Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

GTI – Bonus

VHDL - EXTRA

VHDL - Moore

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Beschreibung

- Gegeben seien die Moore- (Abbildung 1) und Mealy-Automaten (Abbildung 2) der Armbanduhr aus Übungsblatt 11.

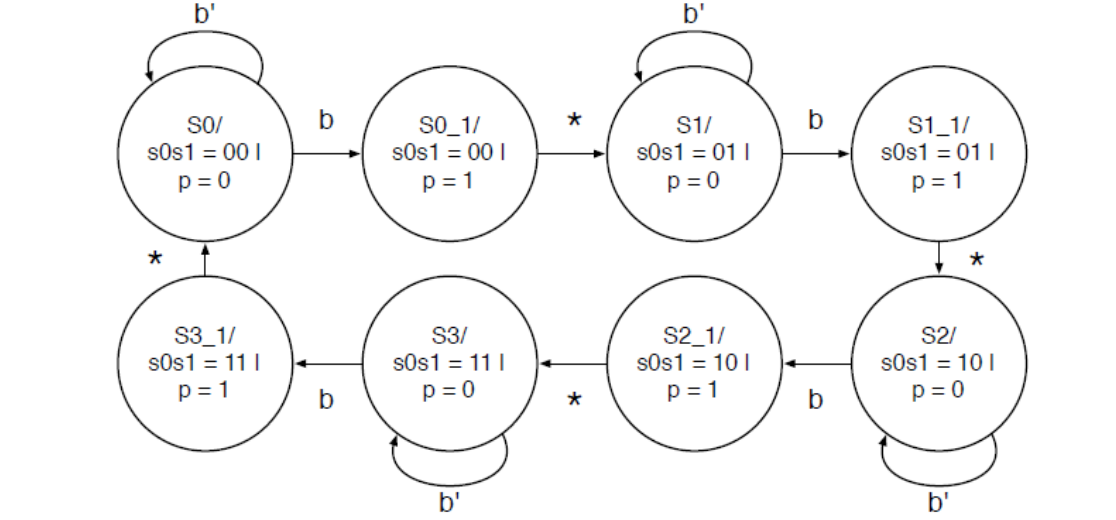


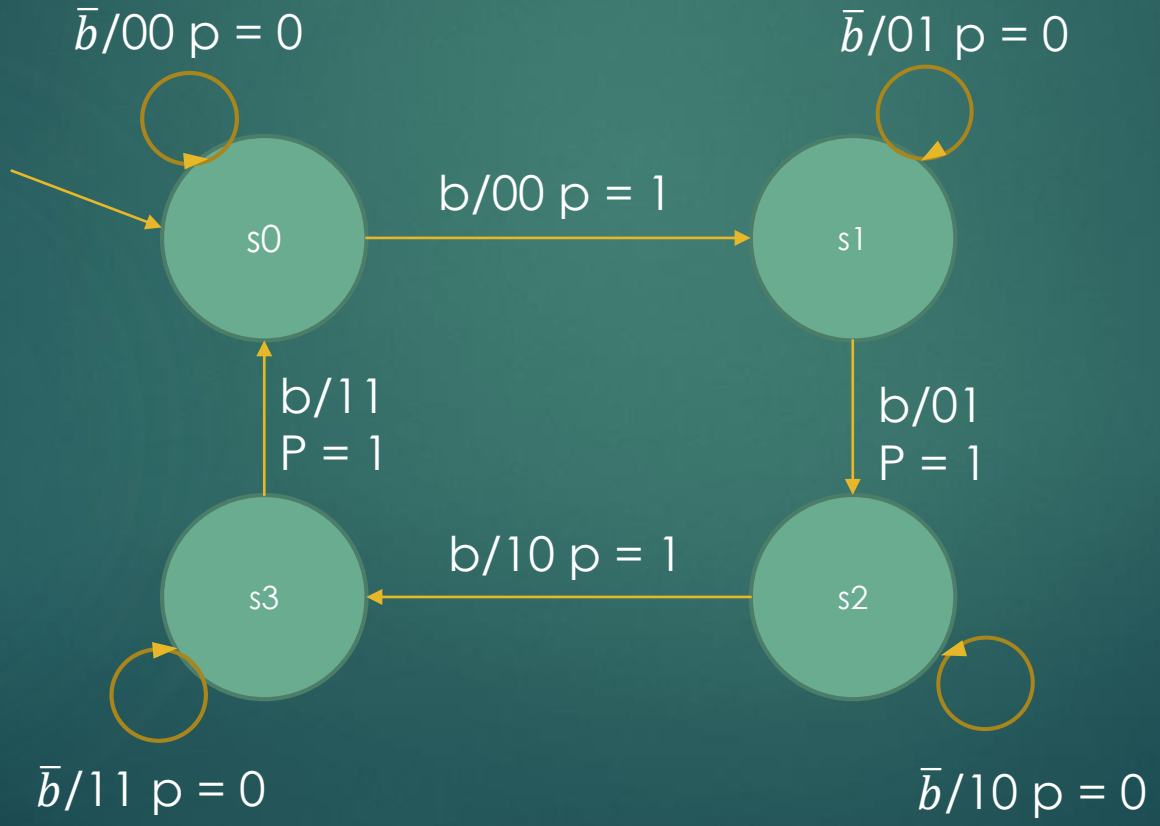
Abbildung 1: Zustandsautomat als Moore-Automat.

