

# Übung 10, Aufgabe 2

Reaktionszeit-Spiel fuer zwei Personen

---

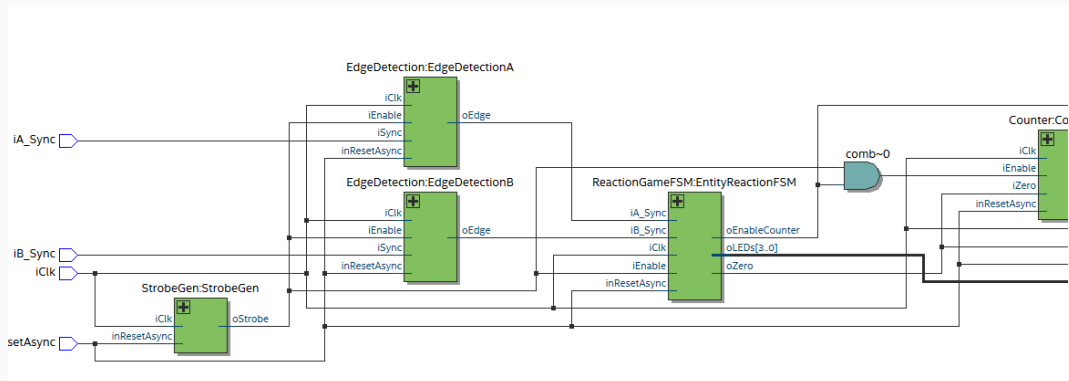
Simon Offenberger S2410306027@fhooe.at

13. Januar 2026

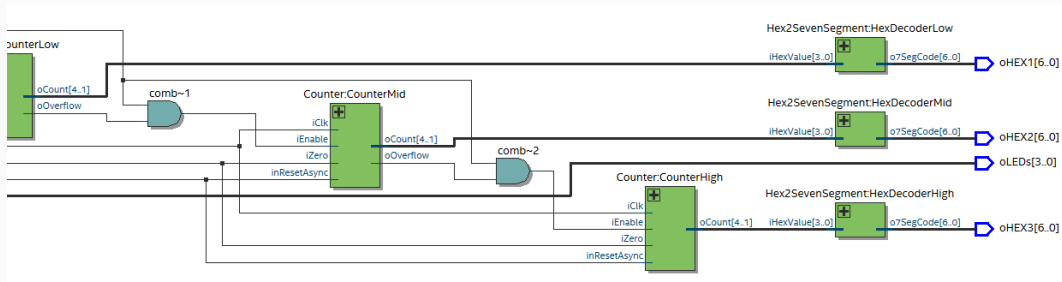
FH Hagenberg

- Decoupling - Spannungsstabilisierung
- GPIO - Schutzbeschaltung
- IR-Emitter
- Spannungsversorgung

