# MIKROPROCESORY PRO VÝKONOVÉ SYSTÉMY

## Logické obvody - sekvenční

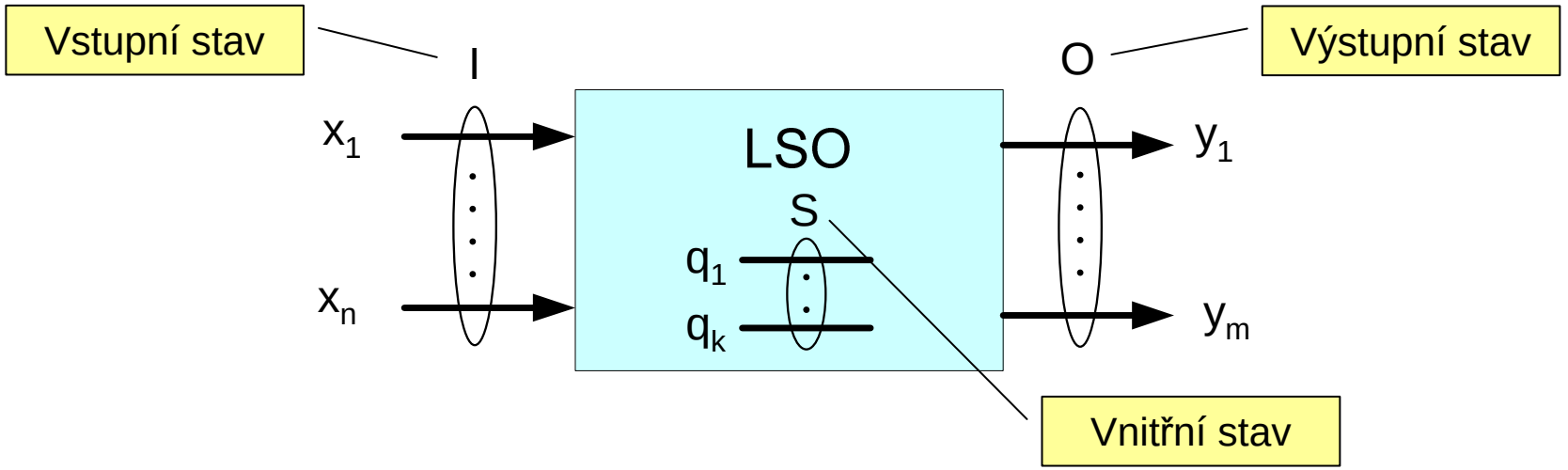
## Formy popisu, konečný automat

## Příklady návrhu

České vysoké učení technické Fakulta elektrotechnická

# Logický sekvenční obvod

- Logický sekvenční obvod (LSO) popsán stavovým diagramem, rovnicemi, tabulkami, HDL jazykem
- Vstupní, výstupní i vniřní ($q_i$) proměnné nabývají pouze hodnot 0 nebo 1




- Hodnoty všech výstupních proměnných jsou v každém časovém okamžiku určeny hodnotami vstupních proměnných ve stejném okamžiku ale též hodnotami vstupních proměnných předcházejících (LSO má vnitřní paměť)

# Konečný automat (FSA), přechodová a výstupní funkce

- Abstraktní model sekvenčního obvodu – konečný automat (FSA – Finite State Automaton nebo také FSM – Finite State Machine)
- Konečný automat – konečný počet vstupních, vnitřních a výstupních stavů
- Danou kombinaci vstupních, vnitřních a výstupních proměnných nazveme vstupní, vnitřní a výstupní stav a označíme $I_i, S_k, O_j$

