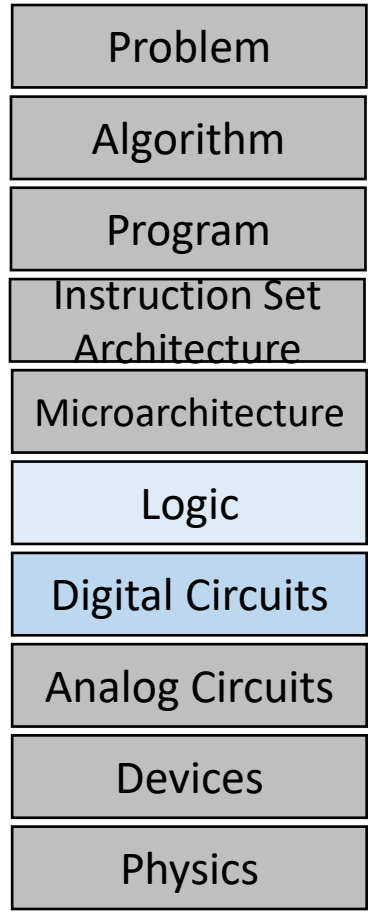
Universität Bremen

Logic Design
Finite State Machines

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



Universität Bremen

Industry 4.0 and Smart Factories

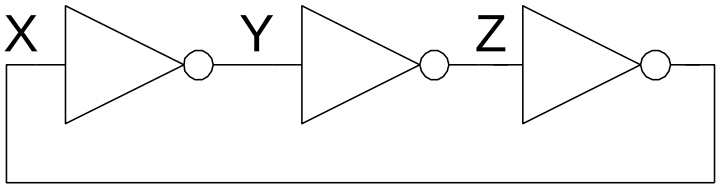
Universität Bremen

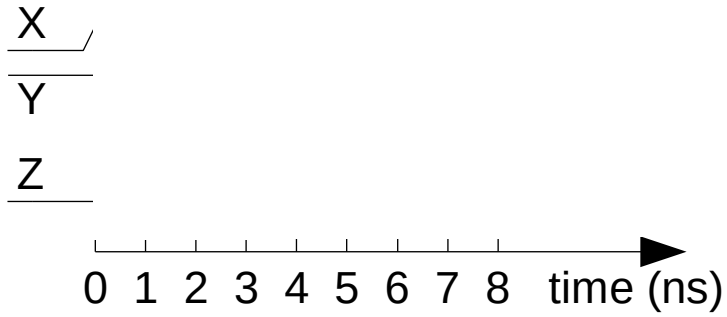
Latch vs Flip Flop

Feature	Latches	Flip-Flops
Triggering	Level-sensitive (respond to level of control signal)	Edge-triggered (respond to changes in the clock signal)
Structure	Simpler, usually fewer transistors	More complex, usually more transistors
Size	Generally smaller	Generally larger
Power Consumption	Can be lower (especially if the clock signal is not distributed to every latch)	Typically higher due to clock distribution
Transparency	Transparent when enable signal is active (output follows input)	Not transparent (output changes only on clock edge)
Timing Requirements	Less strict (more forgiving with setup and hold times)	More strict (requires careful management of setup and hold times)
Suitability for Synchronous Design	Can be used but may complicate timing analysis	Well-suited and commonly used
Suitability for Asynchronous Design	Well-suited (no need for a global clock)	Less common due to reliance on clock edges

Sequential Logic

- Sequential circuits: all circuits that aren't combinational
- A problematic circuit:





- No inputs and 1-3 outputs
- Astable circuit, oscillates
- Period depends on inverter delay
- It has a cyclic path: output fed back to input

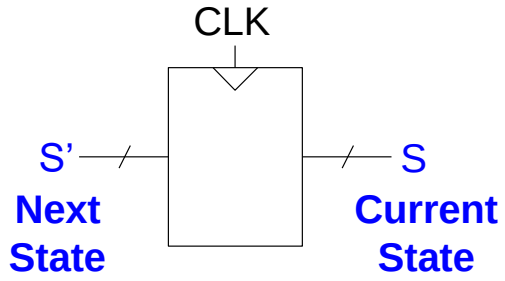
Universität Bremen

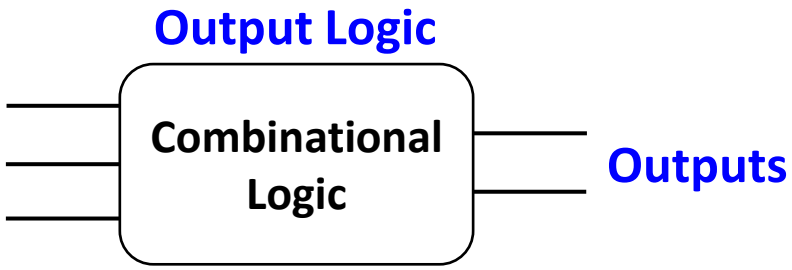
Synchronous Sequential Logic Design

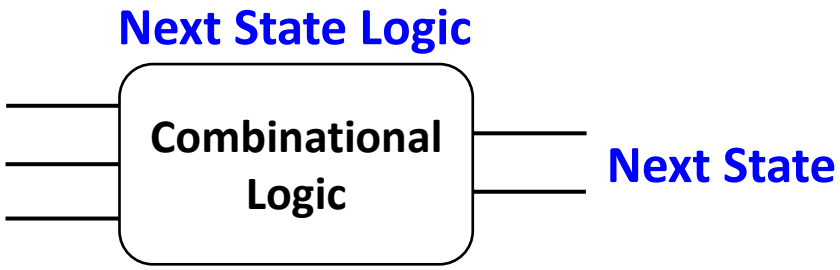
- Breaks cyclic paths by inserting registers
- Registers contain state of the system
- State changes at clock edge: system synchronized to the clock
- Rules of synchronous sequential circuit composition:
  - Every circuit element is either a register or a combinational circuit
  - At least one circuit 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)

