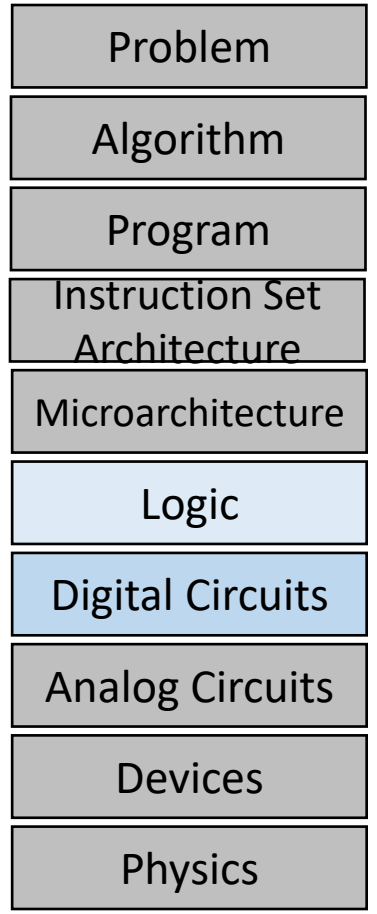
Universität Bremen

Logic Design
Sequential Circuits

Prof. Dr. Rolf Drechsler
Dr. Muhammad Hassan
M.Sc. Jan Zielasko
M.Sc. Milan Funck



Universität Bremen

Announcements

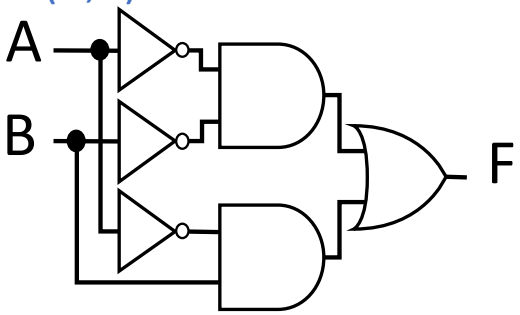
- It’s not compulsory to do exercises
  - However, highly recommended
  - Makes you eligible for Fachgespräch
- You can take oral exam without doing the exercises
- Groups registration
  - Anyone still looking for group members?

Revision

Representations of Boolean Functions

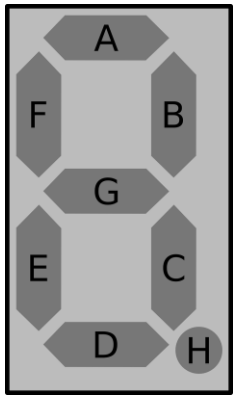
Natural language: F outputs 1 when A is 0 and B is 0, or when A is 0 and B is 1.

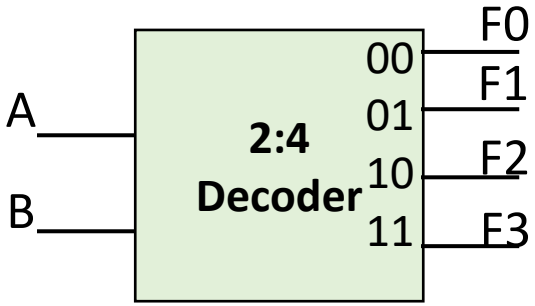
F(A,B) = A'B' + A'B



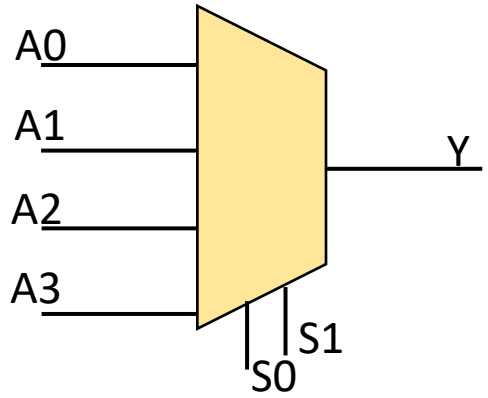
A	B	F
0	0	1
0	1	1
1	0	0
1	1	0

Decoder





Multiplexers



Universität Bremen

Today's Outline

- Clocks
- Memory Elements
  - Latches
  - Flip-Flops
  - Registers
- Finite State Machines (FSMs)

Evolution of Technology ….