VHDL - Moore

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Beschreibung

- Gegeben seien die Moore- (Abbildung 1) und Mealy-Automaten (Abbildung 2) der Armbanduhr aus Übungsblatt 11.



VHDL - Moore

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Beschreibung

- Diese sollen nun exemplarisch in VHDL implementiert werden. Als Anfangszustand soll die aktuelle Uhrzeit angegeben werden. Mittels eines taktsynchronen Reset-Signals soll der Automat in den Anfangszustand versetzt werden können.
- Beschränken Sie sich bei der Umsetzung als case-Anweisung dabei auf die ersten beiden Zustände des jeweiligen Automaten.

VHDL - Moore

a. Implementieren Sie den oben entworfenen Moore-Automat indem Sie die einzelnen Zustände explizit als eigenen Typ definieren.

Hinweis: Typ wird durch type angegeben

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL - Moore

a. Implementieren Sie den oben entworfenen Moore-Automat indem Sie die einzelnen Zustände explizit als eigenen Typ definieren.

Entity:


```
entity  uhr is
port(
        i_clk, i_rst, i_b       :   in      std_logic;
        o_p                     :   out     std_logic
);
end entity uhr;
```

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL - Moore

a. Implementieren Sie den oben entworfenen Moore-Automat indem Sie die einzelnen Zustände explizit als eigenen Typ definieren.

Architecture:


```
architecture   moore   of   uhr    is
--interne Signale
signal  p           :   std_logic;
signal  control     :   std_logic_vector(0 to 1);
--Einbindungen von anderen Entitys
begin
--eigentlicher Code
end architecture moore;
```

1-Bit Signal

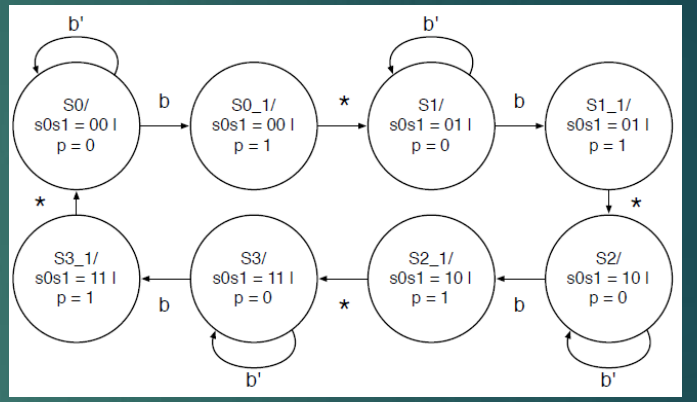
|Kodierung des Zustandes

Bitvektor (d.h. mehrere Bits)
Vgl. mit einem Array (control[0] & control[1])
Also zwei Bit breites Wort

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL - Moore

a. Implementieren Sie den oben entworfenen Moore-Automat indem Sie die einzelnen Zustände explizit als eigenen Typ definieren.




```
architecture   moore   of   uhr    is
signal  p              :   std_logic;
signal  control        :   std_logic_vector(0 to 1);
type moore_states  is  (s0, s0_1, s1, s1_2, s2, s2_3
                        s3, s3_0);
signal  state          :   moore_states;
begin
end architecture moore;
```

Definition eines Signaltyps

Nimmt neuen Typ an

WICHTIG:
Eine Möglichkeit in der Klausur seine Zustände darzustellen!