Finite State Machine (FSM)

- Consists of:
  - State register
    - Stores current state
    - Loads next state at clock edge
  - Combinational logic
    - Computes the next state
    - Computes the outputs

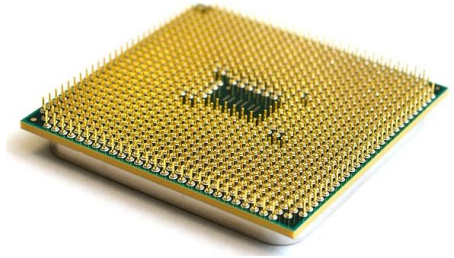




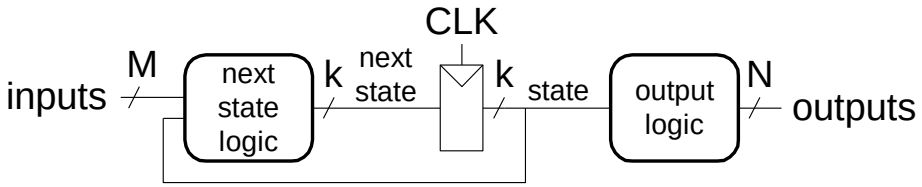


Finite State Machine cont…

- Next state determined by current state and inputs
- Two types of finite state machines differ in output logic:
  - **Moore FSM:** outputs depend only on current state
  - **Mealy FSM:** outputs depend on current state and inputs



Moore FSM



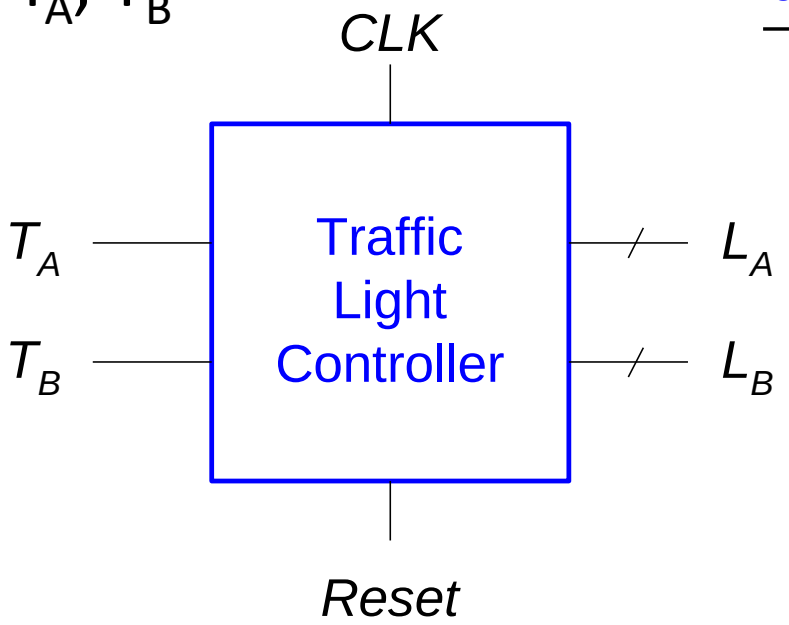
Universität Bremen

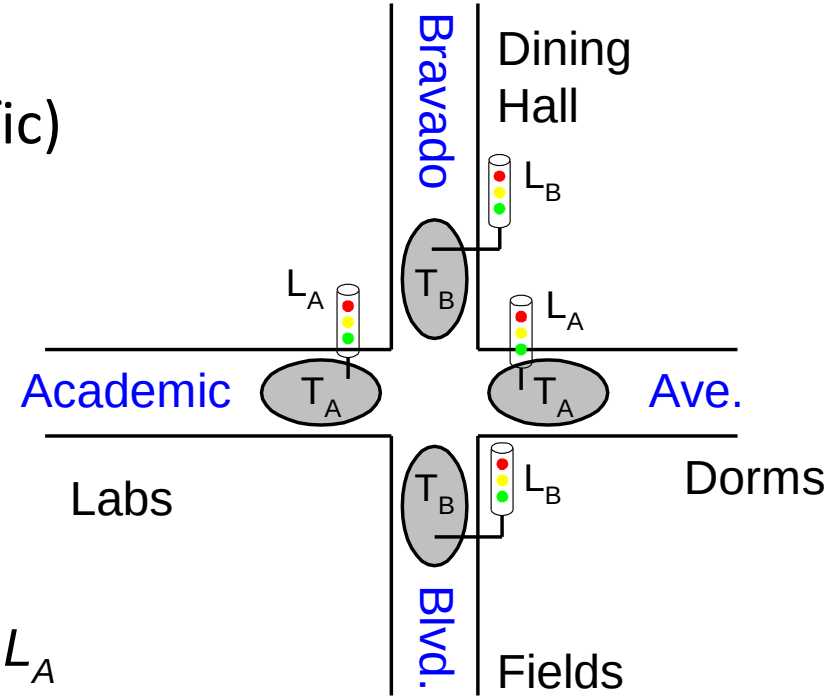
FSM Design Procedure

- Identify inputs and outputs
- Sketch state transition diagram
- Write state transition table and output table
- Moore FSM: write separate tables
- Mealy FSM: write combined state transition and output table
- Select state encodings
- Rewrite state transition table and output table with state encodings
- Write Boolean equations for next state and output logic
- Sketch the circuit schematic