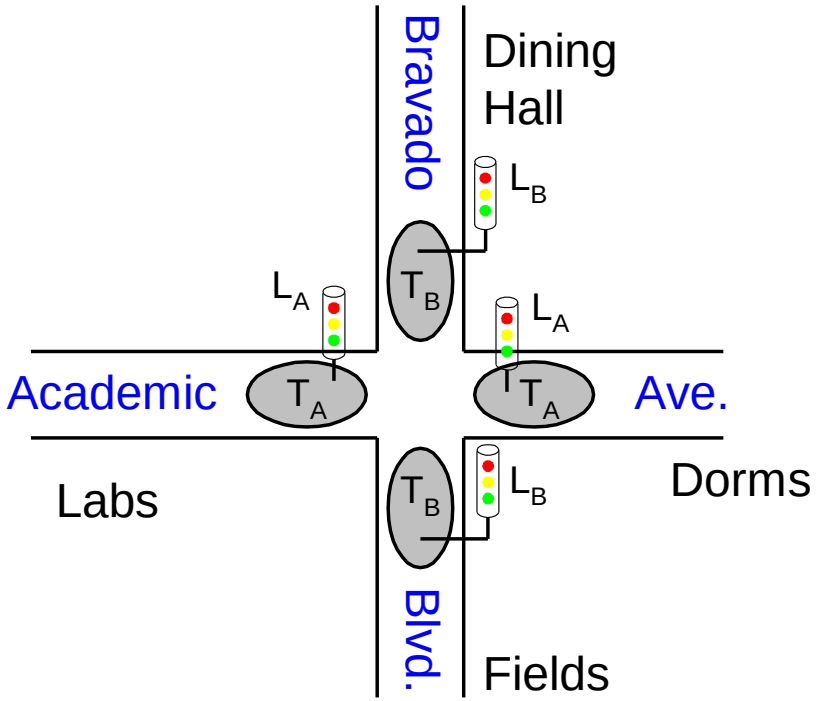
Universität Bremen

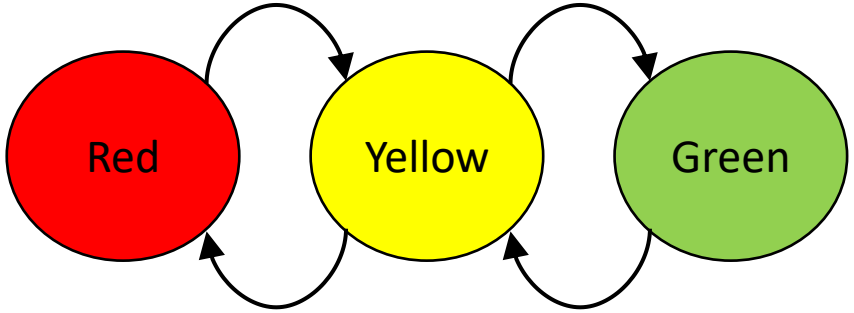
Some Questions …

- Using only combinational circuits, can we develop complex systems?
  - What additional things we might require?
- Why is memory required by complex systems for performing any task?
  - E.g., robots need to remember past events, actions, and sensor readings to make informed decision (avoiding going in circles).
- Why is synchronization important in a complex system?
  - E.g., multiple sensor and actuators working simultaneously require to work in sync
  - How can we achieve it?

Introduction

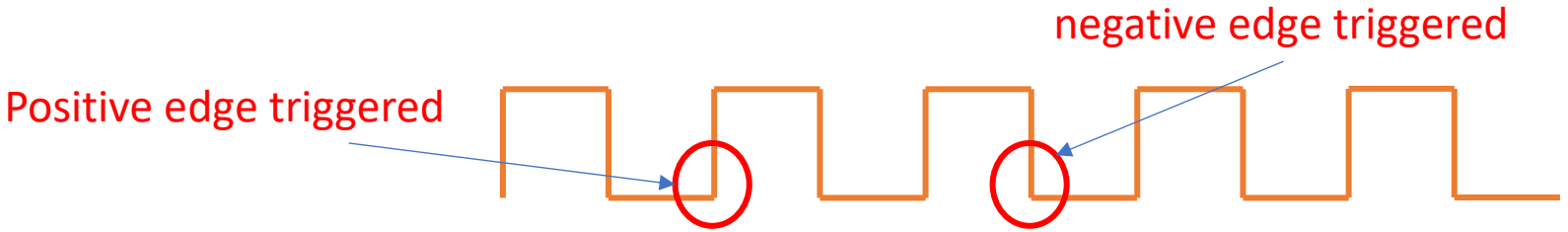
- How does traffic signal controller know whether to switch on RED or YELLOW or GREEN light?