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL - Moore

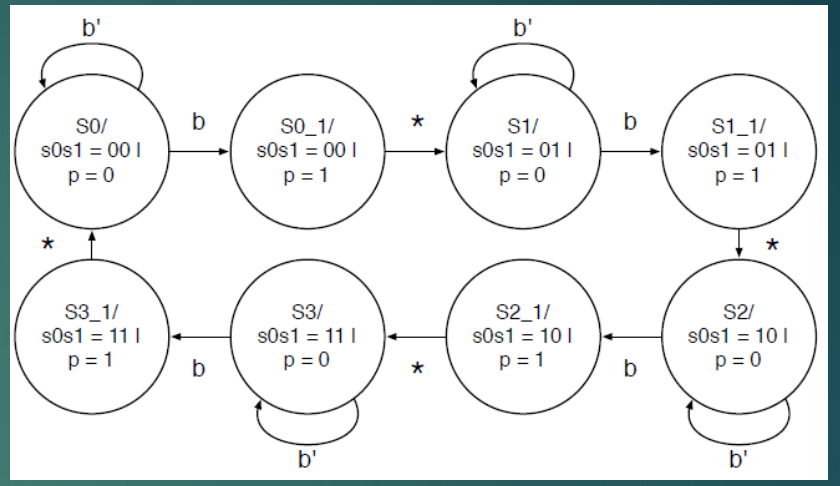
Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

```
architecture moore of uhr is
[…]
begin
FSM : process(i_clk)
begin
    if rising_edge(i_clk) then
        [sequentieller Code]
    end if;
end process FSM;
end architecture moore;
```

Optionaler Name

Sensitivitätsliste

Bei jeder steigenden Flanke von i_clk



process: Immer wenn sich ein Signal der Sensitivitätsliste ändert, wird der process durchlaufen

VHDL - Moore

- Mittels eines taktsynchronen Reset-Signals soll der Automat in den Anfangszustand versetzt werden können.

```
[…]
if  rising_edge(i_clk)          then
     if i_rst = '1' then
        state <= s0;
        control <= „00“;
        p <= '0';
    else
        […]
    end if;
end if;
[…]
```

Wertzuweisung bei Signalen mit <=;
eigene Typen ohne Anführungszeichen

Bitvektorzuweisung mit “…“

Std_logic-Zuweisung mit ‘…‘

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

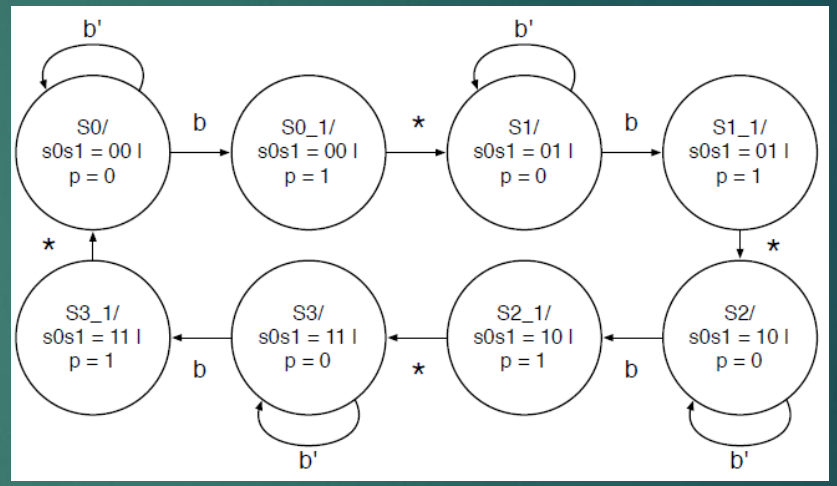
VHDL - Moore

- Zustandsübergang für die ersten beiden Zustände

```
[…]
    else
        case state is
            when s0 =>
                control <= „00“;
                p <= '0';
                if b = '1' then
                    state <= s0_1;
                else
                    state <= s0;
                end if;
            […]
        end case;
    end if;
end if;
```