Moore FSM Example

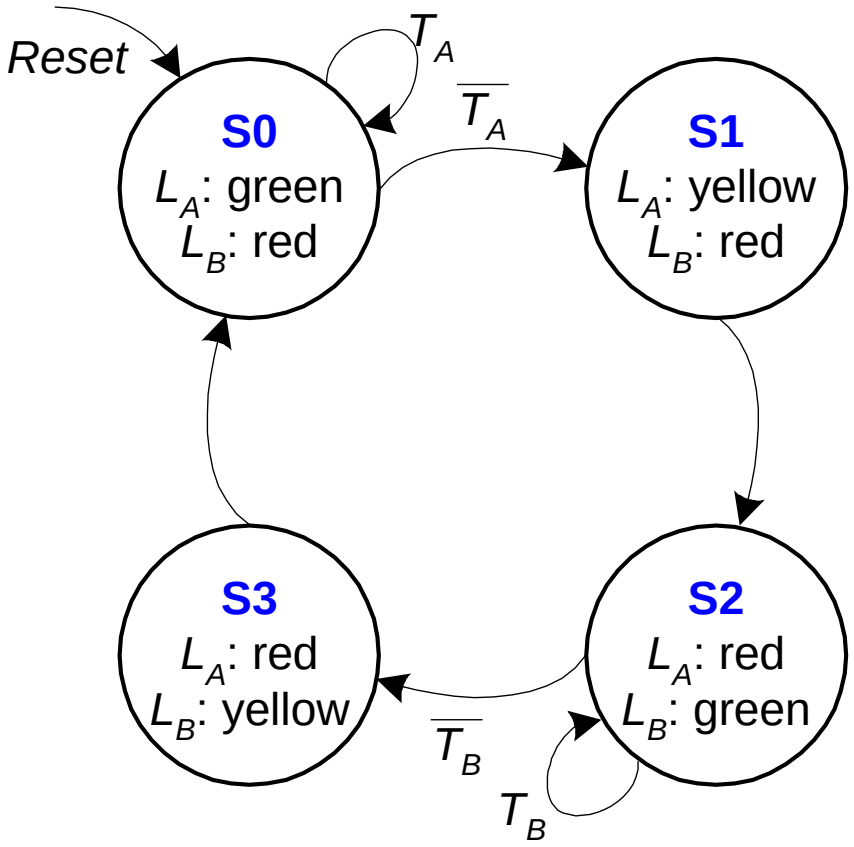
- Traffic light controller
  - Traffic sensors: T_A, T_B (TRUE when there's traffic)
  - Lights: L_A, L_B
- Inputs: CLK, Reset, T_A, T_B
- Outputs: L_A, L_B

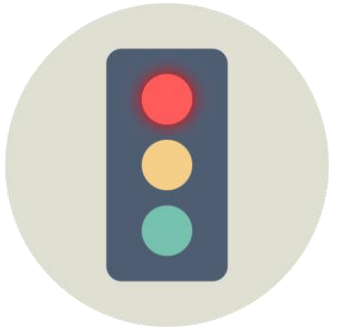


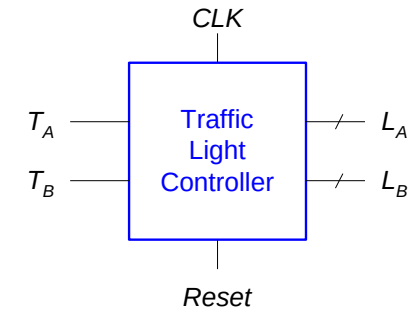


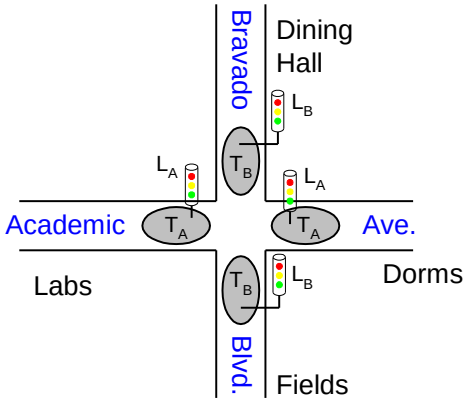
FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs



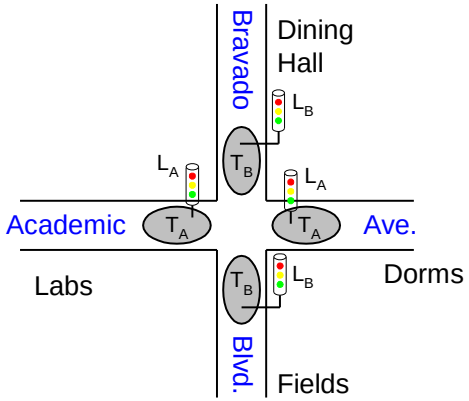


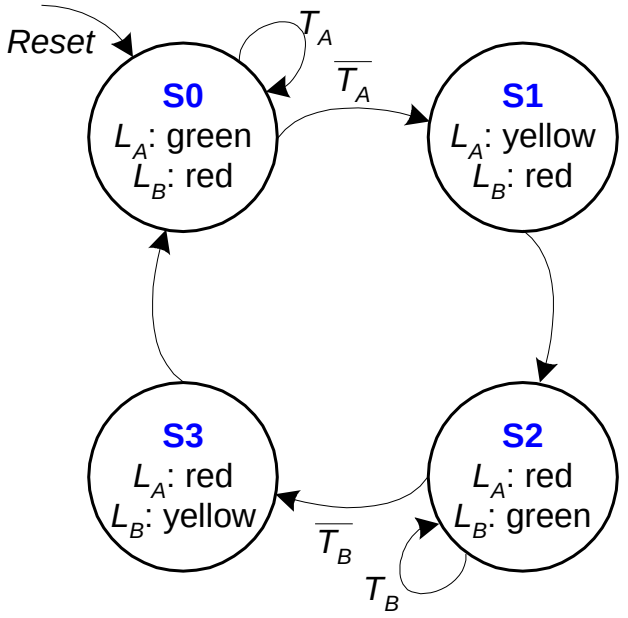




FSM State Transition Table

Current State	Inputs		Next State
S	**T_A**	**T_B**	**S'**
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0