- Přechodová funkce:

$$S^{t+1} = f(S^t, I^t)$$

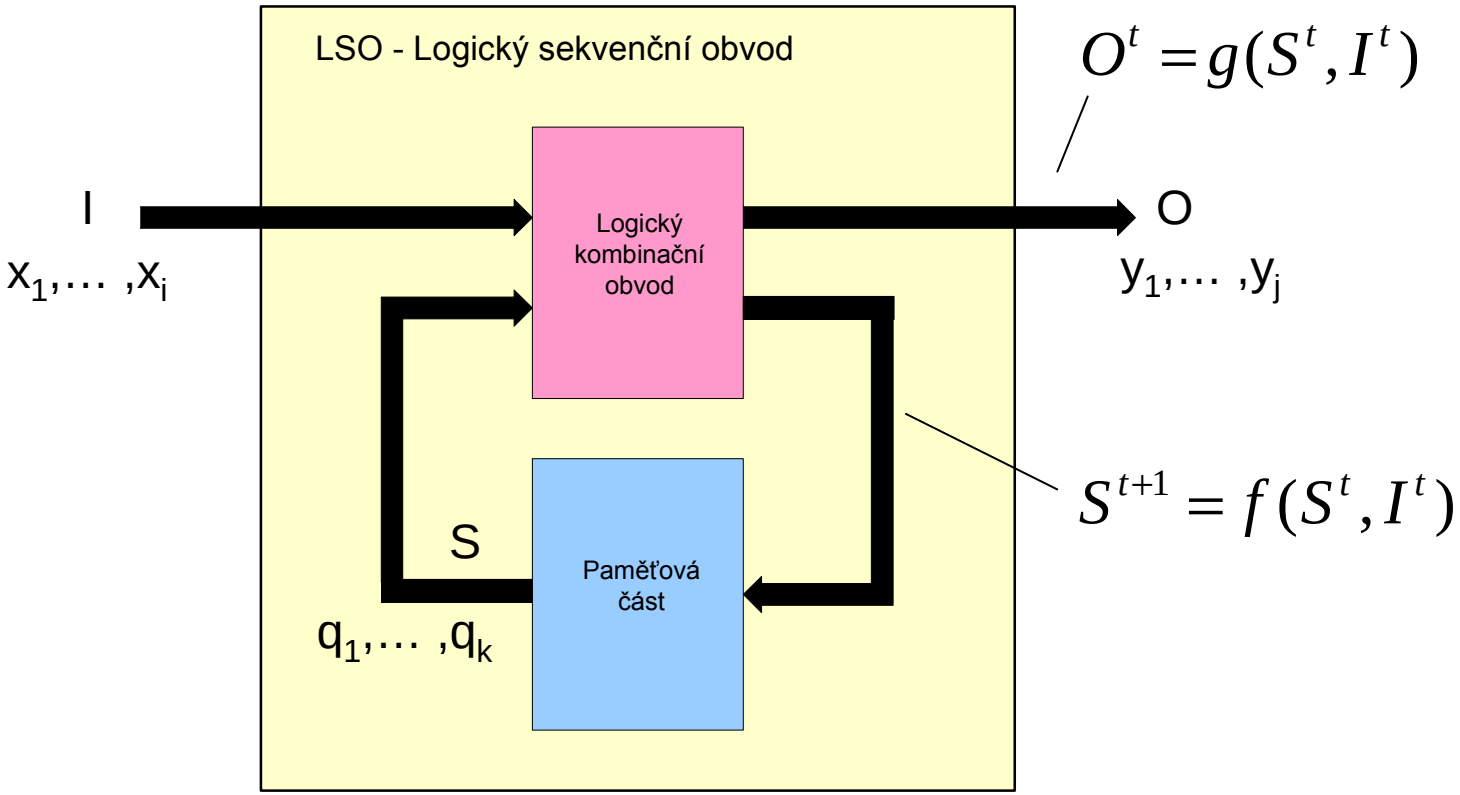
- $S^{t+1}, S^t, I^t$ – vnitřní následující, vnitřní a vstupní současný stav