Typ nach dem unterschieden werden soll

Werteabfrage

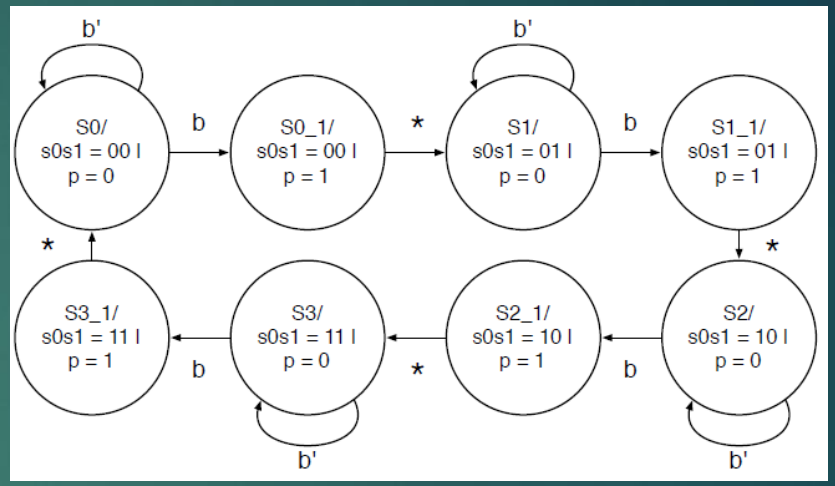


Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL - Moore

- Zustandsübergang für die ersten beiden Zustände

```
[…]
    else
        case state is
            when s0 =>
                […]
            when s0_1 =>
                control <= „00“;
                p <= '1';
                state <= s1;
            when s1 =>
                […]
        end case;
    end if;
end if;
```

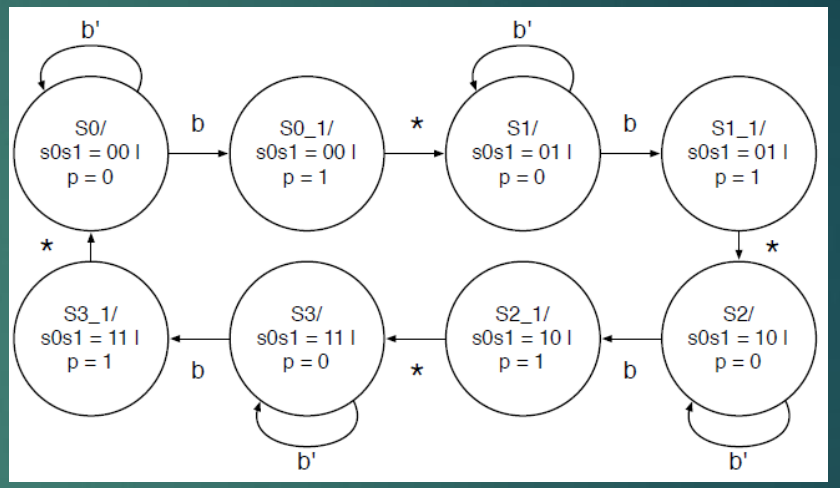


Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL - Moore

- Zustandsübergang für die ersten beiden Zuständen

```
[…]
    else
        case state is
            when s0 =>
                […]
            when s0_1 =>
                […]
            when s1 =>
                -- Folgezustände analog (Musterlösung online -> Klausurvorbereitung)
        end case;
    end if;
end if;
```



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL - Moore


```
architecture moore  of   uhr   is
[…]
begin
FSM : process(i_clk)
begin
    if rising_edge(i_clk)      then
        [sequentieller Code]
    end if;
end process FSM;
o_p <= p;
end architecture moore;
```

Unser internes p, wird auf das externe p des Entity gelegt

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL – One-Hot

b. Implementieren Sie den oben entworfenen Mealy-Automat indem Sie die einzelnen Zustände mittels einer one-hot-Kodierung definieren.