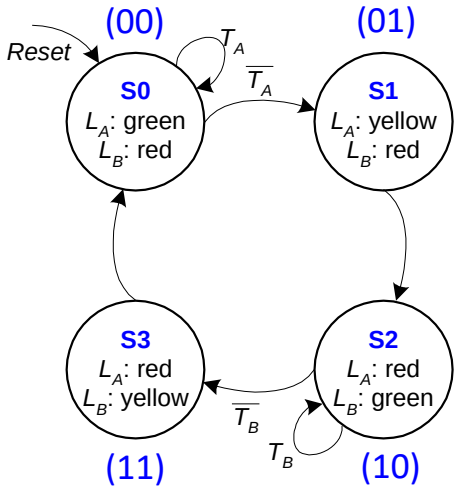


S **:** Current State

S'**:** Next State

FSM Encoded Transition Table

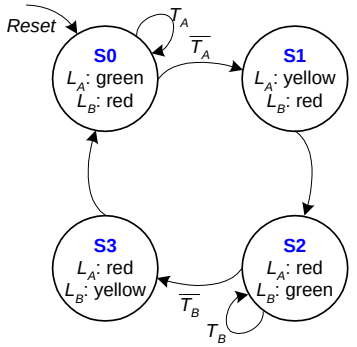
Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
S0		0	X	S1	
S0		1	X	S0	
S1		X	X	S2	
S2		X	0	S3	
S2		X	1	S2	
S3		X	X	S0	



State	Encoding
S0	00
S1	01
S2	10
S3	11

FSM Output Table

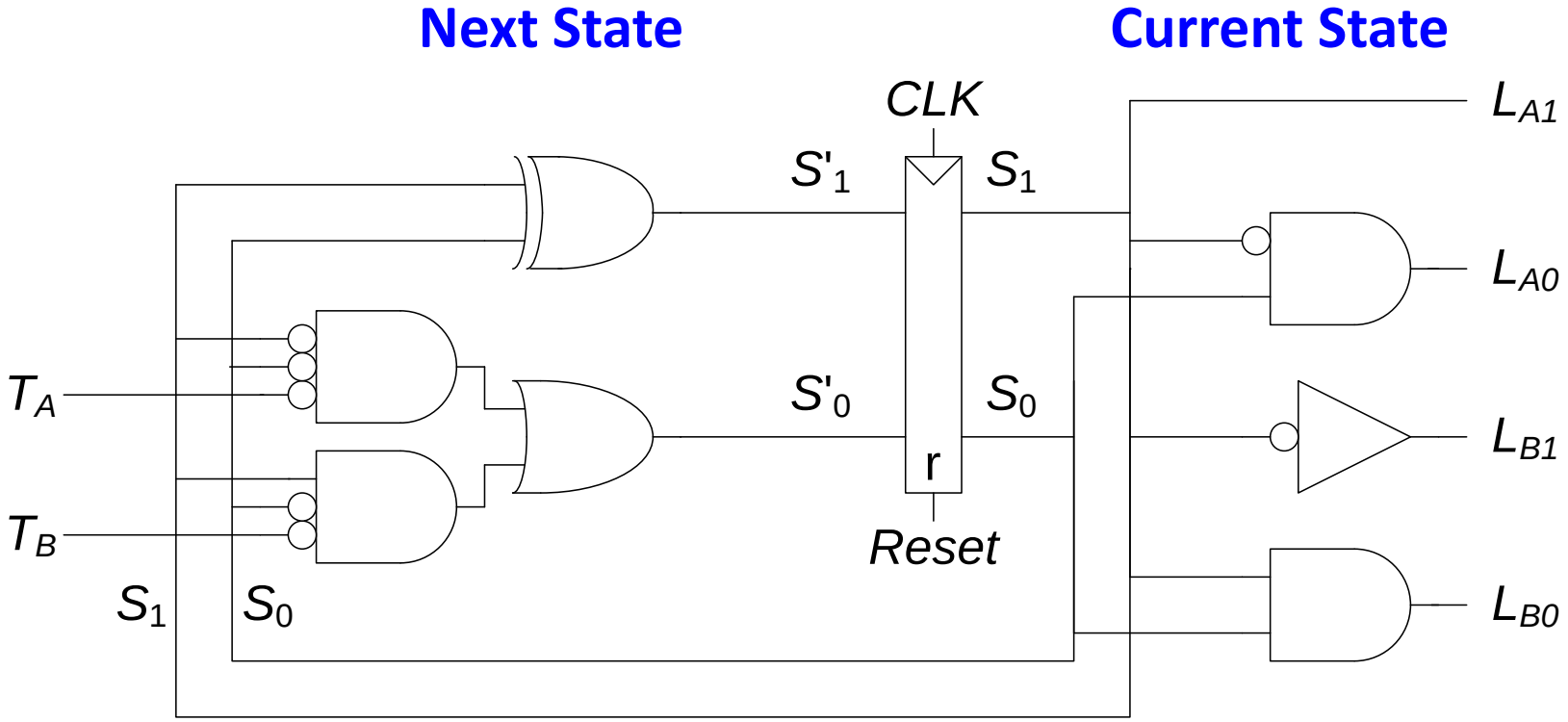
Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
S0		green		red	
S1		yellow		red	
S2		red		green	
S3		red		yellow	



Output	Encoding
green	00
yellow	01
red	10

Complement property
Replace c+c’ by 1: ab(c+c’) = ab(1).

FSM Schematic



$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = \overline{S}_1\overline{S}_0\overline{T}_A + S_1\overline{S}_0\overline{T}_B$$

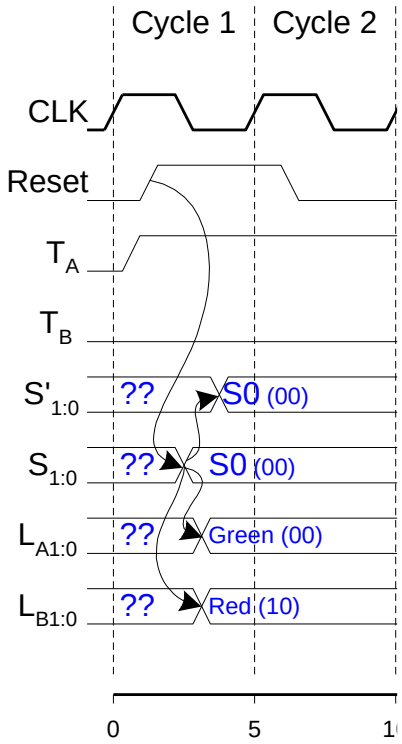
$$L_{A1} = S_1$$

$$L_{A0} = \overline{S}_1 S_0$$

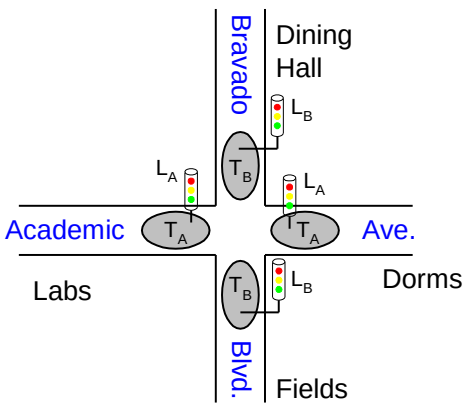
$$L_{B1} = \overline{S}_1$$

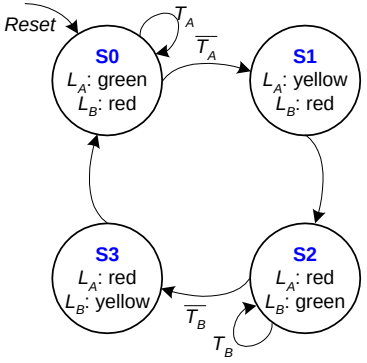
$$L_{B0} = S_1 S_0$$

FSM Timing Diagram

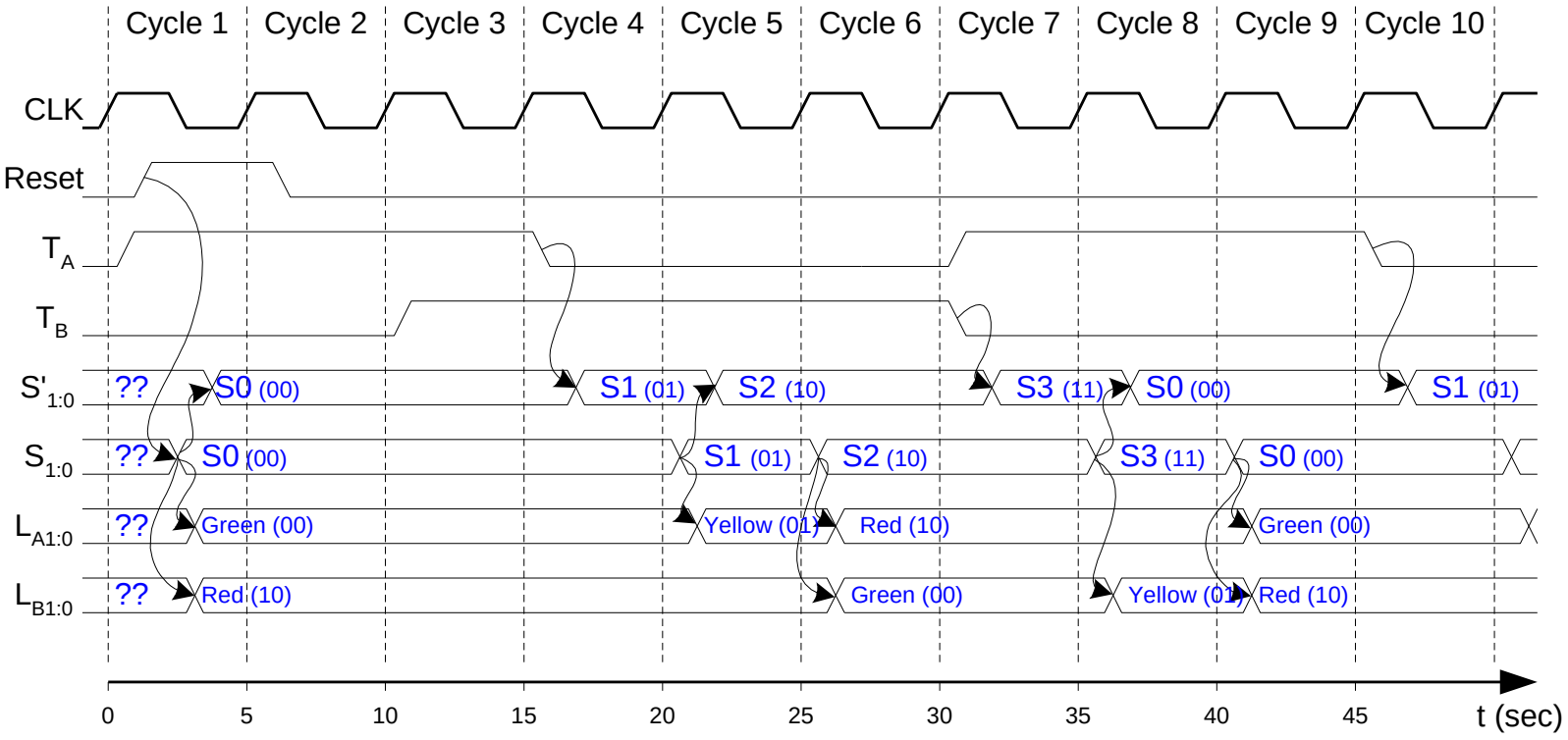


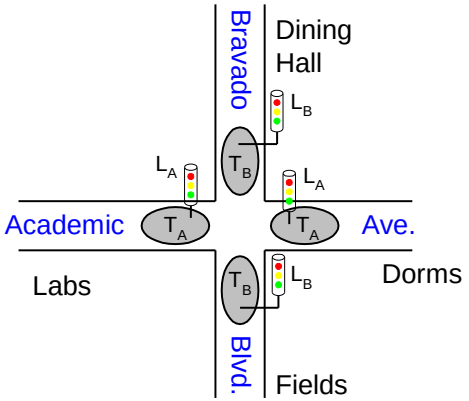
Sensor T_B data disregarded

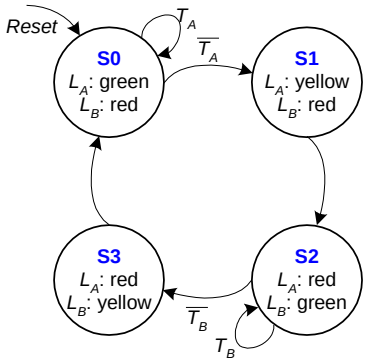




FSM Timing Diagram

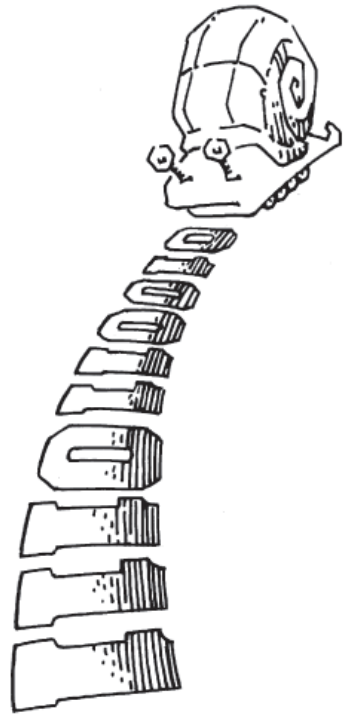






Mealy FSM Example

- Uwe has a snail that crawls down a paper tape with 1's and 0's on it.
- The snail smiles whenever the last two digits it has crawled over are 01.
- Design Mealy FSM of the snail's brain.
  - **Mealy FSM:** outputs depend on current state and inputs



- How many states do we need?
- How many bits do we need to represent the states?
- How many bits do we need for output?

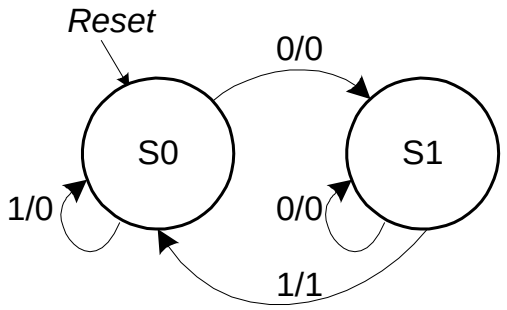
Mealy State Transition and Output Table

Current State	Input	Next State	Output
S_0	A	S'_0	Y
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	1

State	Encoding
S0	0
S1	1

$$S_0' = \overline{A}$$

$$Y = S_0 A$$



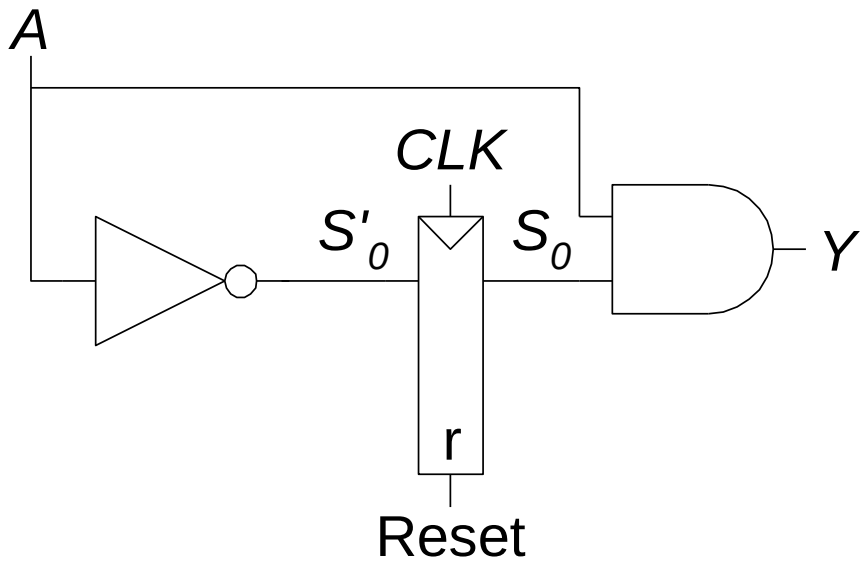
Mealy FSM Schematic

Next State Equation

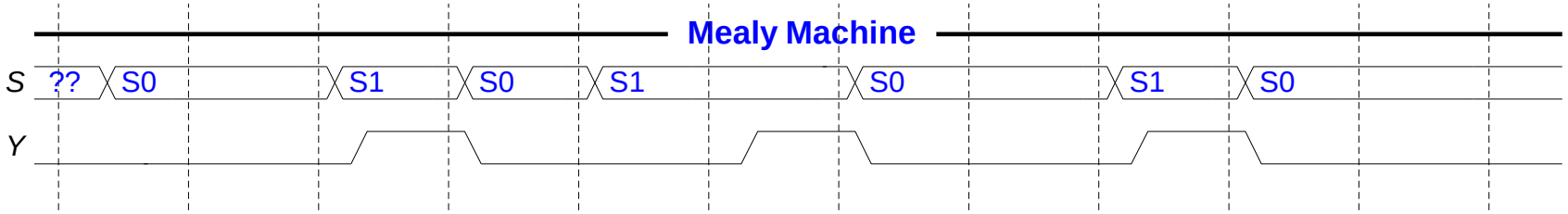
$S_0' = \overline{A}$

Output Equation

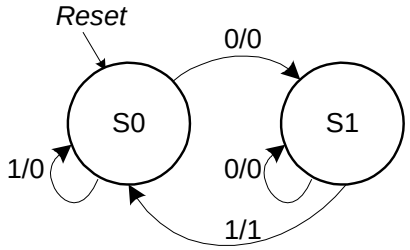
$Y = S_0 A$



Mealy Timing Diagram



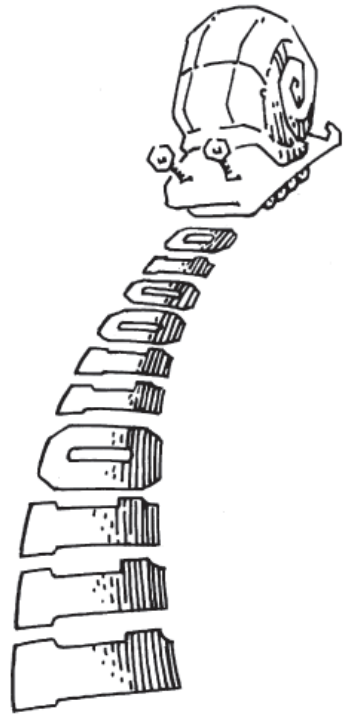
Mealy FSM



Mealy FSM: asserts Y **immediately** when input pattern 01 is detected

Moore FSM Example

- Uwe has a snail that crawls down a paper tape with 1's and 0's on it.
- The snail smiles whenever the last two digits it has crawled over are 01.
- Design Moore FSM of the snail's brain.
  - **Moore FSM:** outputs depend only on current state



- Can we use the same state transition table from Mealy FSM?
- How many states do we need?
- How many bits do we need to represent the states?
- How many bits do we need for output?

Moore Transition Table

Current State		Inputs	Next State	
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

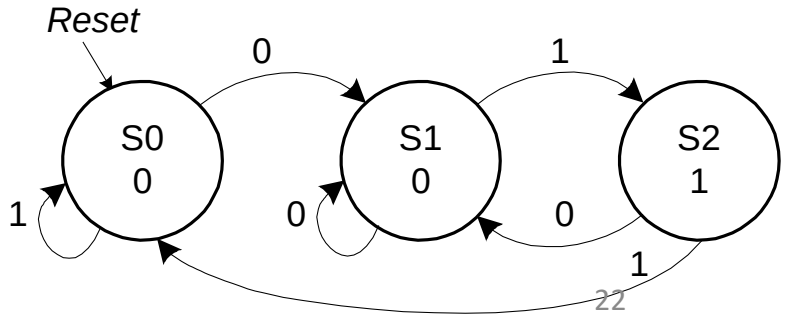
State	Encoding
S0	00
S1	01
S2	10

Can we optimize it?

$$S_1' = \overline{S_1} S_0 A$$

$$S_0' = \overline{A}$$

Moore FSM



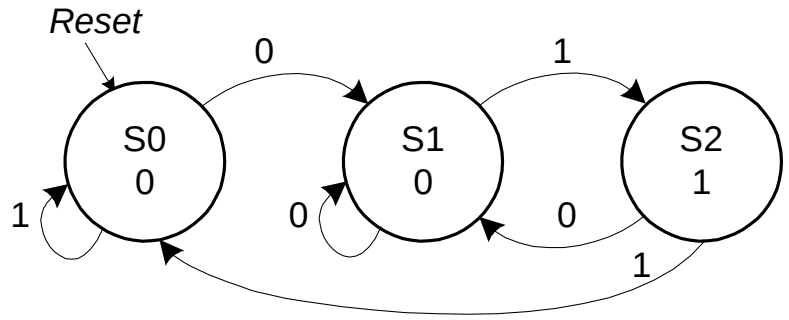
Moore FSM Output Table

Current State		Output
S_1	S_0	Y
0	0	0
0	1	0
1	0	1

State	Encoding
S0	00
S1	01
S2	10

$$Y = S_1 \overline{S_0}$$

Moore FSM



Moore FSM Schematic

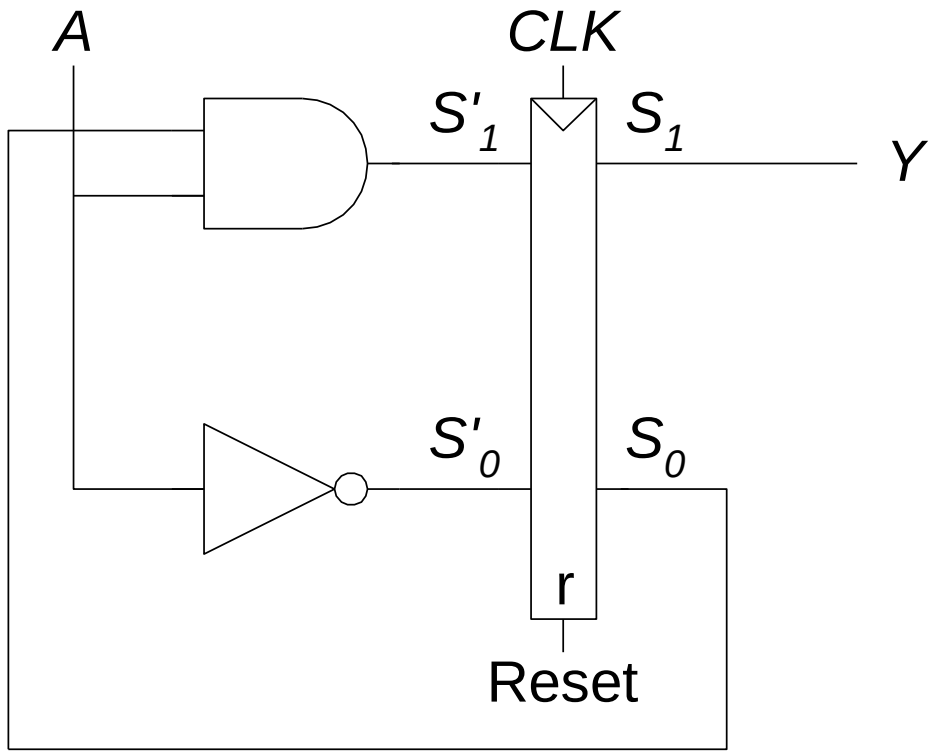
Next State Equations

$S_1' = S_0 A$

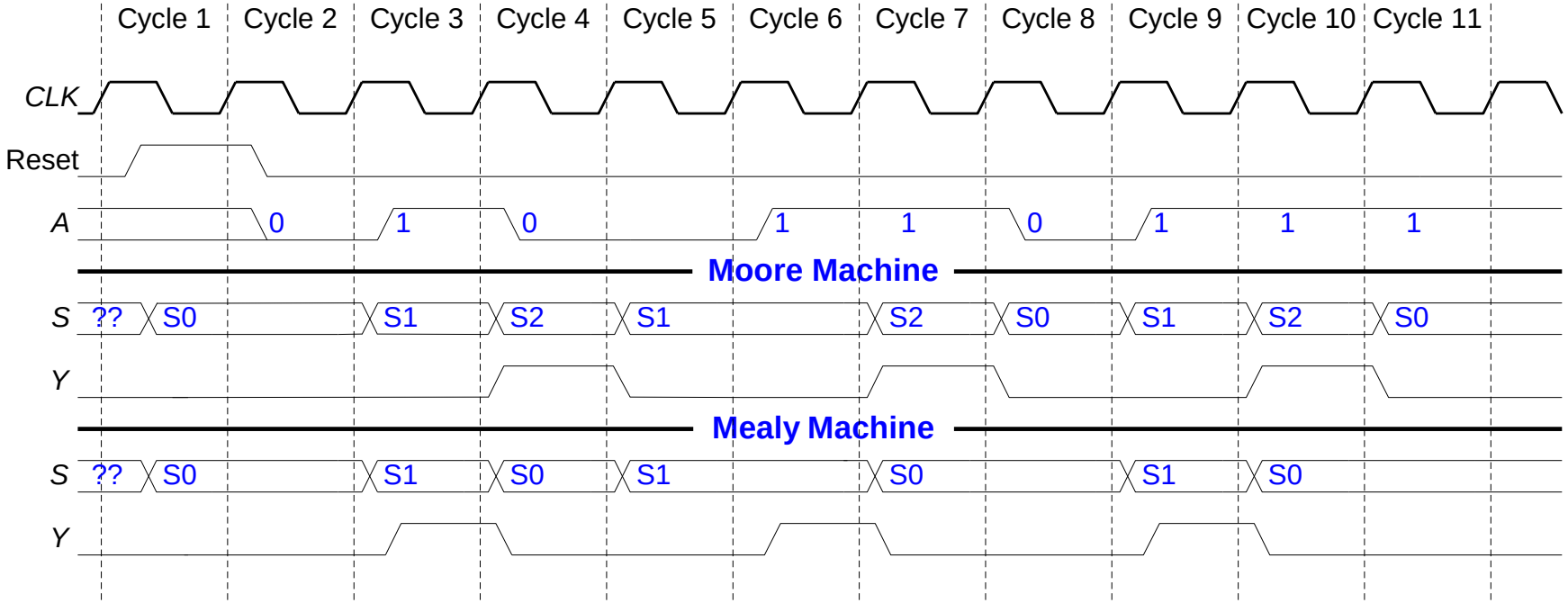
$S_0' = \overline{A}$

Output Equation

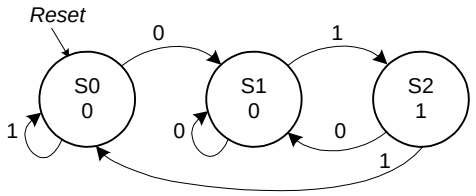
$Y = S_1$



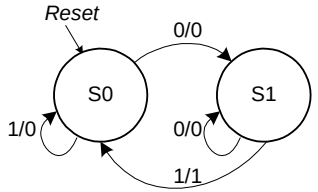
Moore and Mealy Timing Diagram



Moore FSM



Mealy FSM



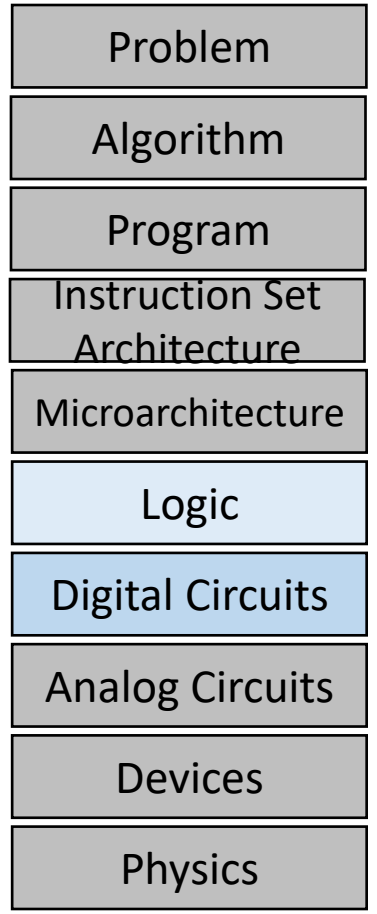
Mealy FSM: asserts Y **immediately** when input pattern 01 is detected

Moore FSM: asserts Y one cycle **after** input pattern 01 is detected

Universität Bremen

Logic Design
Finite State Machines

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.