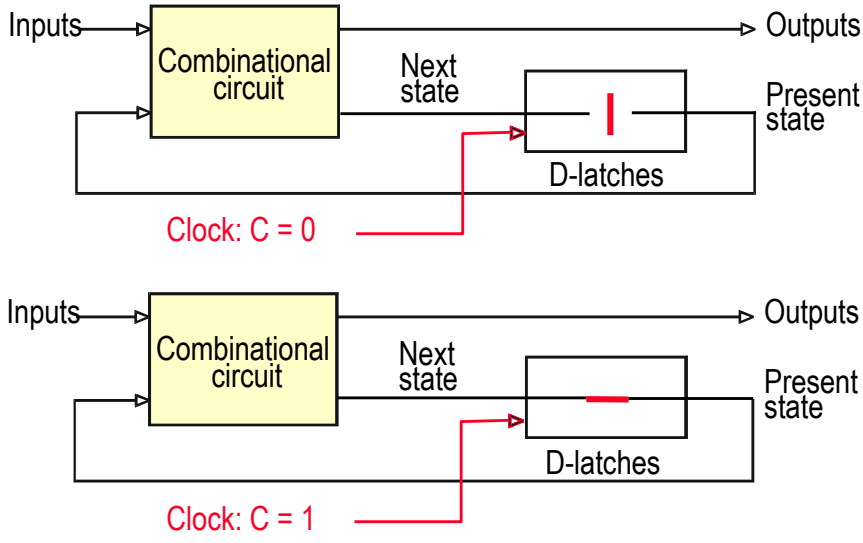
- As long as `C=1` the latches behave as “wires”.
- Only if `C` is `1` for a short moment it could work; this is too critical

° Solution:

- Do not use latches, use flip-flops. Flip-flops only change on an edge of the clock.

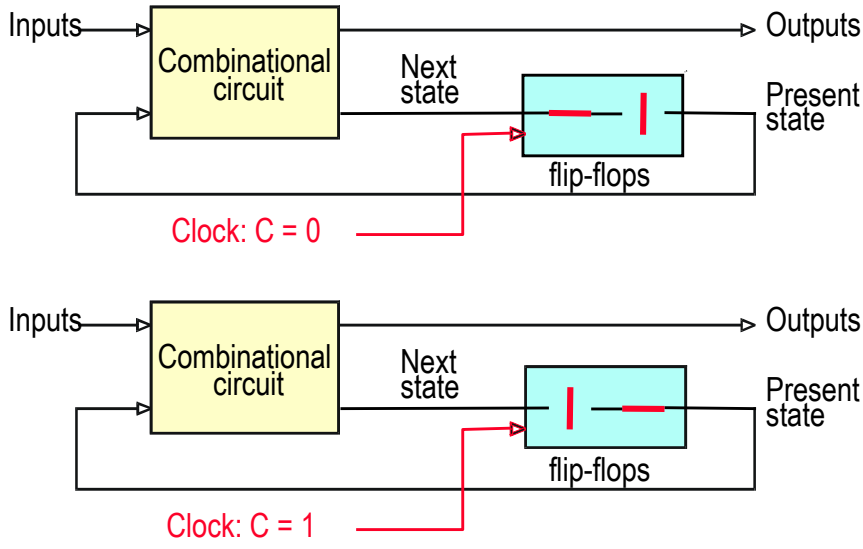
31

Problem of D-latches



32

Use flip-flops



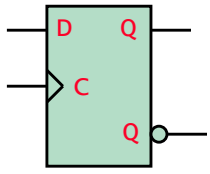
flip-flops

- Edge sensitive clock input `C`, has a small triangle
- Active edge is rising edge of clock, unless the input has a circle (then the active edge is the falling edge)
- Clock input does not appear in descriptions and tables
- Current state is in tables denoted by `Q(t)` or `Q`; next state with `Q(t+1)` or `Q+`
- By far most commonly used is the D flip-flop; in addition also DE flip-flop, SR flip-flop, JK flip-flop and T flip-flop
- Output `Q` changes only on the active edge of the clock

D flip-flop

D	Q+	
0	0	Reset
1	1	Set

Next state is independent of current state

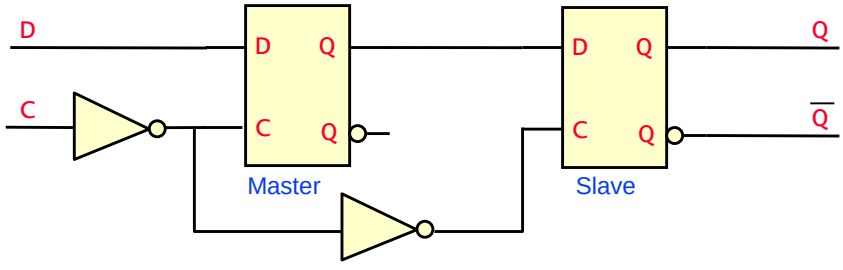


35

D flip-flop realized with two latches

° Master-Slave principle

° In practice an other implementation is used with the same behaviour but faster and smaller (not part of this course).

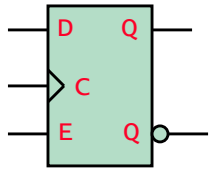


36

DE flip-flop

° DE flip-flop is D flip-flop with enable **E**; when **E=0** flip-flop output is unchanged. When **E=1** flip-flop behaves as D flip-flop

E	D	Q+	
0	0	Q	Unchanged
0	1	Q	Unchanged
1	0	0	Reset
1	1	1	Set



Q	E	D	Q+	
0	0	0	0	Unchanged
0	0	1	0	Unchanged
0	1	0	0	Reset
0	1	1	1	Set
1	0	0	1	Unchanged
1	0	1	1	Unchanged
1	1	0	0	Reset
1	1	1	1	Set

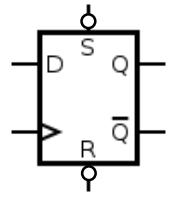
Asynchronous reset (R) and set (S)

° Flip-flops often have asynchronous clear (**R** or **CLRN**) and preset (**S** or **PRN**) inputs.

° These inputs are typically active low (indicated with a bubble at the input) and independent of the clock.

° These inputs can set or reset the flip-flop regardless of the status of the clock.

° These inputs are rarely used during normal operation, they are used to initialize the flip-flop.



Timing

° A D flip-flop itself is an asynchronous sequential circuit.

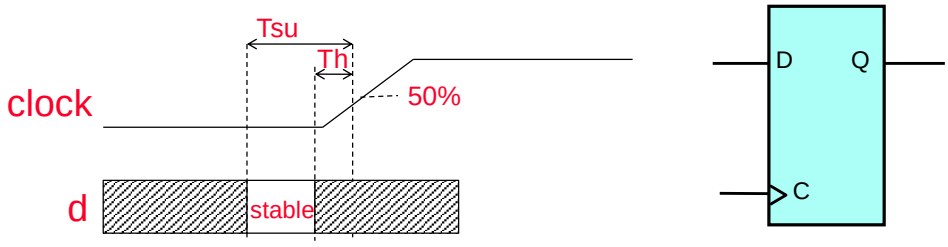
° Hence the behaviour can also be unpredictable if timing constraints are not met (for EE students more detail in “Digital hardware”).

° Important timing constraints are:

- Tsu: Setup time is the minimum amount of time the data signal should be held steady before the active edge of clock.
- Th: Hold time is the minimum amount of time the data signal should be held steady after the active edge of clock

° Tco: The *clock to output delay* is the time required to obtain a valid output after the active edge of the clock.

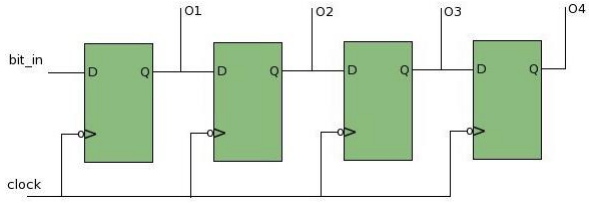
Example

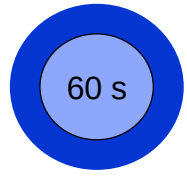


° A more realistic waveform of the transition of the clock edge.

° In this example the hold time is negative.

Example: shift register





° What is active edge of the clock?

° T_{su} = 2ns, T_h = 1ns, $T_{propagation\ delay\ wires}$ = 0ns

- When must the input *bit_in* be stable?
- What is the minimum T_{co} for a correct operation?
- What is the maximum clock frequency (with minimum T_{co})?

a) bit_in stable: 2ns before falling edge of clock until 1 ns after falling edge of clock
b) Input D of flipflop should be stable at least 1 ns after active edge → Tco ≥ 1ns
c) Max clock freq = 1 /(Tsu + Tco) = 1/ (3 ns) ~ 333 MHz

41

° **What did you learn:**

- **You can design combinational logic**
- **You understand the principles of a synchronous sequential system**

° **Next lecture**

- **Finite state machines**

42