Hinweis: One-Hot-Codierung: z.B. 1000, 0100, 0010, 0001

d.h. wir haben eine wandernde Eins

für jeden Zustand ein Bit

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL – One-Hot


```
architecture   mealy   of   uhr    is

constant    s0  :   std_logic_vector(0 to 3)   := „1000“;
constant    s1  :   std_logic_vector(0 to 3)   := „0100“;
constant    s2  :   std_logic_vector(0 to 3)   := „0010“;
constant    s3  :   std_logic_vector(0 to 3)   := „0001“;

signal  state           :   std_logic_vector(0 to 3);
signal  state_next      :   std_logic_vector(0 to 3);
signal  control         :   std_logic_vector(0 to 1);

begin
[…]
end architecture mealy;
```

Konstantenzuweisung:
Einmalig (!) mit ‘:=‘

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL – One-Hot


```
architecture mealy of uhr is
[...]
begin
next_state_logic : process (i_clk)
begin
    if rising_edge(i_clk) then
        if i_rst = '1' then
            state <= s0;
        else
            state <= state_next;
        end if;
    end if;
end process next_state_logic;
```

Synchroner Prozess für jeden Takt

Synchrones Reset

Synchroner Übergang zum nächsten Zustand

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL – One-Hot


```
[…]
end process;
FSM : process(state, b)
begin
    control <= „00“;
    p <= '0';
    case state is
        when s0 =>
            […]
        when s1 =>
            […]
    end case;
end process FSM;
```

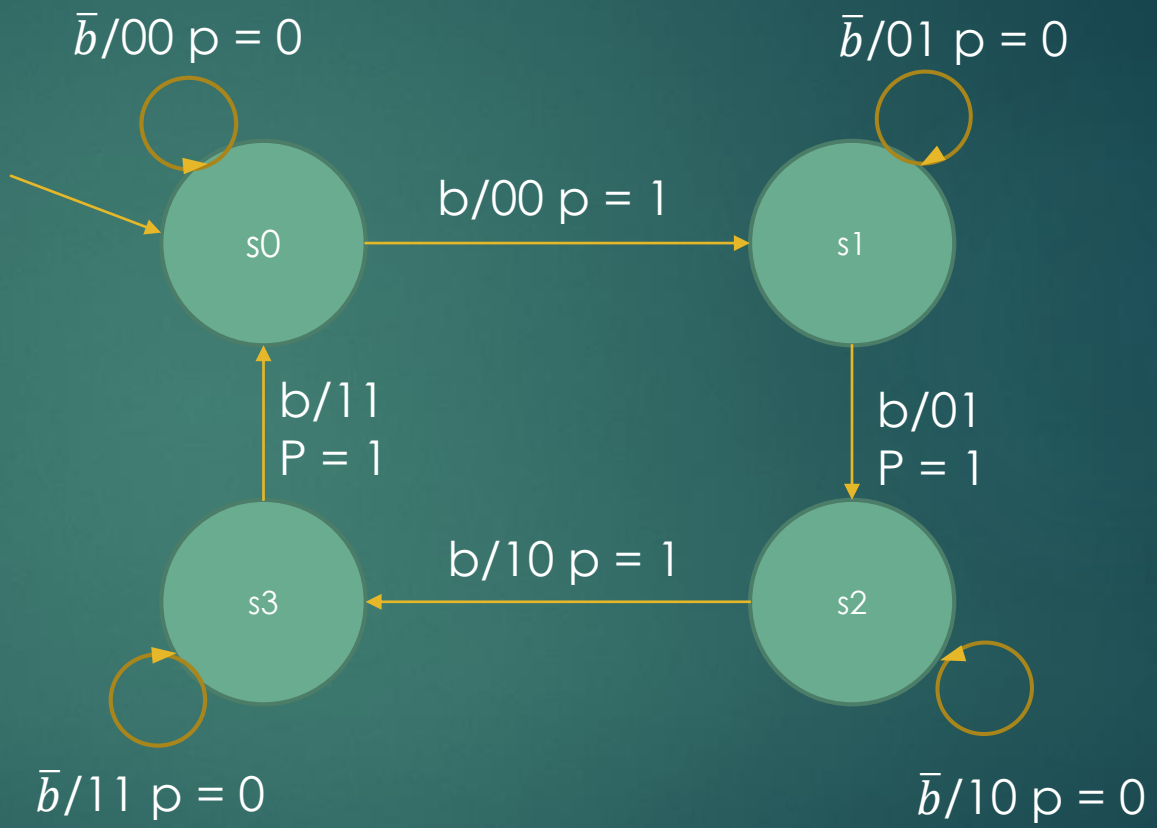
Asynchroner Prozess, welcher bei Änderung von state oder b ausgeführt wird