# Struktur des Reaktionszeit-Spiels



## Struktur des Reaktionszeit-Spiels

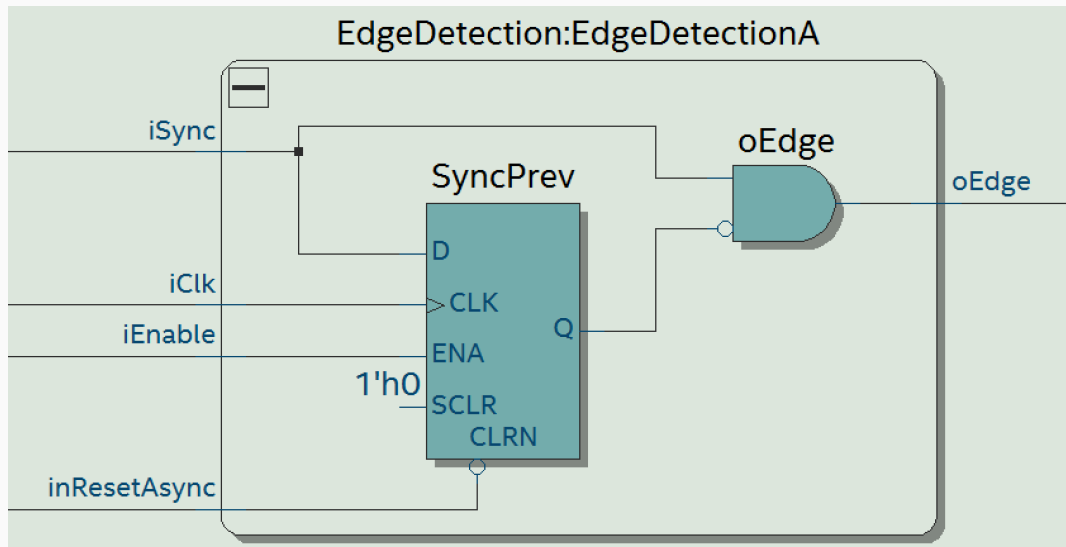


# Edge Detection

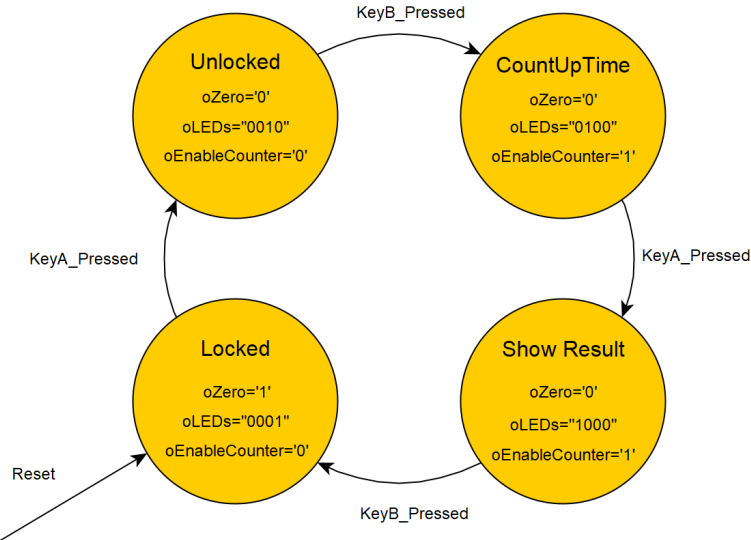
```
1  entity EdgeDetection is
2  port (
3      iClk : in std_ulogic;
4      inResetAsync : in std_ulogic;
5      iEnable : in std_ulogic;
6      iSync : in std_ulogic;
7      oEdge : out std_ulogic);
8  end EdgeDetection;
9
```

```
1  architecture RTL of EdgeDetection is
2      signal SyncPrev : std_ulogic;
3  begin
4
5  process (iClk, inResetAsync) is
6      begin
7          -- asynchronous reset
8          if (inResetAsync = not('1')) then
9              SyncPrev <= '0';
10
11         elsif (rising_edge(iClk)) then
12             if(iEnable = '1') then
13                 SyncPrev <= iSync;
14             else
15                 SyncPrev <= SyncPrev;
16             end if;
17         end if;
18     end process;
19
20     -- combinational logic for edge detection
21     oEdge <= '1' when (iSync = '1' and SyncPrev = '0') else '0';
22
23 end architecture RTL;
24
```

# Edge Detection RTL Viewer



# Finite State Machine Moore



# State Register

```
1  -- State Register
2  process (iClk, inResetAsync) is
3  begin
4      if (inResetAsync = not('1')) then
5          State <= Locked;
6      elsif (rising_edge(iClk)) then
7          if(iEnable = '0') then
8              State <= State; -- hold state when not enabled
9          else
10             State <= NextState;
11         end if;
12     end if;
13 end process;
```