The Notion of Clock

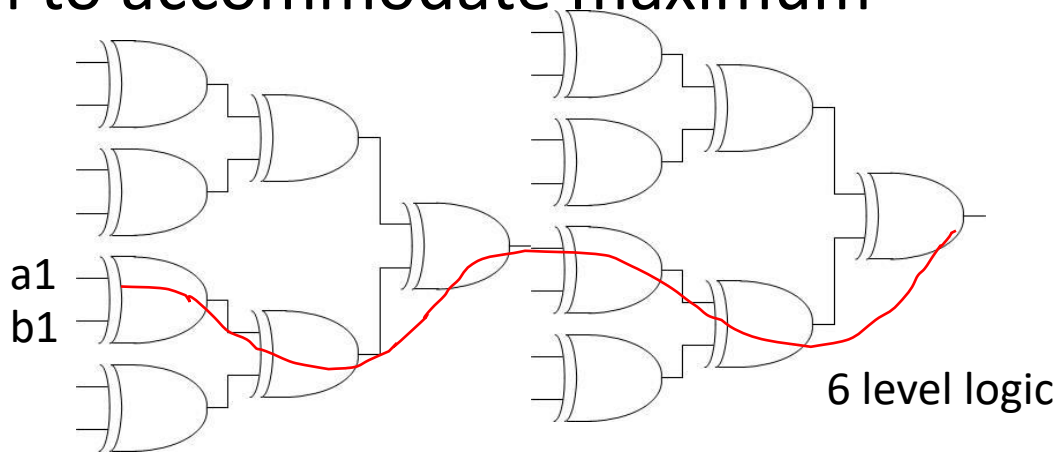
- When should the light change from one state to another?
- We need a **clock** to dictate when to change state
- Clock signal alternates between 0 & 1



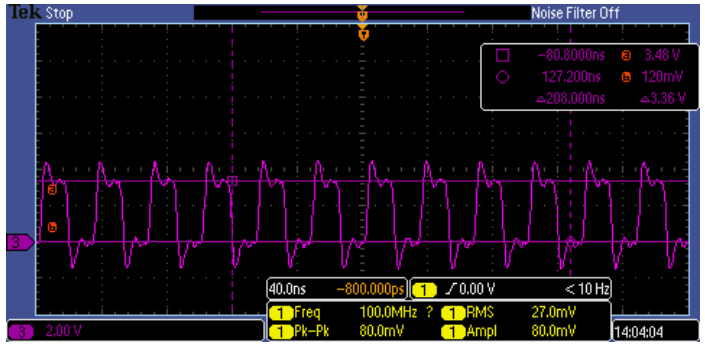
- At the start of a clock cycle, system state changes
  - During a clock cycle, the state stays constant
  - In traffic light example, we are assuming the traffic light stays in each state an equal amount of time

The Notion of Clock cont…

- **Clock** is a general mechanism that **triggers transition from one state to another** in a sequential circuit
- Clock **synchronizes state changes** across many sequential circuit elements
- Combinational logic evaluates for the length of the clock cycle
- Clock cycle should be chosen to accommodate maximum combinational circuit delay



Role of Clock in Complex Systems



- Synchronization
  - Without synchronization, data might be read incorrectly
- Timing reference
  - Allows the system to know when an event should take place
- Data transfer
  - Timing of data transfer, e.g., serial communication baud rate
- Determining speed
  - How fast a system can process data

Sequential Logic

- Outputs of sequential logic depend on current and prior input values
  - it has memory.
- Some definitions:
  - **State:** all the information about a circuit necessary to explain its future behavior
  - **Latches and flip-flops:** state elements that store one bit of state
  - **Synchronous sequential circuits:** Sequential circuits using flip-flops sharing a common clock