Zustandsübergänge

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL – One-Hot

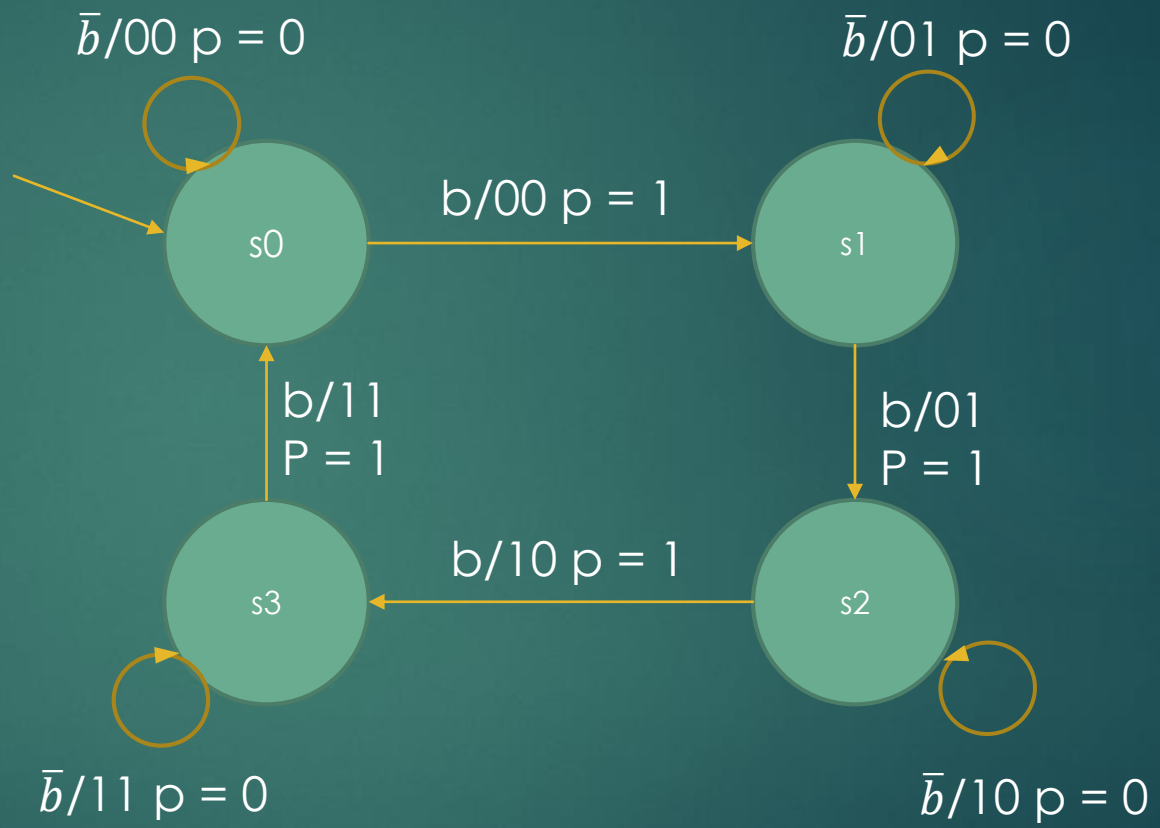

```
[…]
when s0 =>
    if b = '1' then
        control <= „01“;
        state_next <= s1;
    else
        control <= „00“;
        state_next <= s0;
    end if;
when s1 =>
[…]
end case;
```



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL – One-Hot


```
[…]
when s1 =>
    if b = '1' then
        control <= „10“;
        state_next <= s2;
    else
        control <= „01“;
        state_next <= s1;
    end if;
when s2 =>
[…]
end case;
```



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

VHDL – One-Hot


```
    […]
    end case;
end process FSM;

o_p <= p;
end architecture mealy;
```

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck