Industrial Automation
MKT4152

Midterm Results Review

Results Review

MIDTERM EXAMINATION

Question 1

(1) You are given the following function equation:

$$Y = Y1 \vee Y2 = \overline{(A} \wedge \bar{B}) \vee (A \wedge B)$$

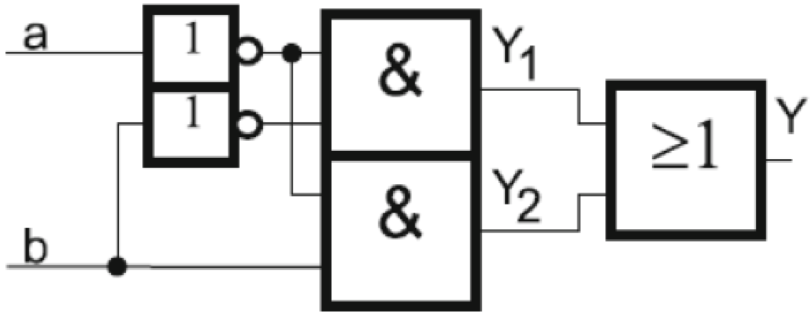
(1a) create the function chart for above function [6points]

(1b) create logic plan using function blocks [6points]

Question 2

(2a) Create the function chart [6points]

(2b) Create the function equation [6points]



Question 3

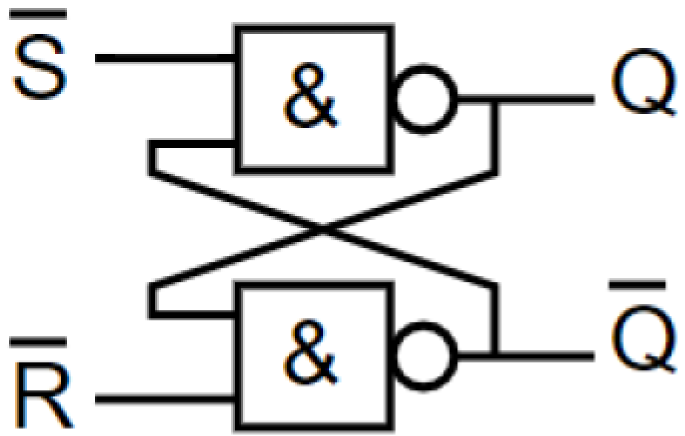
(3a) Draw the logic plan for a **RS-Flip-Flop** using basic elements such as AND, OR, NOT. [10points]

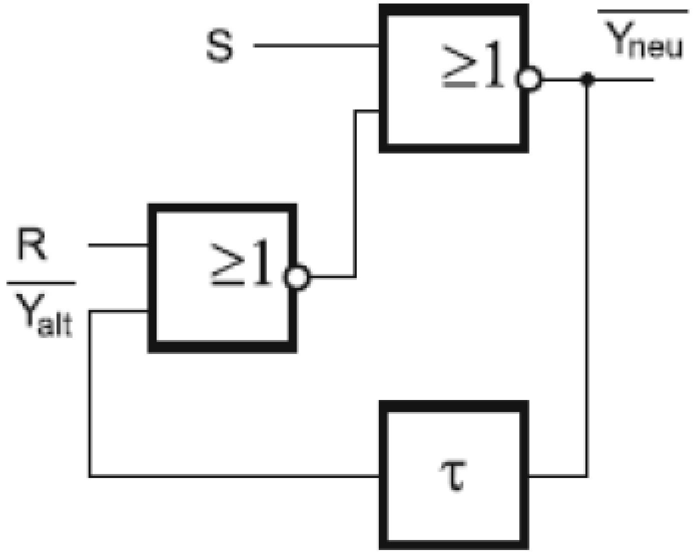
(3b) Create the function chart for 3a [8points]

(3c) What is special about both Set and Reset inputs equal to TRUE? [4points]

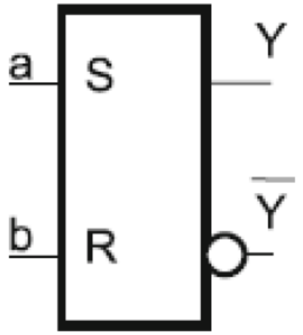
(3d) How can you modify your RS FLIP FLOP for DOMINANT RESET behavior? Draw a logic diagram, again using AND, OR, NOT elements only. [6points]

Question 3

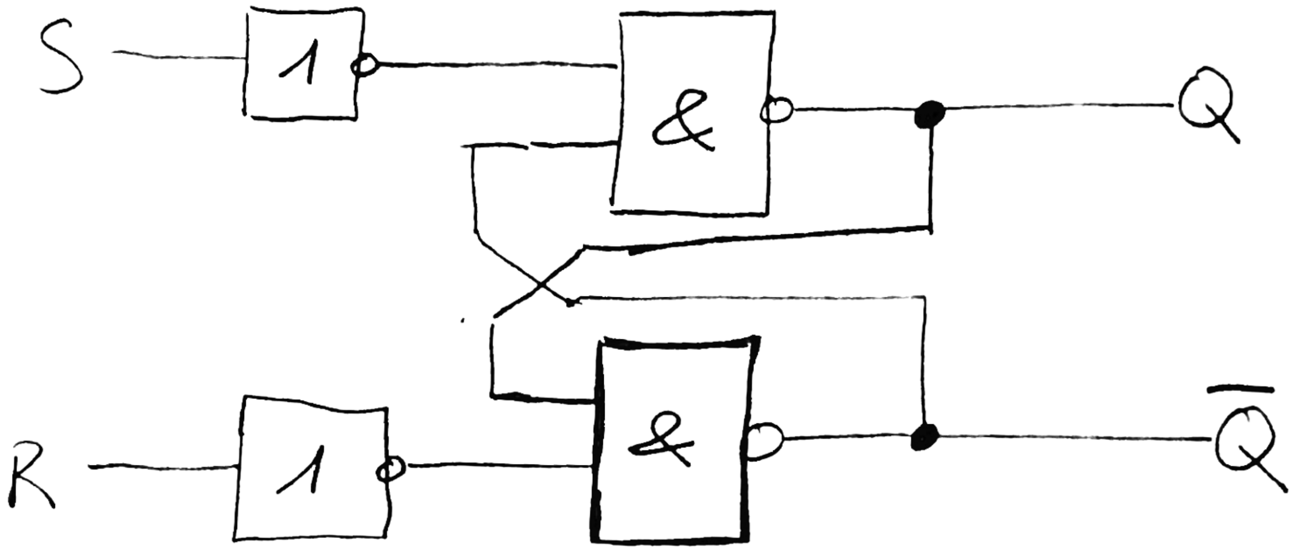




a	b	Y
0	0	Y (no change)
1	0	1 (set)
0	1	0 (reset)
1	1	?? forbidden



Bistable Gates - RS Flip Flop



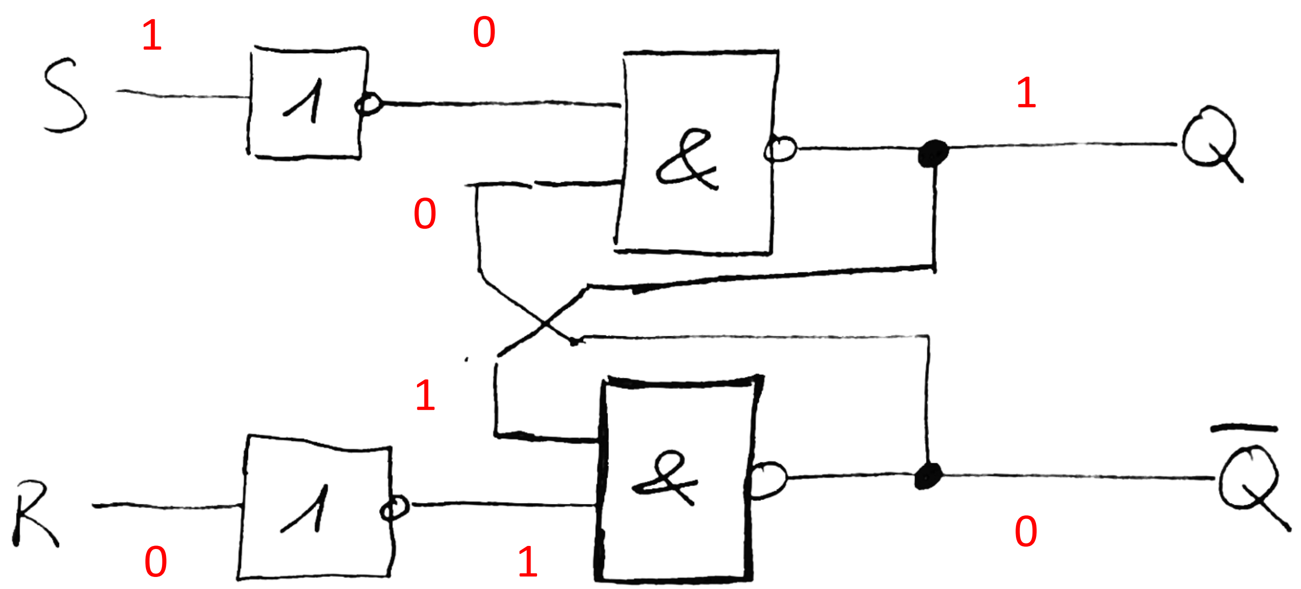
NAND Gate

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

RS Flip Flop

S	**R**	**$\underline{S}$**	**$\underline{R}$**	**Q**
0	0	1	1	**no change**
1	0	0	1	**1**
0	1	1	0	**0**
1	1	0	0	**--**