Universität Bremen

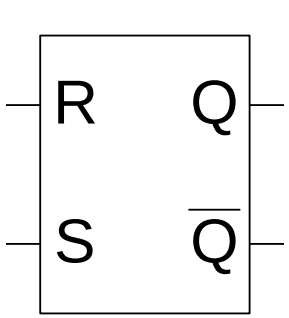
Sequential Circuits

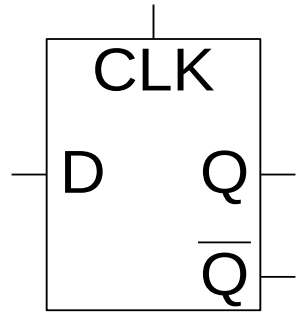
- Give sequence to events
- Have memory (short-term)
- Use feedback from output to input to store information

State Elements

- **State**: everything about the prior inputs to the circuit necessary to predict its future behavior
  - Usually just 1 bit, the last value captured
- State elements store state
  - SR Latch
  - D Latch
  - D Flip-flop
  - Registers

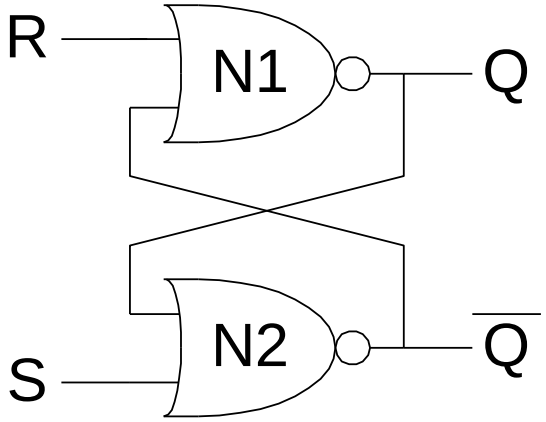
SR Latch Symbol





SR (Set/Reset) Latch

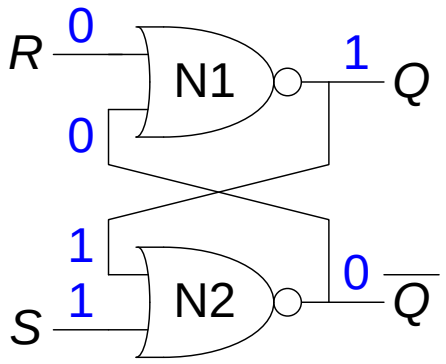
- **SR Latch**



- Consider the four possible cases:
  - ***S* = 1, *R* = 0**
  - ***S* = 0, *R* = 1**
  - ***S* = 0, *R* = 0**
  - ***S* = 1, *R* = 1**

SR Latch Analysis

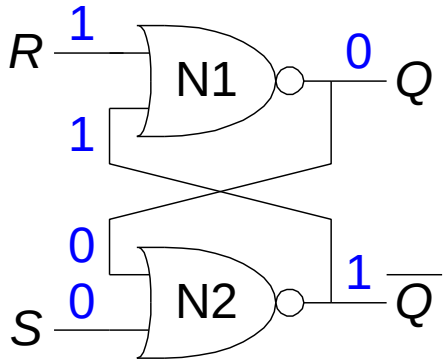
- **S = 1**, R = 0:
  - then **Q = 1** and Q’ = 0
  - **Set the output**



A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

NOR gate

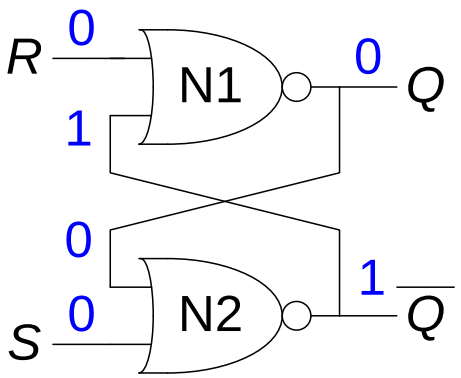
- S = 0, **R = 1**:
  - then **Q = 0** and Q’ = 1
  - **Reset the output**