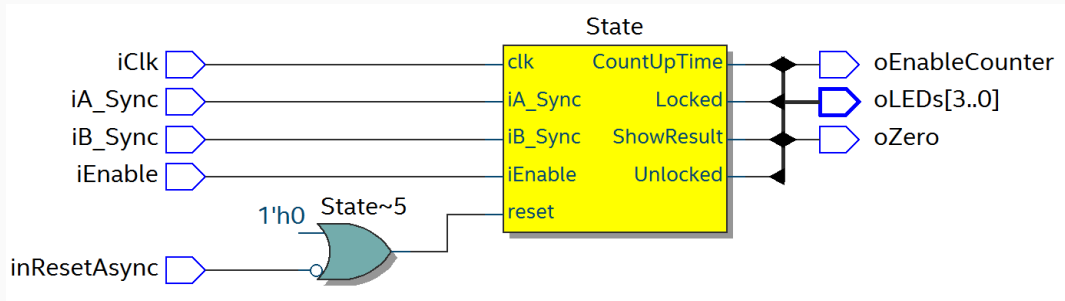


# Next State Logic

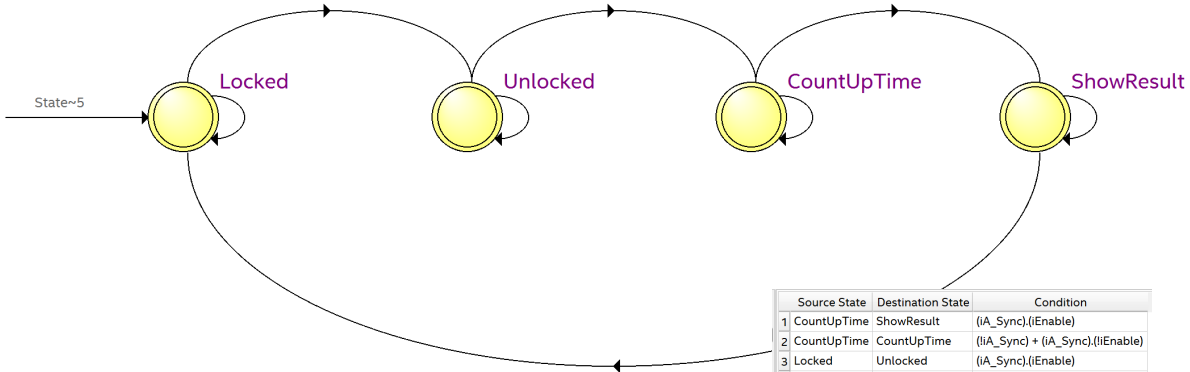
```
1  -- State Transition Process
2  NextStateLogic : process (State,iEnable,iA_Sync,iB_Sync) is
3  begin
4
5      NextState <= State;    -- default hold state
6      oLeds <= cLEDOFF;      -- default all leds off
7      oZero <= '0';          -- default zero off
8      oEnableCounter <= '0'; -- default counter disabled
9
10     case State is
11     when Locked =>
12         if(iA_Sync = '1') then
13             NextState <= Unlocked;
14         end if;
15         oZero <= '1';          -- reset counters
16         oLeds(cLED_LOCKED_INDEX) <= '1'; -- indicate locked
17                                     state
18     when Unlocked =>
19         if(iB_Sync = '1') then
20             NextState <= CountUpTime;
21         end if;
22         oLeds(cLED_UNLOCKED_INDEX) <= '1'; -- indicate locked
23                                     state
```

```
1  when CountUpTime =>
2      oEnableCounter <= '1'; -- enable counter
3      if(iA_Sync = '1') then
4          NextState <= ShowResult;
5      end if;
6      oLeds(cLED_COUNTUP_INDEX) <= '1'; -- indicate locked
7                                     state
8  when ShowResult =>
9      if(iB_Sync = '1') then
10         NextState <= Locked;
11     end if;
12     oLeds(cLED_SHOWRESULT_INDEX) <= '1'; -- indicate locked
13                                     state
14  when others =>
15      NextState <= cStateAllOff;
16  end case;
17
18 end process;
```

# Finite State Machine RTL-Viewer



# Finite State Machine State Machine Viewer



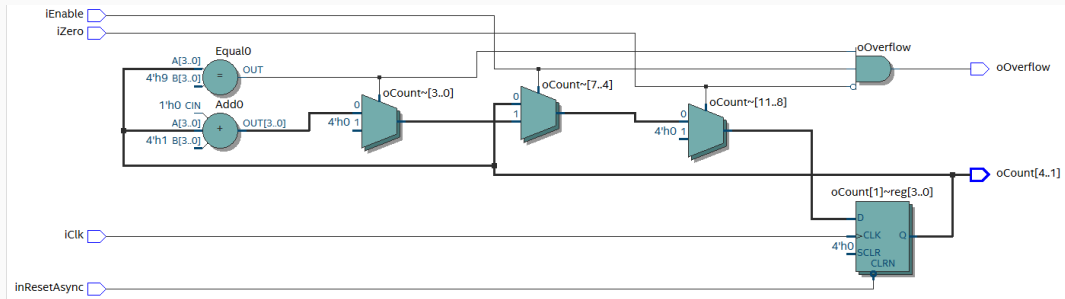
| Source State  | Destination State | Condition                         |
|---------------|-------------------|-----------------------------------|
| 1 CountUpTime | ShowResult        | (iA_Sync).(iEnable)               |
| 2 CountUpTime | CountUpTime       | (liA_Sync) + (iA_Sync).(liEnable) |
| 3 Locked      | Unlocked          | (iA_Sync).(iEnable)               |
| 4 Locked      | Locked            | (liA_Sync) + (iA_Sync).(liEnable) |
| 5 ShowResult  | ShowResult        | (liB_Sync) + (iB_Sync).(liEnable) |
| 6 ShowResult  | Locked            | (iB_Sync).(iEnable)               |
| 7 Unlocked    | CountUpTime       | (iB_Sync).(iEnable)               |
| 8 Unlocked    | Unlocked          | (liB_Sync) + (iB_Sync).(liEnable) |

# Counter

```
1 entity Counter is
2   generic(
3     gCounterOverflowVal : natural := 10
4   );
5   port (
6     iClk      : in  std_ulogic;
7     iEnable   : in  std_ulogic;
8     inResetAsync : in std_ulogic;
9     iZero     : in  std_ulogic;
10    oOverflow  : out std_ulogic;
11    oCount     : out unsigned(LogDualis(gCounterOverflowVal)
12                        downto 1));
13 end Counter;
```

```
1 architecture RTL of Counter is
2   begin
3
4     -- combinational logic for overflow output
5     oOverflow <= '1' when (iEnable = '1' and iZero = '0' and
6                          oCount = gCounterOverflowVal - 1) else '0';
7
8     process(iClk, inResetAsync) is
9       begin
10        -- asynchronous reset
11        if inResetAsync = not('1') then
12          oCount <= (others => '0');
13
14        elsif rising_edge(iClk) then
15
16          if iZero = '1' then
17            oCount <= (others => '0');
18
19          elsif iEnable = '1' then
20            if oCount = gCounterOverflowVal - 1 then
21              oCount <= (others => '0');
22            else
23              oCount <= oCount + 1;
24            end if;
25          end if;
26        end if;
```

# Counter RTL Viewer



# Hex to 7-Segment Decoder

```
1 architecture Rtl of Hex2SevenSegment is
2
3     function ToSevSeg(cValue : std_ulogic_vector(3 downto 0))
4         return std_ulogic_vector is
5     begin
6         case cValue(3 downto 0) is
7             when "0000" => return "0111111";
8             when "0001" => return "0000110";
9             when "0010" => return "1011011";
10            when "0011" => return "1001111";
11            when "0100" => return "1100110";
12            when "0101" => return "1101101";
13            when "0110" => return "1111101";
14            when "0111" => return "0000111";
15            when "1000" => return "1111111";
16            when "1001" => return "1101111";
17            when "1010" => return "1110111";
18            when "1011" => return "1111100";
19            when "1100" => return "0111001";
20            when "1101" => return "1011110";
21            when "1110" => return "1111001";
22            when "1111" => return "1110001";
23            when others => return "XXXXXXX";
24        end case;
25    end ToSevSeg;
```

```
1 begin
2
3     o7SegCode <= ToSevSeg(iHexValue);
4
5 end Rtl;
6
```

## Flow Summary

 <<Filter>>

|                             |  |
|-----------------------------|--|
| Flow Status                 | Successful - Tue Jan 13 09:35:01 2026          |
| Quartus Prime Version       | 23.1std.1 Build 993 05/14/2024 SC Lite Edition |
| Revision Name               | ReactionGameOnDE1SOC                           |
| Top-level Entity Name       | ReactionGameOnDE1SOC                           |
| Family                      | Cyclone V                                      |
| Device                      | 5CSEMA5F31C6                                   |
| Timing Models               | Final  |
| Logic utilization (in ALMs) | 36 / 32,070 ( < 1 % )                          |
| Total registers             | 39   |
| Total pins                  | 29 / 457 ( 6 % )                               |
| Total virtual pins          | 0  |
| Total block memory bits     | 0 / 4,065,280 ( 0 % )                          |

- SyncStage: 2 x 2 Flip-Flops
- EdgeDetection: 2 x 1 Flip-Flops
- StrobeGen: 17 Flip-Flops
- FSM: 4 Flip-Flops
- Counter: 3 x 4 Flip-Flops



Fragen?