Bistable Gates - RS Flip Flop - SET



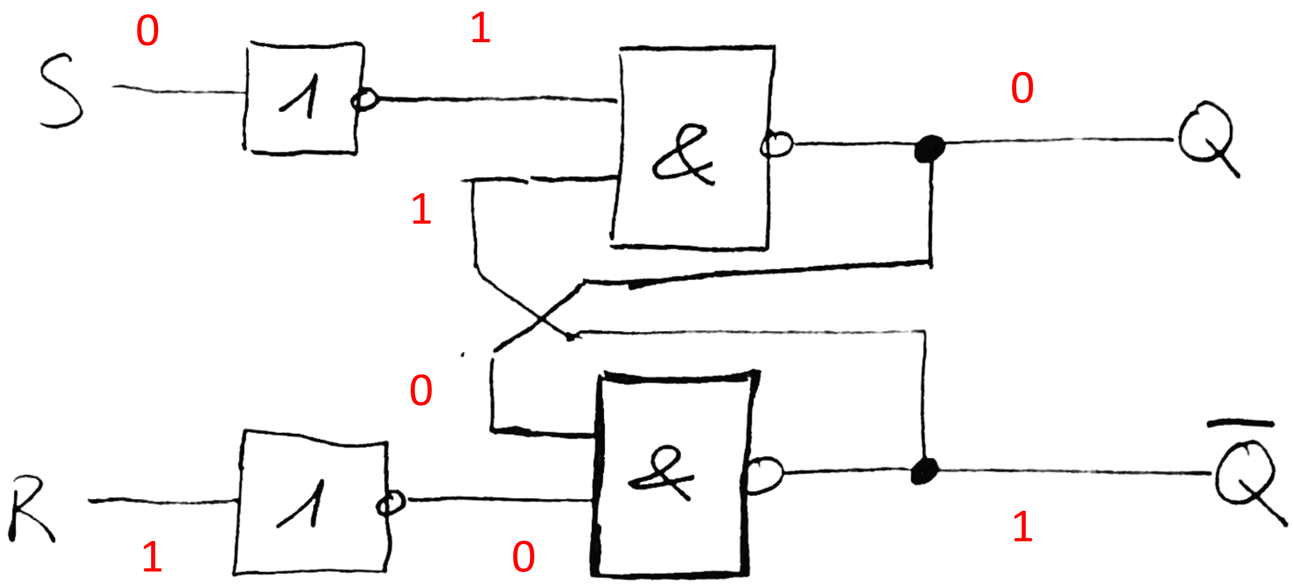
NAND Gate

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

RS Flip Flop

S	**R**	**S**	**R**	**Q**
0	0	1	1	**no change**
1	0	0	1	**1**
0	1	1	0	**0**
1	1	0	0	**--**

Bistable Gates - RS Flip Flop - RESET



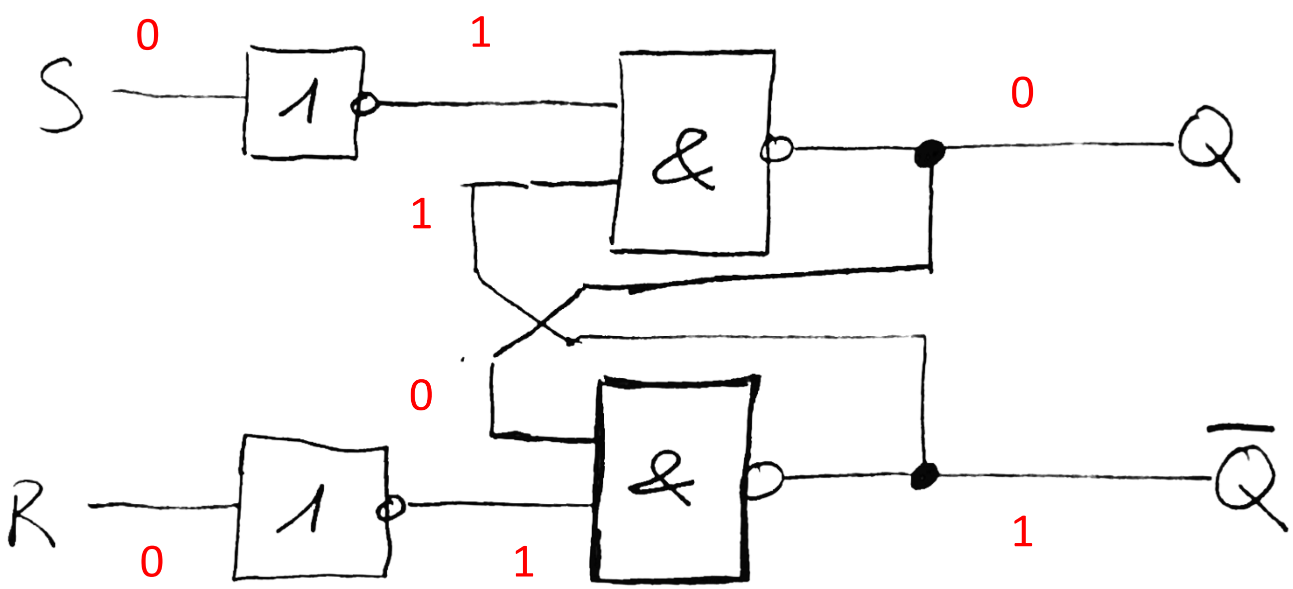
NAND Gate

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

RS Flip Flop

S	**R**	**S**	**R**	**Q**
0	0	1	1	**no change**
1	0	0	1	**1**
0	1	1	0	**0**
1	1	0	0	**--**

Bistable Gates - RS Flip Flop – No Change Situation 1



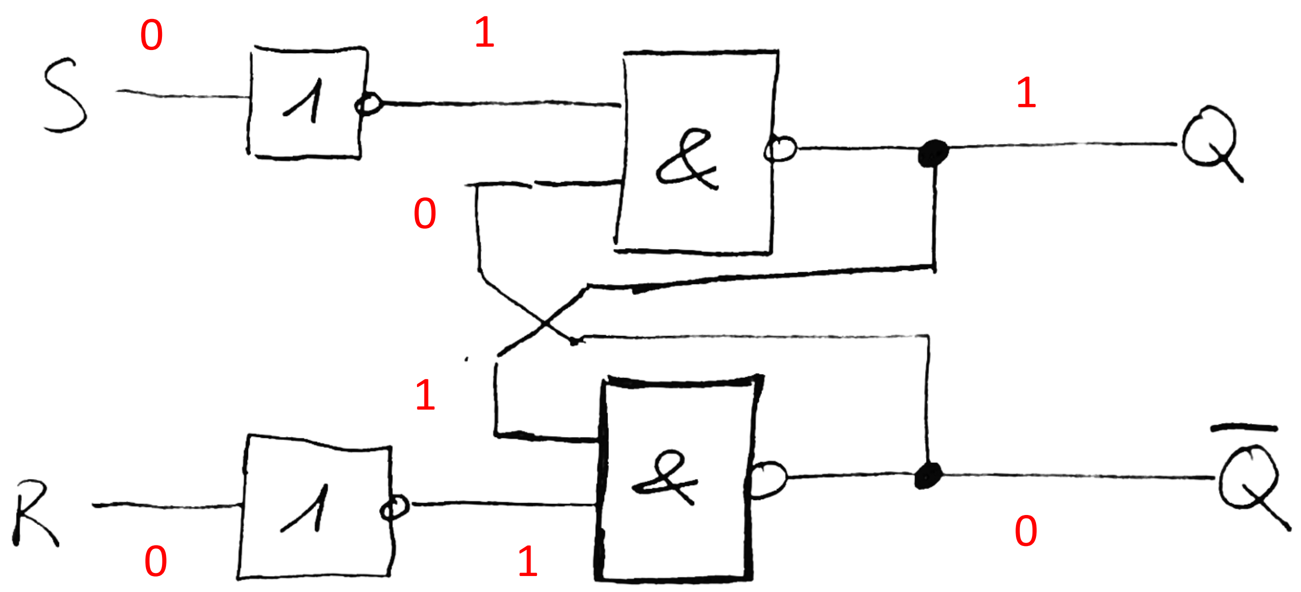
NAND Gate

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

RS Flip Flop

S	R	$\underline{S}$	$\underline{R}$	**Q old 1**	**Q new 1**	**Q old 2**	**Q new 2**
0	0	1	1	**0**	**0**	**1**	**1**
1	0	0	1	1	1	1	1
0	1	1	0	0	0	0	0
1	1	0	0	--	--	--	--