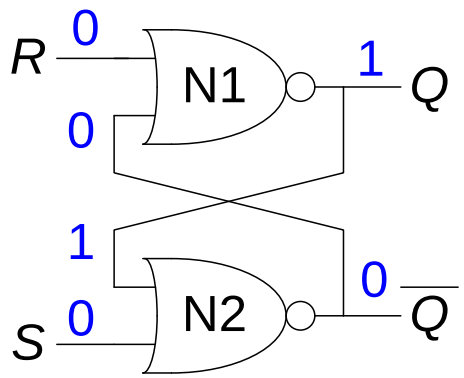
SR Latch Analysis cont…

- S = 0, R = 0:
  - then Q = Q_{prev}
  - **Memory!**

$Q_{prev} = 0$

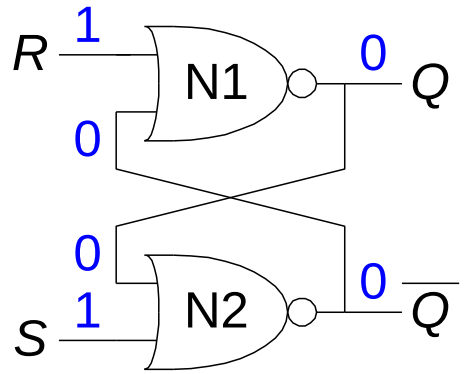


$Q_{prev} = 1$



A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

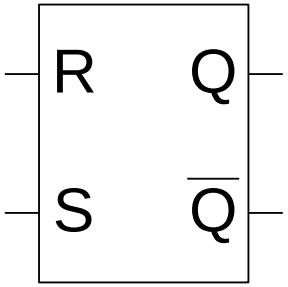
- S = 1, R = 1:
  - then Q = 0, $\overline{Q}$ = 0
  - **Invalid State**
  - **Q ≠ NOT $\overline{Q}$**



SR Latch

- SR stands for Set/Reset Latch
  - Stores one bit of state (Q)
- Control what value is being stored with S, R inputs
  - **Set:** Make the output 1
  - S = 1, R = 0, Q = 1
  - **Reset:** Make the output 0
  - S = 0, R = 1, Q = 0
  - **Memory:** Retain value
  - S = 0, R = 0, Q = Q_{prev}

SR Latch Symbol

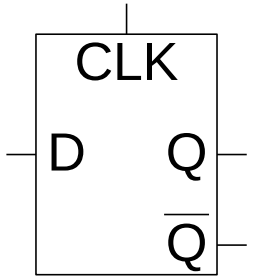


- **Must do something to avoid invalid state (when S = R = 1)**

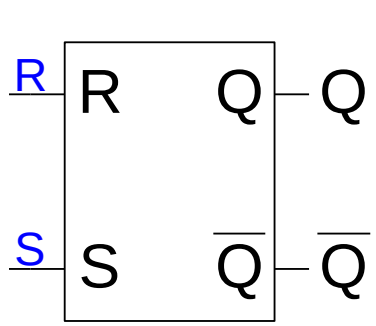
D Latch

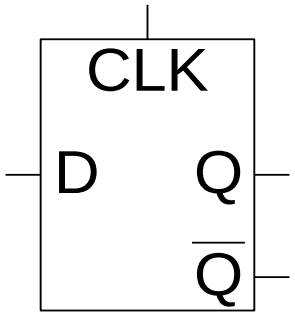
- Two inputs: CLK, D
  - CLK: controls when the output changes
  - D (the data input): controls what the output changes to
- Function
  - When CLK = 1,
    - D passes through to Q (transparent)
  - When CLK = 0,
    - Q holds its previous value (opaque)
- Avoids invalid case when
  - Q ≠ NOT $\overline{Q}$

D Latch Symbol



D Latch Internal Circuit

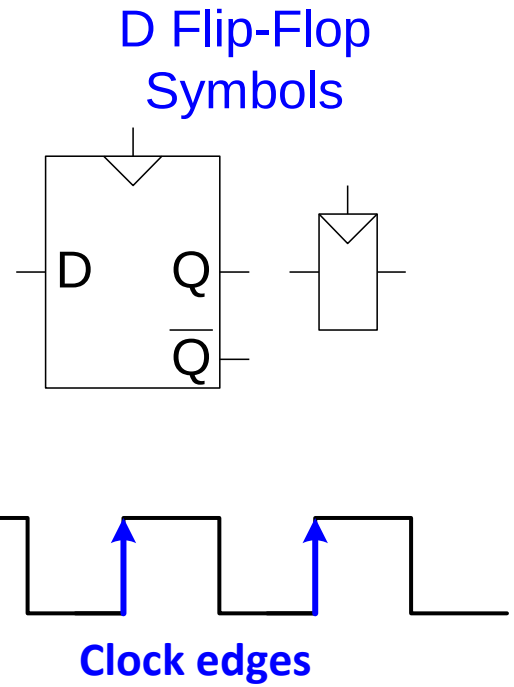




CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

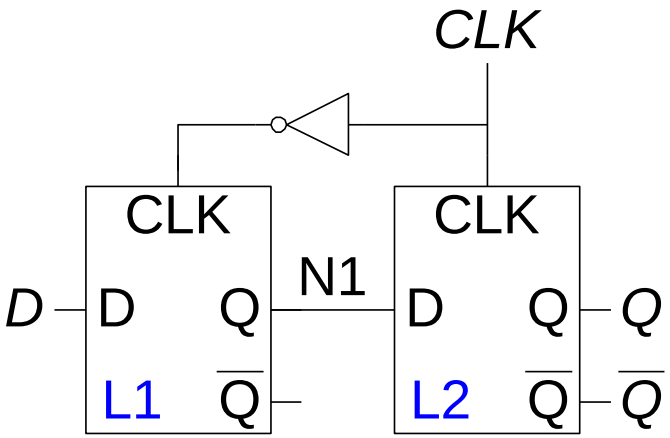
D Flip-Flop

- Inputs: CLK, D
- Function:
  - Samples D on **rising edge** of CLK
  - When CLK rises from 0 to 1, D passes through to Q
  - Otherwise, Q holds its previous value
  - Q changes only on rising edge of CLK
- Called **edge-triggered**
  - Activated on the clock edge

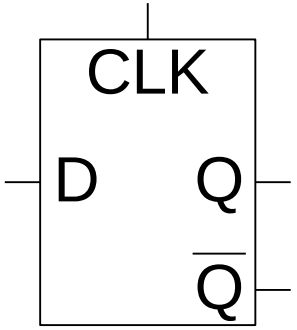


D Flip-Flop Internal Circuit

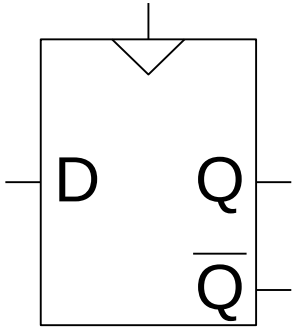
- Two back-to-back D latches (L1 and L2) controlled by complementary clocks
- When CLK = 0
  - L1 is transparent
  - L2 is opaque
  - D passes through to N1
- When CLK = 1
  - L2 is transparent
  - L1 is opaque
  - N1 passes through to Q
- Thus, on the edge of the clock (when CLK rises from 0 → 1)
  - D passes through to Q



D Latch vs D Flip-Flop



D Latch



D Flip-flop

Synchronized to the clock

CLK

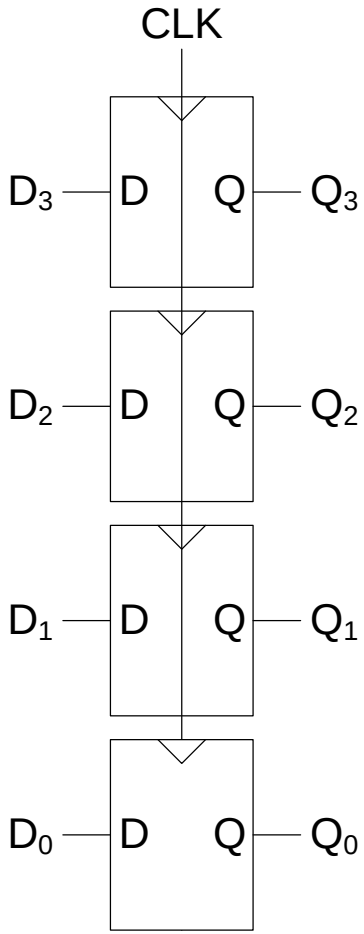
D

Q (latch)

Q (flop)

Registers

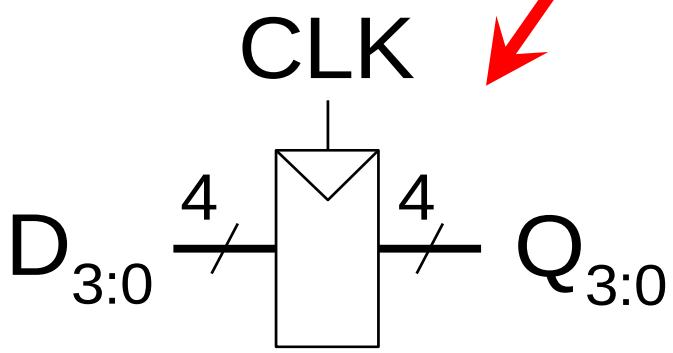
- One or more Flip-flops



4-bit Register

Two ways to draw a register

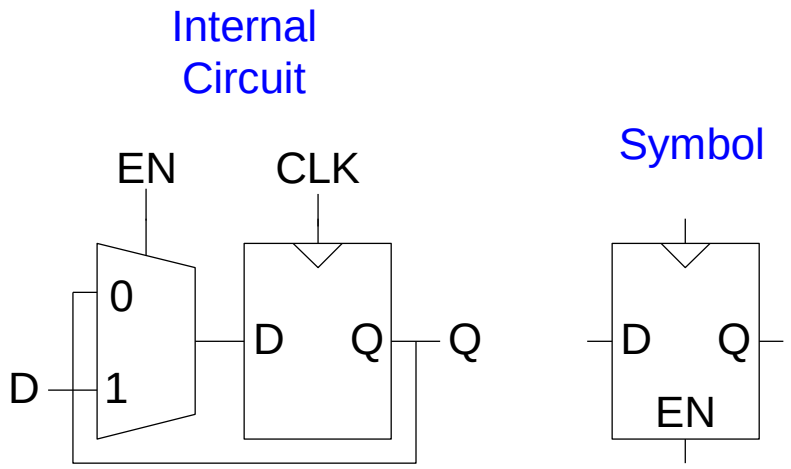
Easier to draw!



4-bit Register

Flip-Flop with Enable

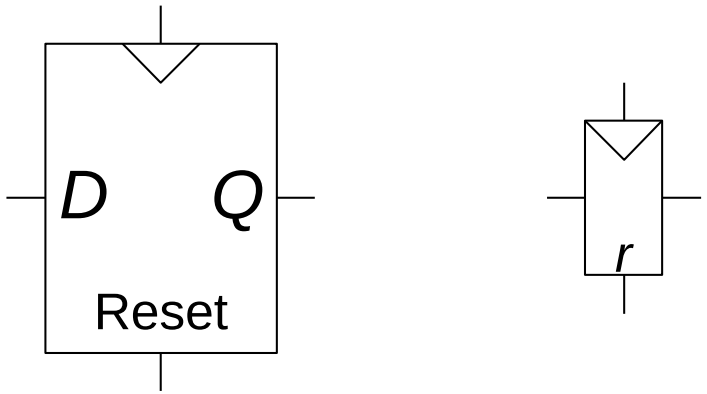
- Inputs: CLK, D, EN
  - The enable input (EN) controls when new data (D) is stored
- Function
  - EN = 1: D passes through to Q on the clock edge
  - EN = 0: the flip-flop retains its previous state



Flip-Flop with Reset

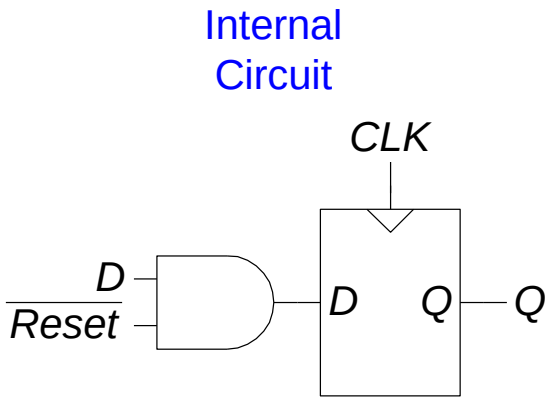
- Inputs: CLK, D, Reset
- Function:
  - Reset = 1: Q is forced to 0
  - Reset = 0: flip-flop behaves as ordinary D flip-flop

Symbols



Flip-Flop with Reset cont….

- Two types:
  - Synchronous: resets at the clock edge only
  - Asynchronous: resets immediately when Reset = 1
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop
- Synchronously resettable flip-flop?



Universität Bremen

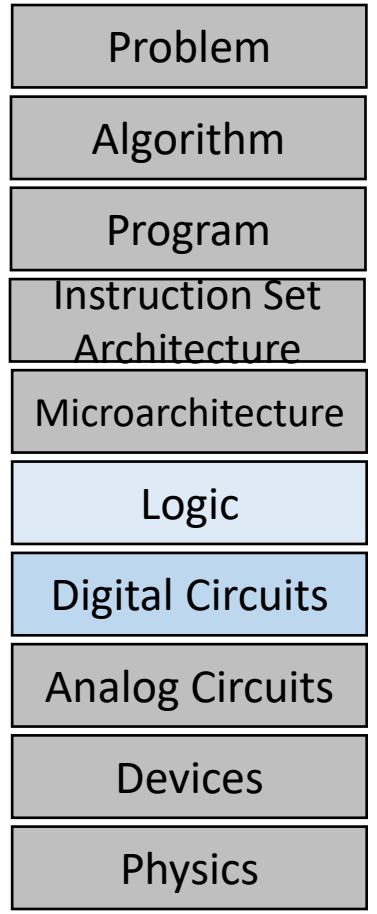
Synchronous Sequential Logic Design

- Breaks cyclic paths by inserting registers
- Registers contain state of the system
- State changes at clock edge: system synchronized to the clock
- Rules of synchronous sequential circuit composition:
  - Every circuit element is either a register or a combinational circuit
  - At least one circuit element is a register
  - All registers receive the same clock
  - Every cyclic path contains at least one register
- Two common synchronous sequential circuits
  - Finite State Machines (FSMs)
  - Pipelines (Lecture 10)

Universität Bremen

Logic Design
Sequential Circuits

Prof. Dr. Rolf Drechsler
Dr. Muhammad Hassan
M.Sc. Jan Zielasko
M.Sc. Milan Funck



Universität Bremen

Literature

- D. Patterson, J. Hennessy: Computer Organization and Design RISC-V Edition – The Hardware Software Interface, Elsevier, 2020.
- S. L. Harris, D. Harris: Digital Design and Computer Architecture, RISC-V Edition, Elsevier, 2021.
- ARM University program: Introduction-to-Computer-Architecture-Education.
- J. Teich, C. Haubelt: Digitale Hardware/Software-Systeme – Synthese und Optimierung, Springer Verlag, 2. Auflage, 2007.
- G. De Micheli: Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.
- G. D. Hachtel, F. Somenzi: Logic Synthesis and Verification Algorithms, Kluwer, 1996.

Universität Bremen

Literature

- H. Bähring, J. Dunkel, G. Rademacher: Mikrorechner-Systeme: Mikroprozessoren, Speicher, Peripherie, Springer-Verlag, 2. Auflage, 1994.
- J. P. Hayes: Computer Architecture and Organization, McGraw-Hill, 1998.
- M. G. Arnold: Verilog Digital Computer Design: Algorithms to Hardware, Prentice Hall, 1998.
- A. Tanenbaum: Structured Computer Organization, Prentice Hall, 5th Edition, 2006.
- D. A. Patterson, J. L. Hennessy: Computer Organization and Design - The Hardware/Software-Interface, Morgan Kaufmann, 3. Auflage, 2007.
- J. L. Hennessy, D. A. Patterson: Computer Architecture - A Quantitative Approach, Morgan Kaufmann, 4. Auflage, 2007.
- H. Kaeslin: Digital Integrated Circuit Design, From VLSI to CMOS Fabrication, Cambridge University Press, 2008.
- A. Biere, D. Kroening, G. Weissenbacher, C. M. Wintersteiger: Digitaltechnik – Eine praxisnahe Einführung, Springer-Verlag, 2008.