Bistable Gates - RS Flip Flop – No Change Situation 2



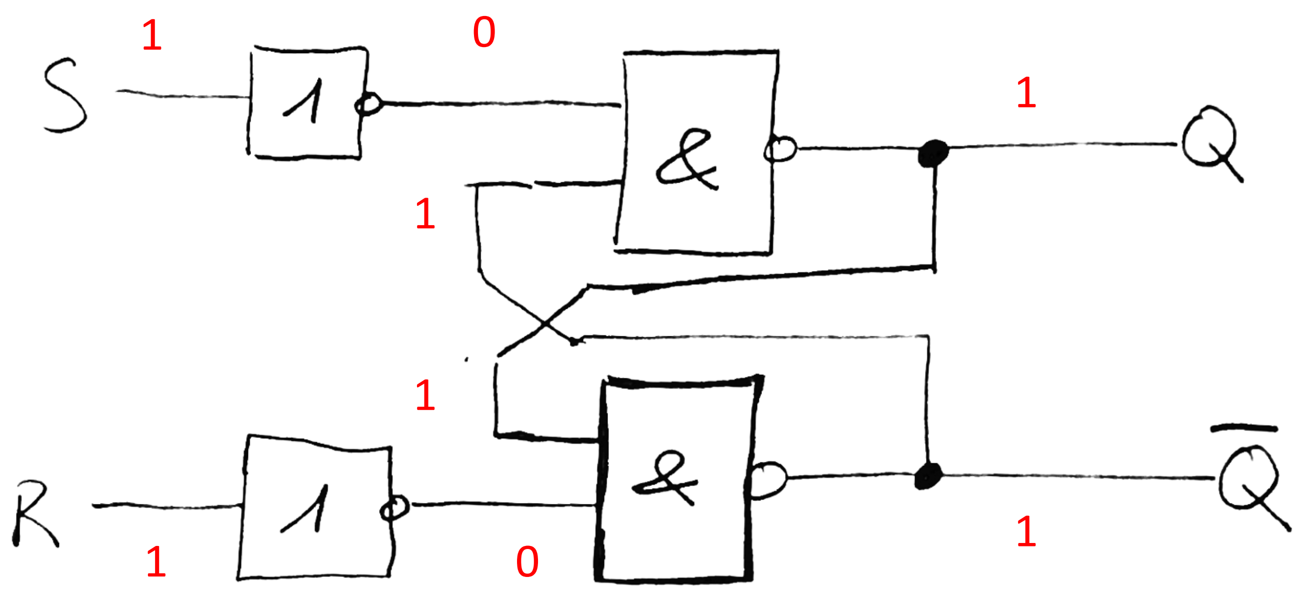
NAND Gate

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

RS Flip Flop

S	R	$\underline{S}$	$\underline{R}$	**Q old 1**	**Q new 1**	**Q old 2**	**Q new 2**
0	0	1	1	**0**	**0**	**1**	**1**
1	0	0	1	1	1	1	1
0	1	1	0	0	0	0	0
1	1	0	0	--	--	--	--

Bistable Gates - RS Flip Flop – !!!S=R=1!!!



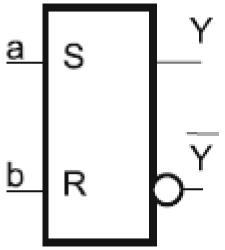
NAND Gate

A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

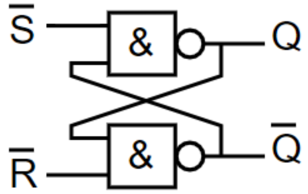
Why is this a problem?
-> We don’t exactly know what will happen after,
In the following step. That is what is not defined.

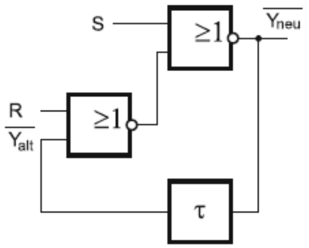
Bi-Stable Gates: RS & SR Flip Flop

RS-FLIP FLOP

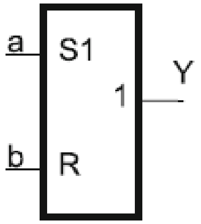


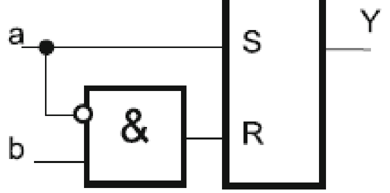
a	b	Y
0	0	Y (no change)
1	0	1 (set)
0	1	0 (reset)
1	1	?? forbidden





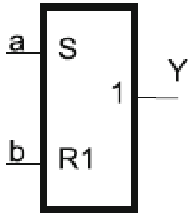
RS-FLIP FLOP – Dominant Set

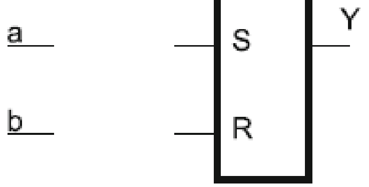




a	b	Y
0	0	Y
1	0	1 (Setzen)
0	1	0 (Rücksetzen)
1	**1**	**1 (Priorität)**

RS-FLIP FLOP – Dominant Reset

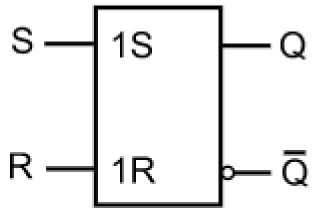


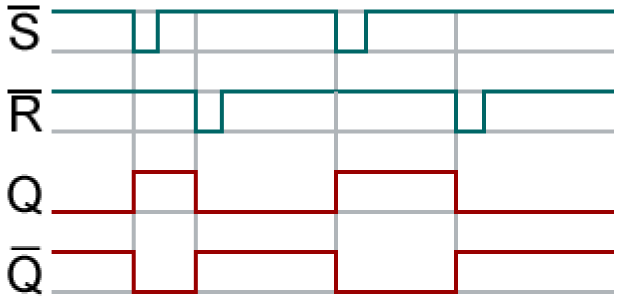


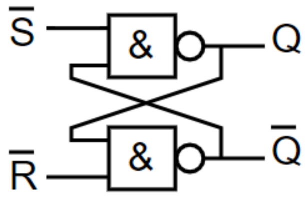
a	b	Y
0	0	Y
1	0	1 (Setzen)
0	1	0 (Rücksetzen)
1	**1**	**0 (Priorität)**

Trigger Activated Flip Flop I

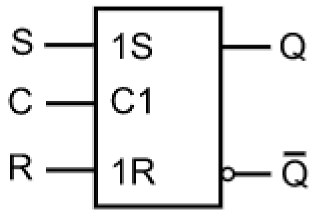
Asynchronous RS Flip Flop

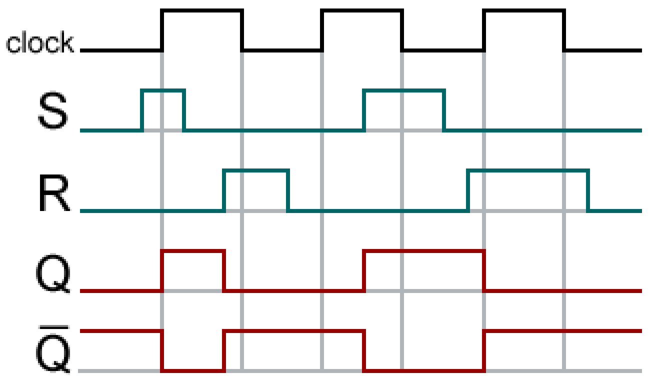


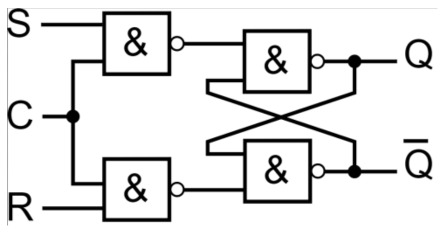




Synchronous RS Flip Flop







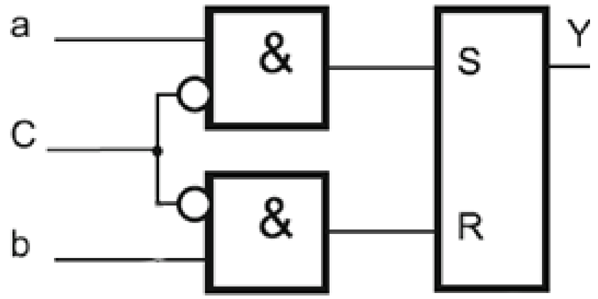
Trigger Activated Flip Flop II

Triggered Flip Flops are operational as long as control input C is active.
This might not always desired. It might be necessary to allow only ONE change in its output state each time the control signal becomes active.

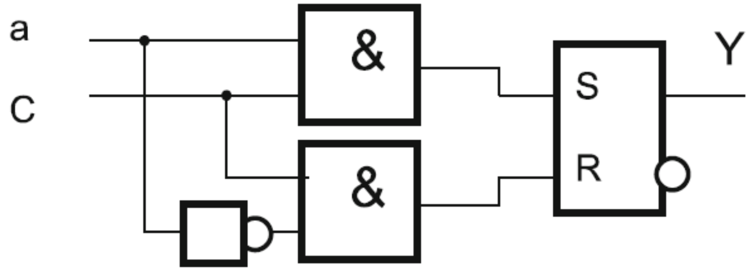
Question:
How can we realize a Flip-Flop, that changes its state only with rising edges, or falling edges?

RS Flip Flop with Dynamic Input

Synchronous RS Flip Flop

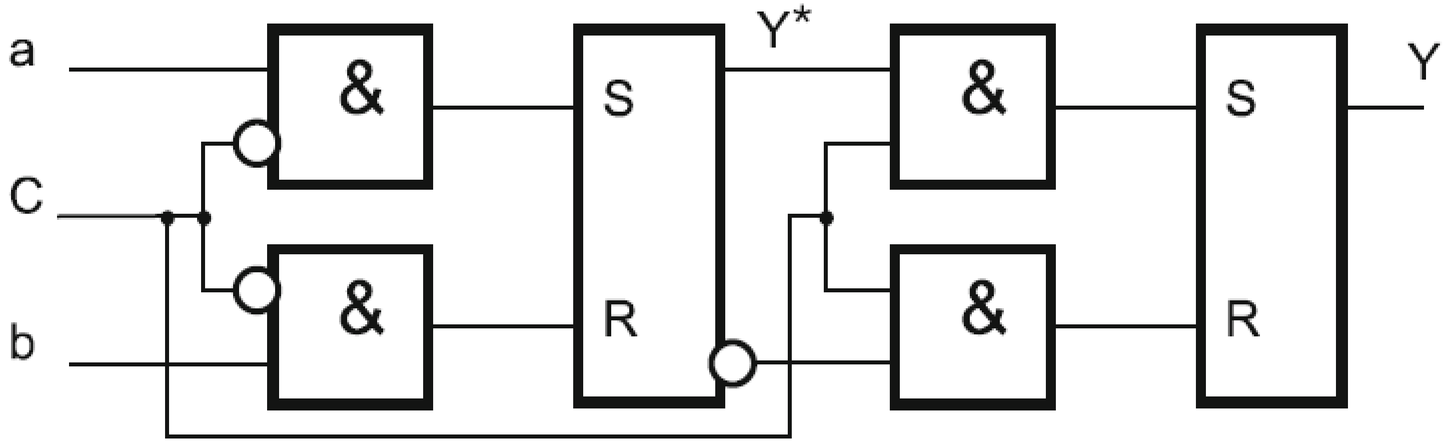


Triggered Memory



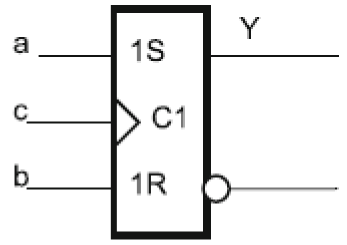
Pre-Storage

Post-Storage



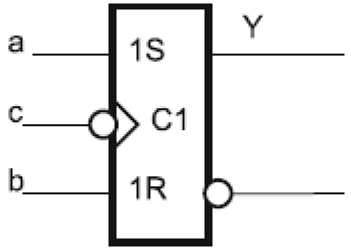
RS Flip Flop with Dynamic Input

Triggered on rising edge



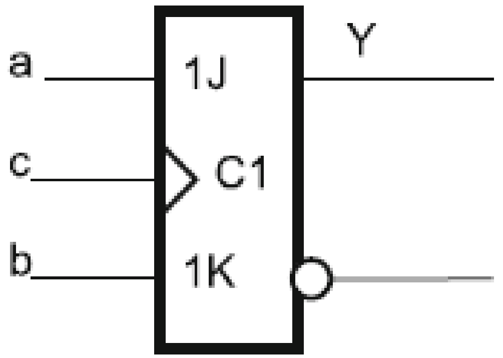
a	b	Y vor Flanke	Y nach Flanke
0	0	No change	
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	forbidden	

Triggered on falling edge



a	b	Y vor Flanke	Y nach Flanke
0	0	No change	
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	forbidden	

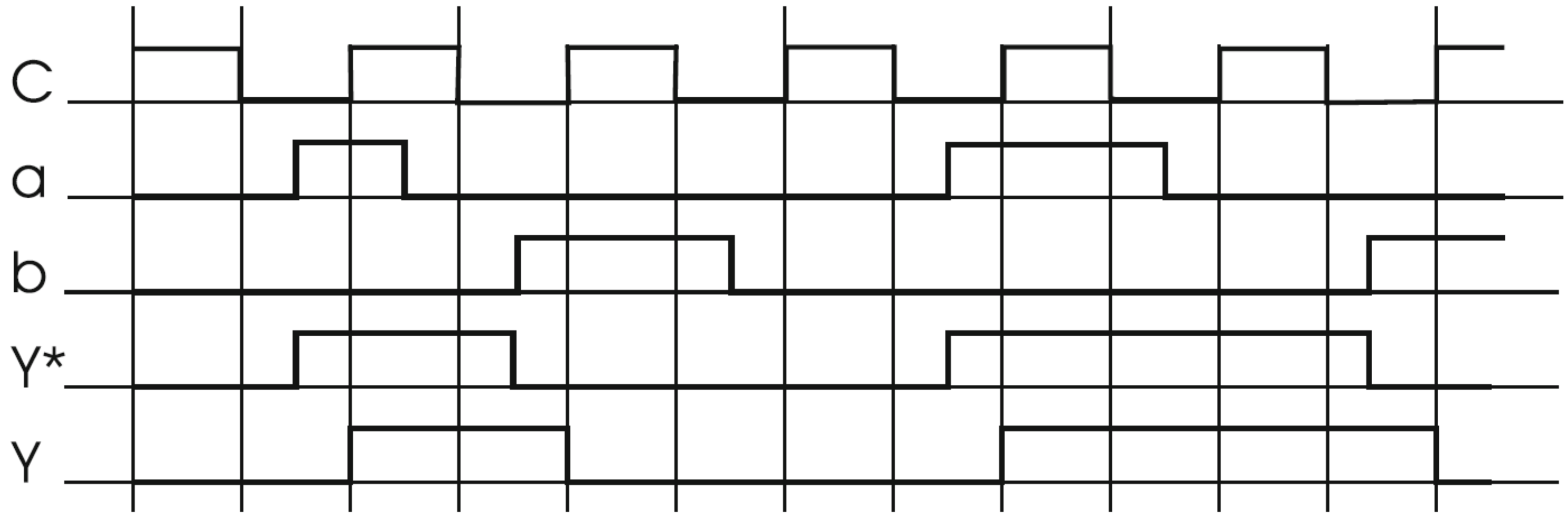
JK FLIP FLOP



a	b	Y vor Flanke	Y nach Flanke
0	0	No change	
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	**0**	**1**
1	1	**1**	**0**

JK Flip Flop
with a AND b, Y changes its state with each trigger edge!!!

RS Flip Flop with Dynamic Input



Remember:

RS FLIP FLOP with Dynamic Input:

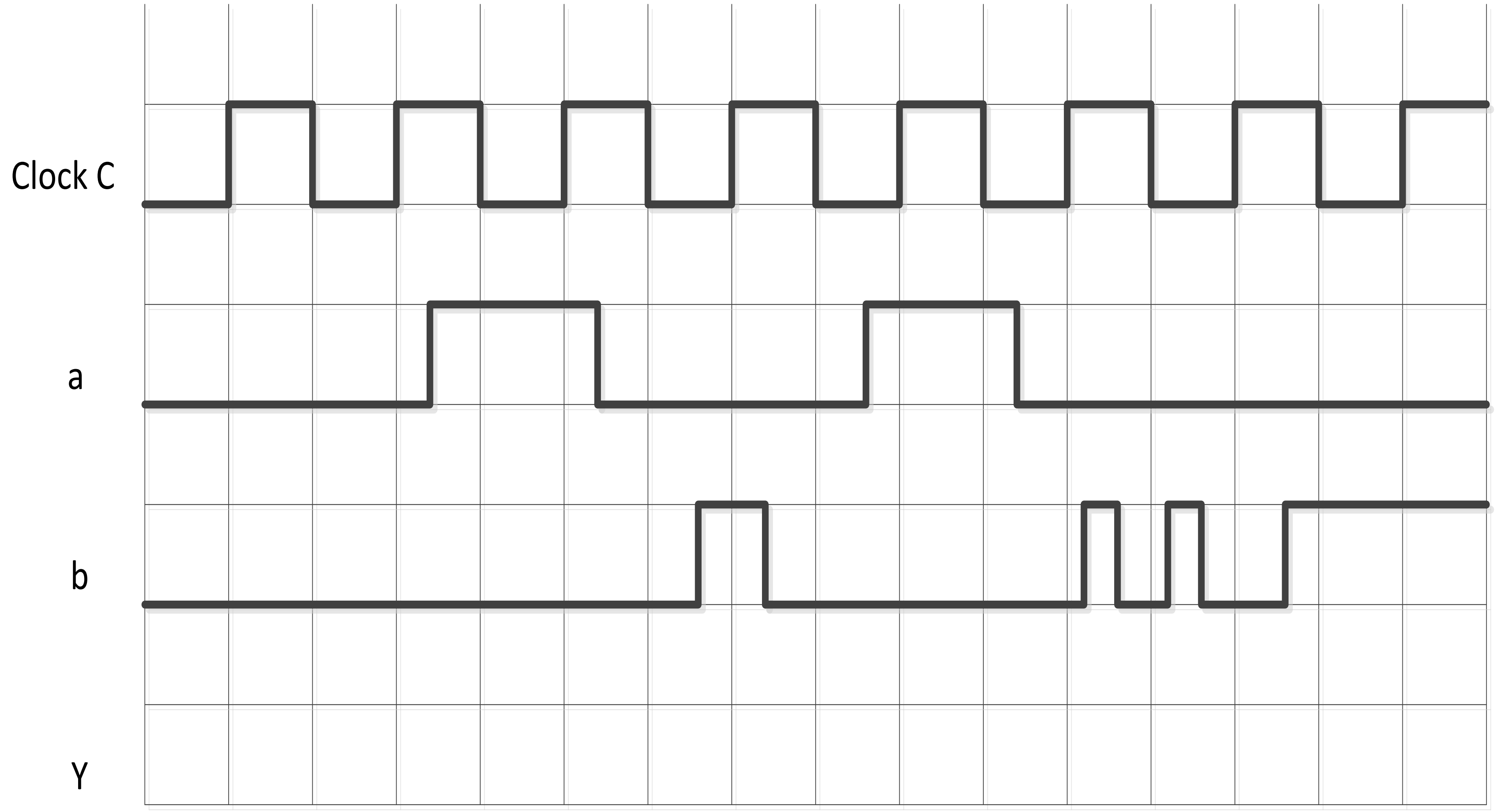
(A AND B) simultaneously true is not defined!!!!

J-K-FLIP FLOP:

(A AND B) simulataneously true causes change of state with each rising edge

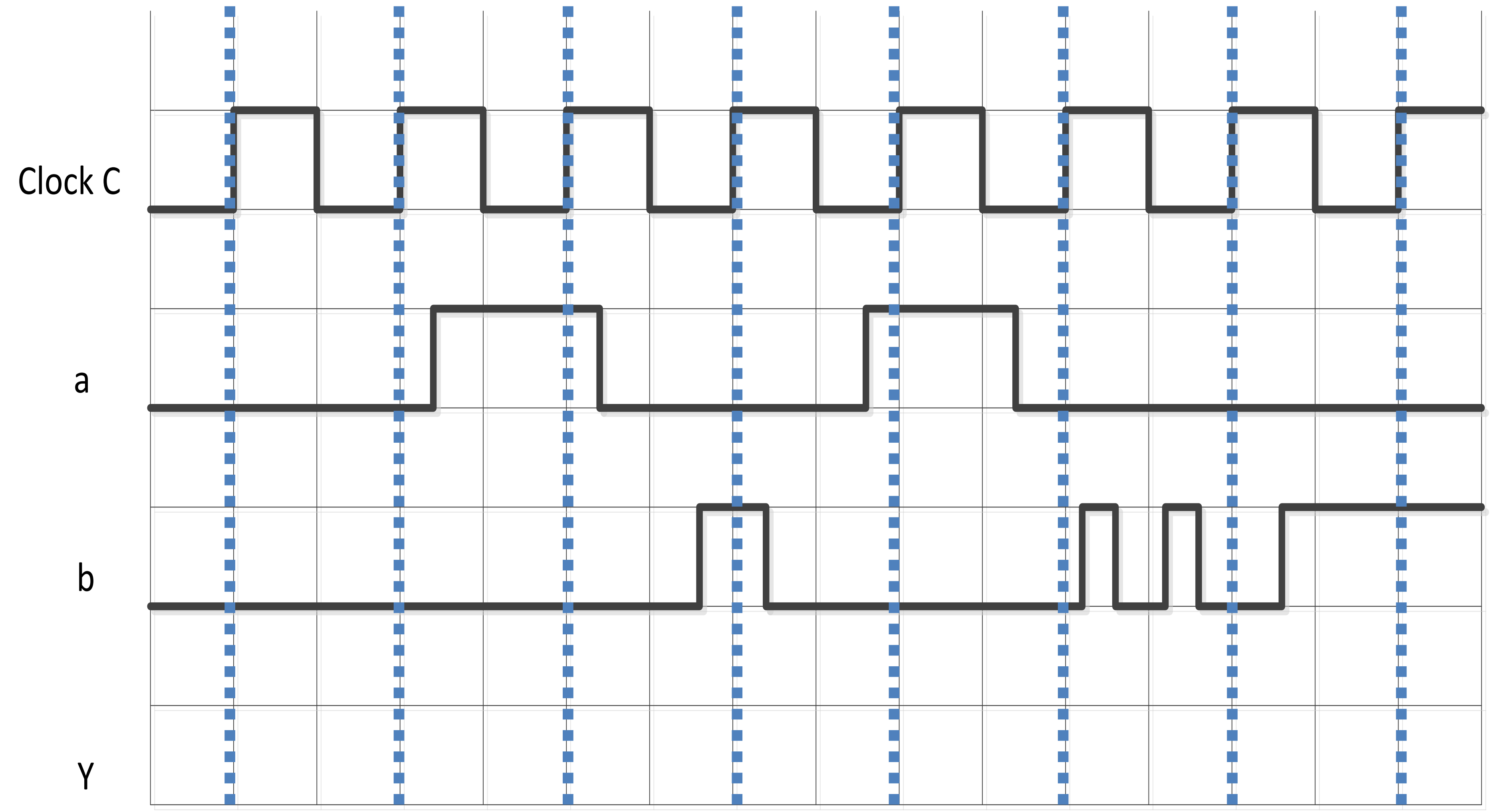
Question 4

(4) You are given a JK-Flip-Flop (**RS Flip Flop with dynamic input**), as shown in below picture. Complete the timing diagram for output Y. [10points]



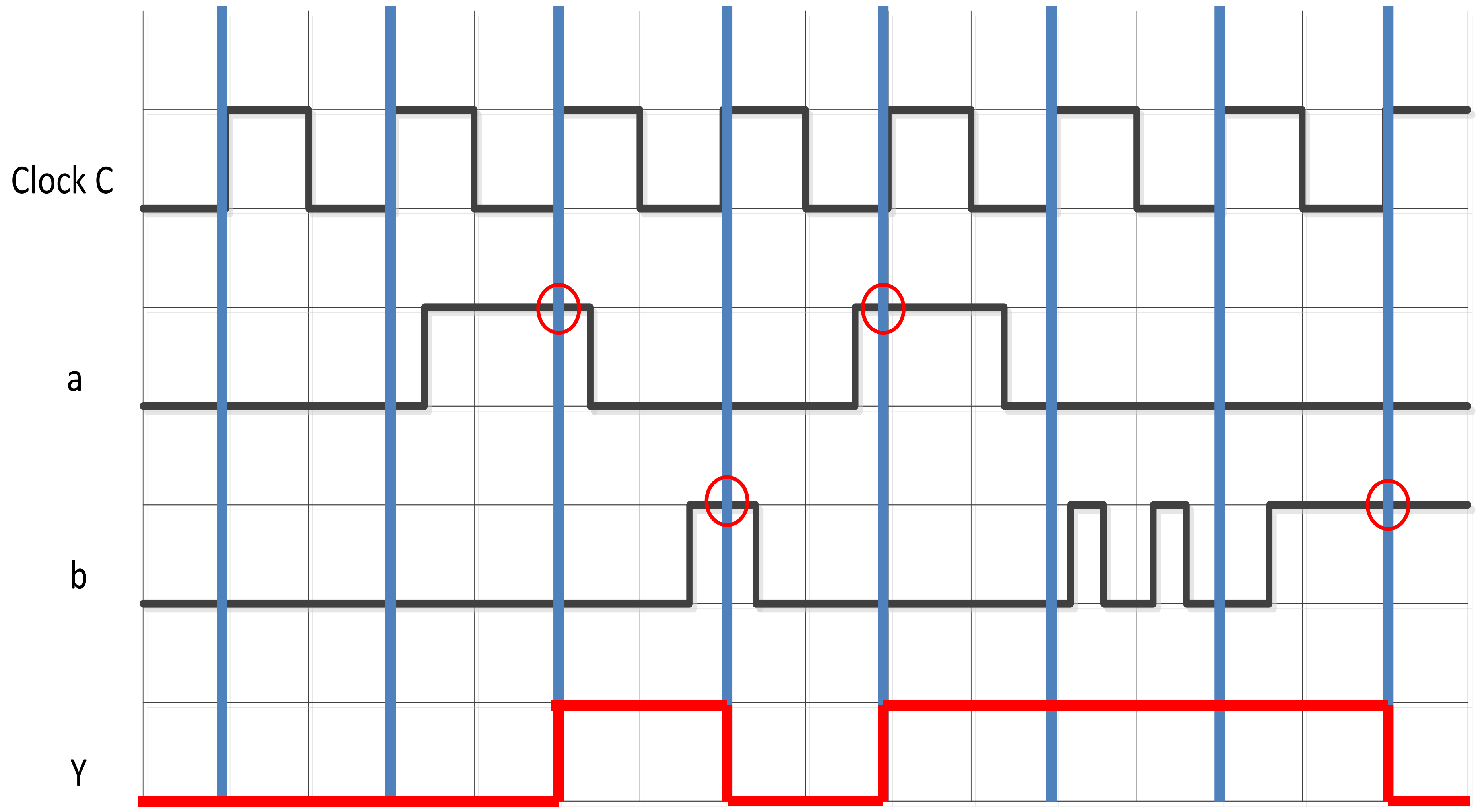
Question 4

JK Flip Flop only reacts upon positive rising edges!!!!!



Question 4

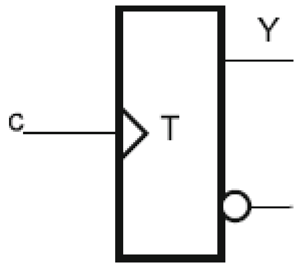
Results should look like this…..

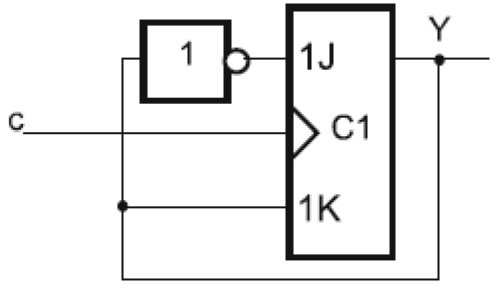


Question 5

(5a) What is a Bistable T-Gate? What is it’s behavior? [4points]
(5b) What can it be used for? [4points]

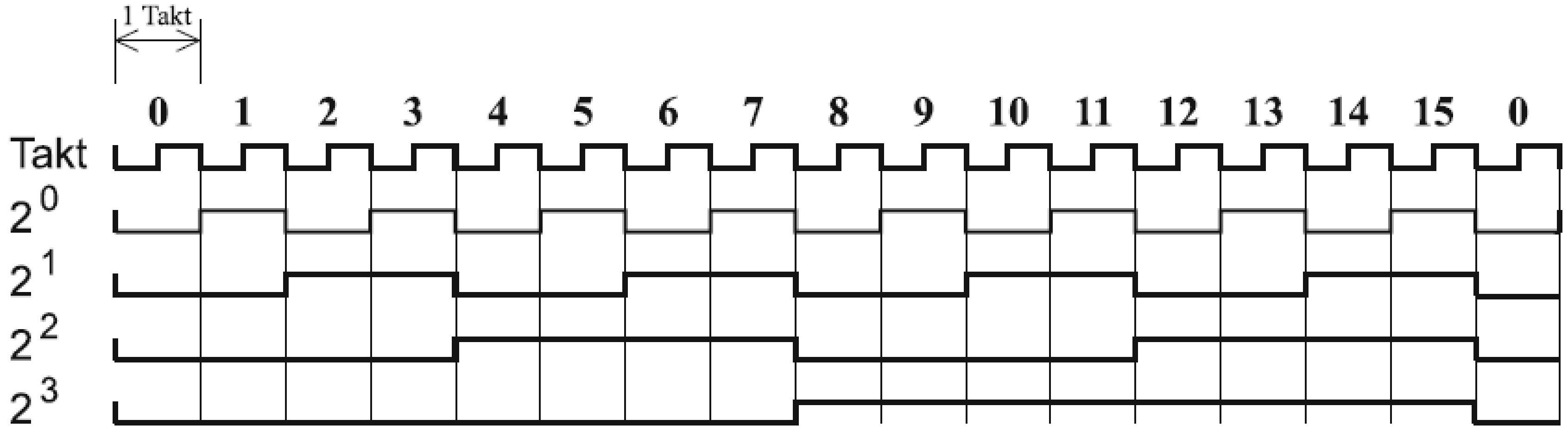
Bistable T-Gate (T Flip Flop)

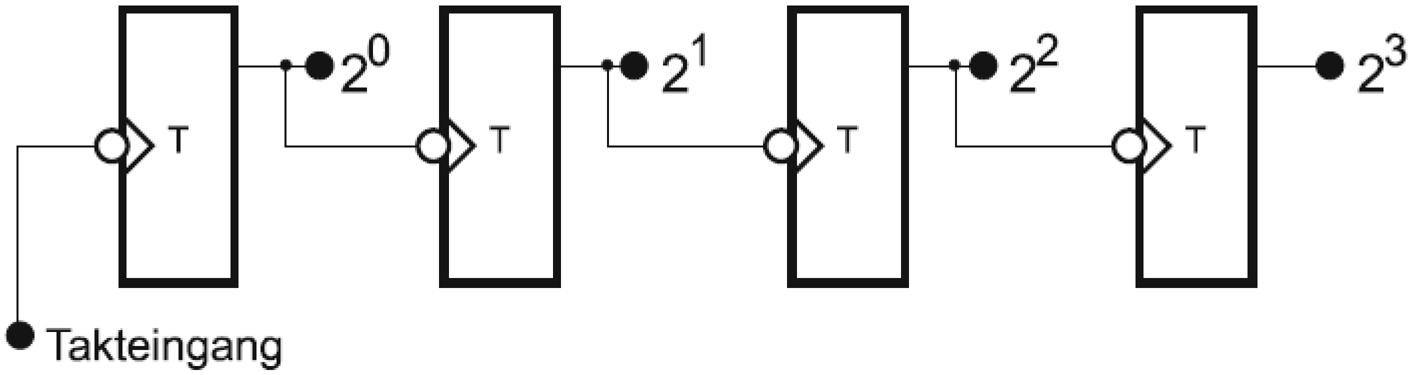




- Single input C
- Y changes its state with each rising edge

Asynchronous Counter

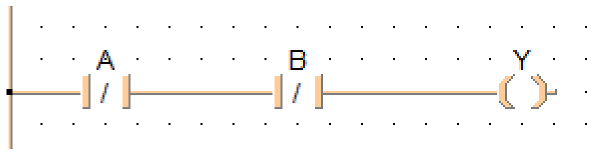




4 Bit asynchronous counter example, counts from 0 through 15

Question 6

(6) Draw a NAND Gate using Ladder Logic with inputs A, B and output Y [4points]



Question 7

(7) Structured Text. What is the difference between A=X and A:=X? [4points]

Question 8

- (8) A three story building has a stairway with light Y and a total of three switches A, B, C. Create a plc program that changes the state of light by switching any of the switches A, B or C.
- (8a) Draw the function chart [6points]
- (8b) Draw the Block Diagram [6points]
- (8c) Create PLC Ladder Diagram [6points]

Question 9

Structured Text: Explain the following code and its result: [4points]

A:= 4 AND 12;

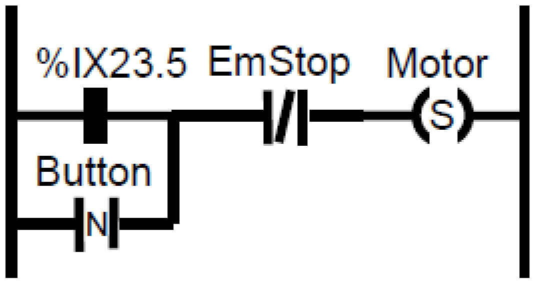
What did we do so far?

MIDTERIM WRAP-UP

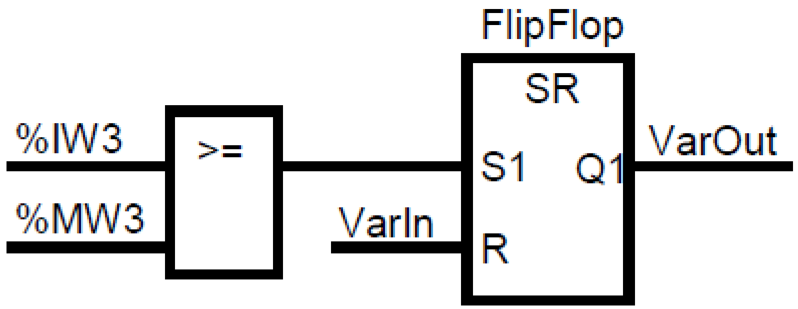
Overview of past weeks

- Application areas of PLC devices
- Principal architecture, real-time operating systems
  - Hardware-PLC’s (compact PLC’s and Modular PLC’s)
  - Soft-PLC’s (PC based)
- Logic & Boolean Arithmetic
- IEC61131-3
  - Ladder Diagrams
  - Function Block Diagrams
  - **Structured Text**
- Applied PLC Programming examples
  - Codesys
  - Control FPWin Pro
  - Panasonic FP06 PLC’s
    (basic programming, visualization, debugging, timing and interrupts)

PLC Programming Languages



Ladder Diagram



Function Block Diagram

```
LD      %IX23.5
ORN     Button
ANDN    EmStop
S       Motor
```

Instruction List

```
FlipFlop (   S1 := (%IW3 >= %MW3),
             R  := VarIn );

VarOut := FlipFlop.Q1;
```

Structured Text

…and yet to come

- Feedback control.
- PLC Compatible Terminals and Drivers.
- PLC Network Systems and Structures
- Analog Signal Acquisition
- Digital Bus Communication – CAN, CANopen, Modbus, EtherCat, …..
- …
- Who knows what else ;)

PLC Programming

IEC 61131-3
INTERNATIONAL PROGRAMMING STANDARD

International Programming Standard IEC61131-3

- open international standard for programmable logic controllers
- First published in 1993
- Part 3 deals with basic software architecture and programming languages within PLC’s
- Defines two graphical and two textual programming language standards:
  - **LD**: Ladder Diagrams
  - **FBD**: Function Block Diagram
  - **ST**: Structured Text
  - **IL**: instruction List
  - **SFC**: Sequential Function Chart

- Various implementations:
  - CodeSys
  - TwinCat (Beckhoff)
  - Control FP Win Pro (Panasonic)
- Different Implementations can be compatible, but are not required to
  GUI largly similar across implementations

POU Program organization unit

- Functions
  - Standard: ADD, SQRT, SIN, COS, MIN, …..
  - Custom Functions: user-definable
- Function Blocks
  - Standard: TOF, TON, RS, SR, …..
  - Custom Functions: User-definable
- Programs
  - Custom…

Variables

- Global
- Direct
- I/O Mapping
- ….

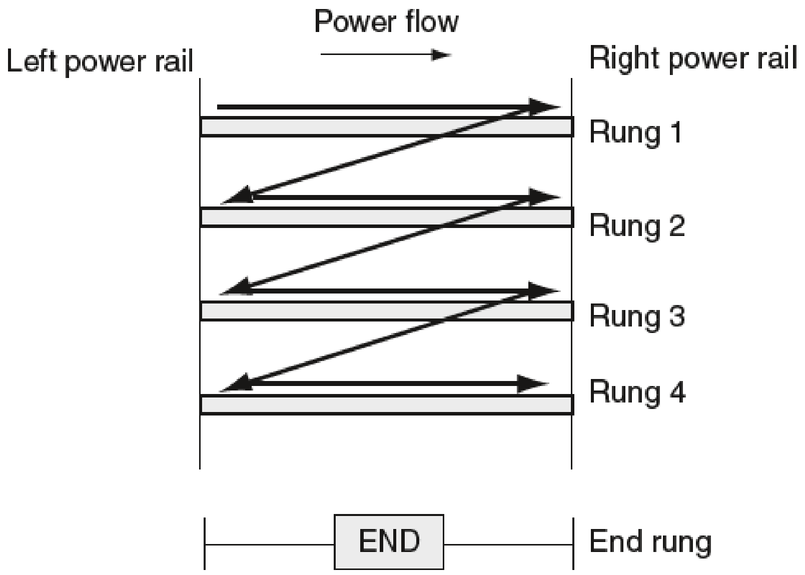
Basic Data Types

- Boolean [Bool] True/False
- Integer
- SINT - signed short integer (1 byte, -128..127)
- INT   - signed integer (2byte, -32768..32767)
- DINT – signed double integer (4byte, -2^31 .. (2^31)-1)
- UINT – unsigned integer (2 byte, 0..65536)
- REAL – floating point, 4 byte
- ….

Programming Languages in IEC 61131-3

LADDER DIAGRAMS
FUNCTION BLOCKS DIAGRAMS

Ladder Diagrams



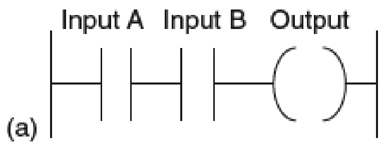
- The vertical lines of the diagram represent the power rails between which circuits are connected. The power flow is taken to be from the left-hand vertical across a rung.
- Each rung on the ladder defines one operation in the control process.
- A ladder diagram is read from left to right and from top to bottom.
  The figure is showing the scanning motion employed by the PLC. The top rung is read from left to right. Then the second rung down is read from left to right and so on.
- When the PLC is in its run mode, it goes through the entire ladder program to the end, the end rung of the program being clearly denoted, and then promptly resumes at the start. This procedure of going through all the rungs of the program is termed **a cycle**. The end rung might be indicated by a block with the word END or RET for return, since the program promptly returns to its beginning.

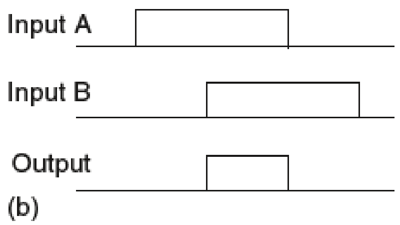
Ladder Diagrams

- Each rung must start with an input or inputs and must end with at least one output. The term input is used for a control action, such as closing the contacts of a switch, used as an input to the PLC. The term output is used for a device connected to the output of a PLC, e.g., a motor.
- Electrical devices are shown in their normal condition. Thus a switch, which is normally open until some object closes it, is shown as open on the ladder diagram. A switch that is normally closed is shown closed.
- A particular device can appear in more than one rung of a ladder. For example, we might have a relay that switches on one or more devices. The same letters and/or numbers are used to label the device in each situation.
- The inputs and outputs are all identified by their addresses, the notation used depending on the PLC manufacturer. This is the address of the input or output in the memory of the PLC.

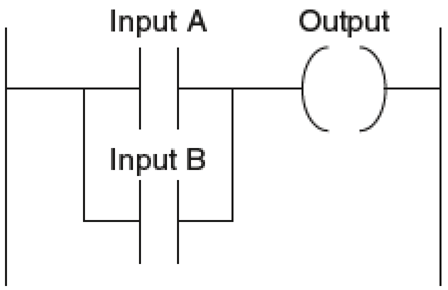
Ladder Logic: Logic Elements: AND, OR, NOT

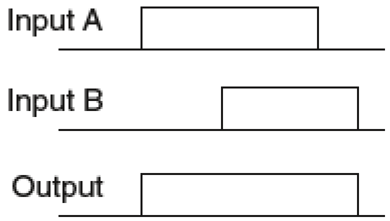
AND Gate



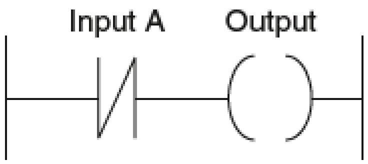


OR Gate

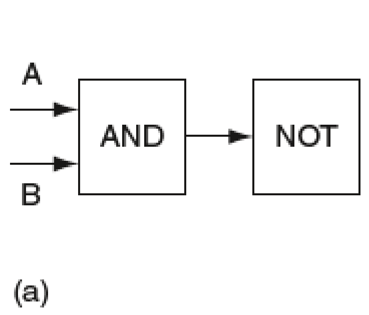


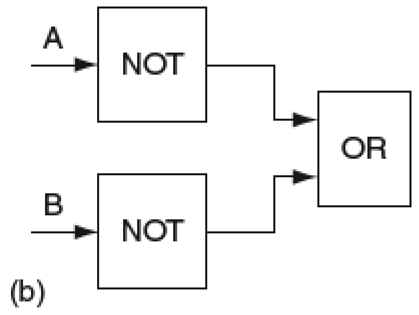


NOT Gate

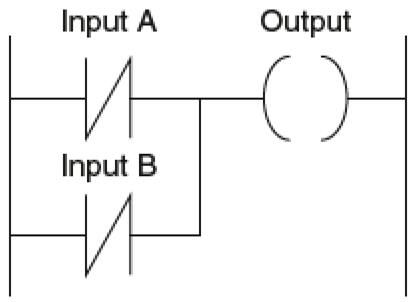


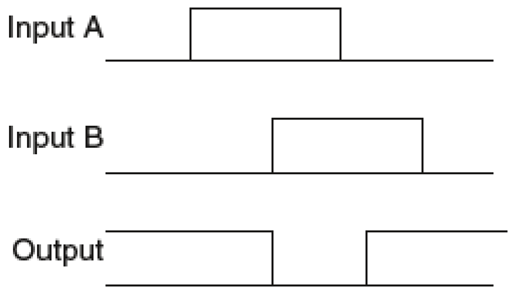
NAND Gate



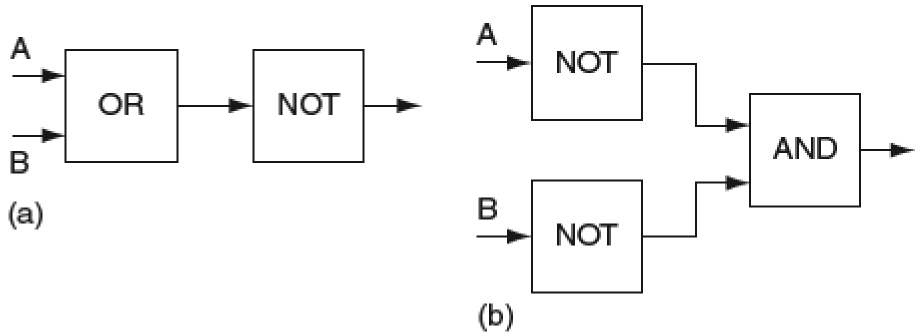


Inputs		Output
A	B	
0	0	1
0	1	1
1	0	1
1	1	0

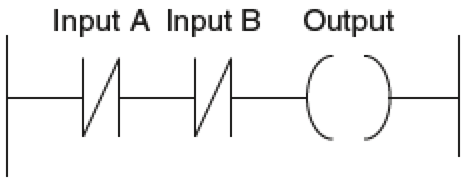


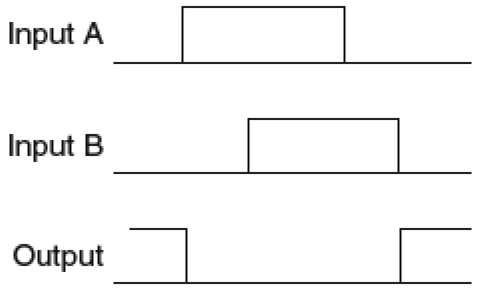


NOR Gate

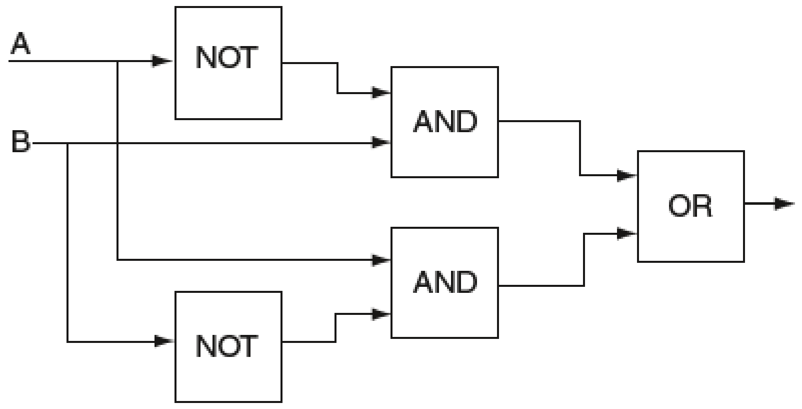


Inputs		Output
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

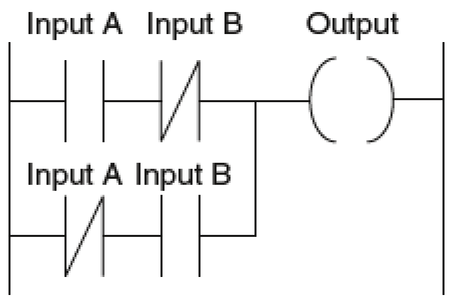


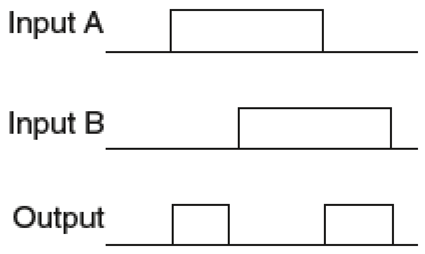


Exlusive OR - XOR Gate



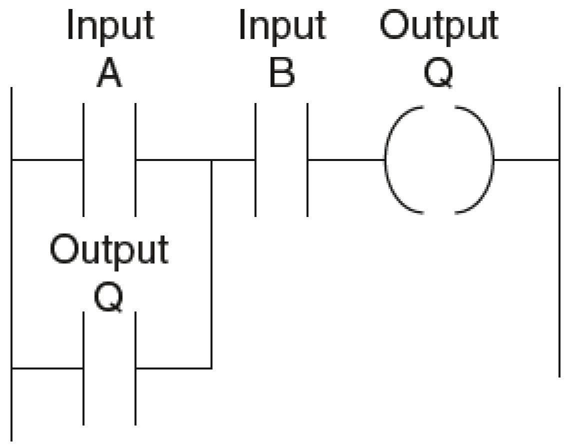
Inputs		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

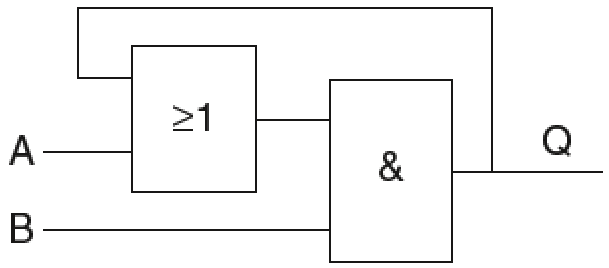




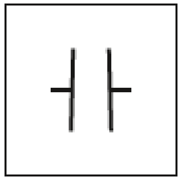
Latching

- There are often situations where it is necessary to hold an output energized, even when the input ceases.
- Simple example: motor, which is started by pressing a push button switch. Though the switch contacts do not remain closed, the motor is required to continue running until a stop push button switch is pressed. The term **latch circuit** is used for the circuit used to carry out such an operation.
- Latch Circuit:
  Self-maintaining circuit that, after being energized, maintains that state until another input is received.

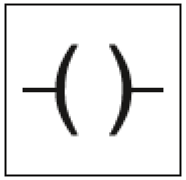




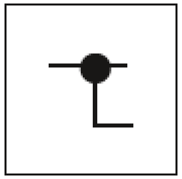
Ladder Symbols & Function Blocks



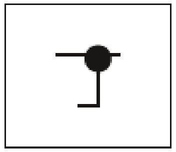
Contact (Input)



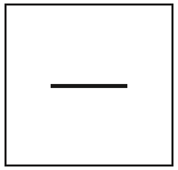
Coil (Output)



Junction Start



Junction End



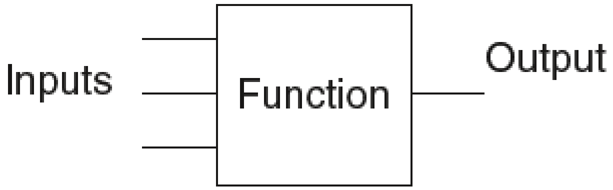
Connection Line

Function Blocks

Function blocks are program instruction units. They can be used within Ladder Diagrams. They have one or more inputs, and one or more outputs.

Examples:

- TON – timer on delay
- TOF - timer off delay
- RS / SR Flip Flops
- User-defined custom functions


Inputs
Function
Output