- Výstupní funkce:

$$O^t = g(S^t, I^t)$$

- $O^t, S^t, I^t$ – výstupní, vnitřní a vstupní současný stav

# Obecný model logického sekvenčního obvodu (Huffmann)



# Rozdělení LSO

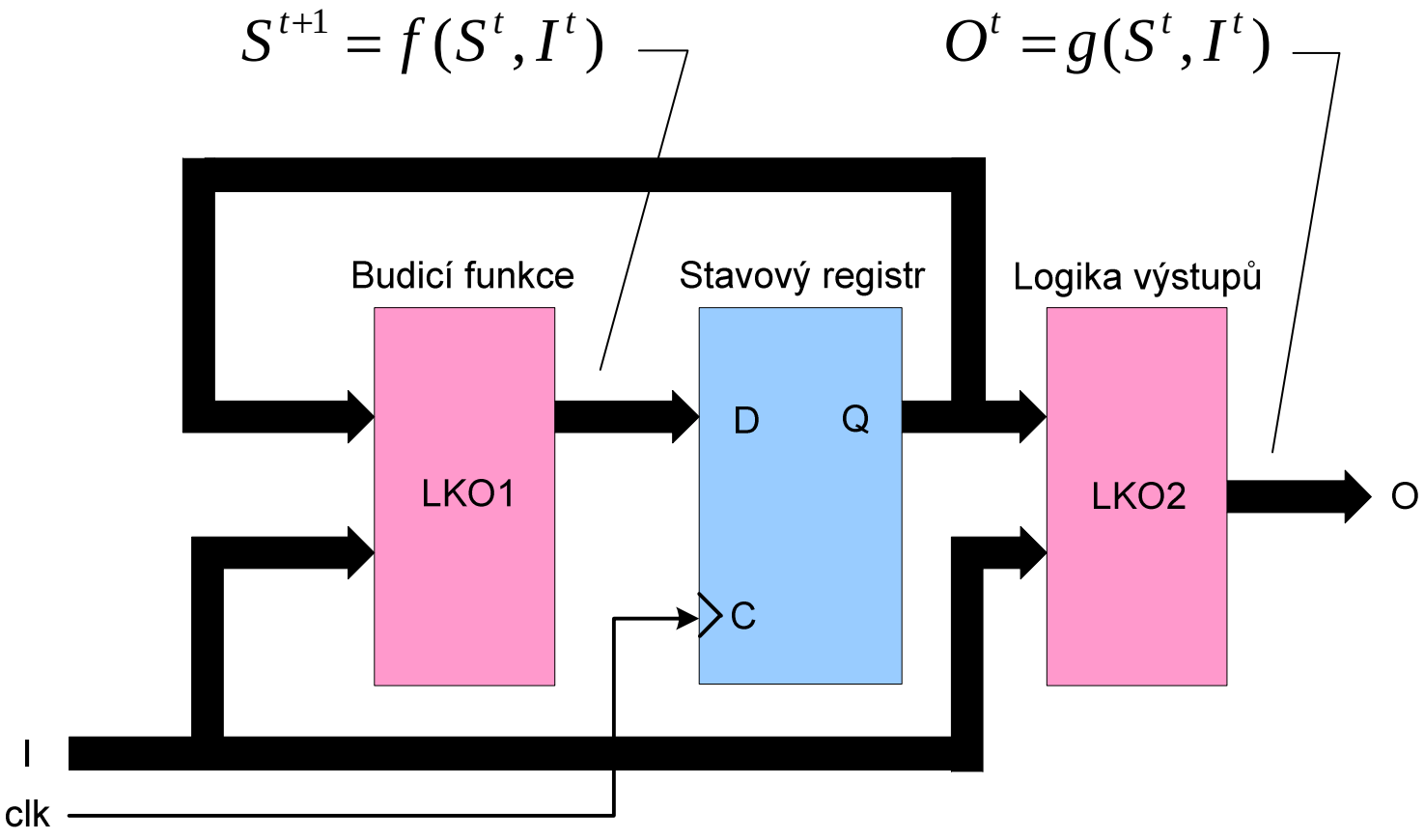
- Podle časové reakce na změnu vstupních proměnných
  - Asynchronní – změna stavu LSO po změně vstupních proměnných ihned (resp. s malým zpožděním v důsledku reakce vnitřních obvodů LSO).
  - Synchronní – změna stavu LSO synchronizována vnějšími synchronizačními impulsy (tzv. hodiny, clock). LSO jsou navrhovány většinou jako synchronní – snadnější kontrola vnitřních signálů LSO.

- Podle způsobu výpočtu přechodové a výstupní funkce

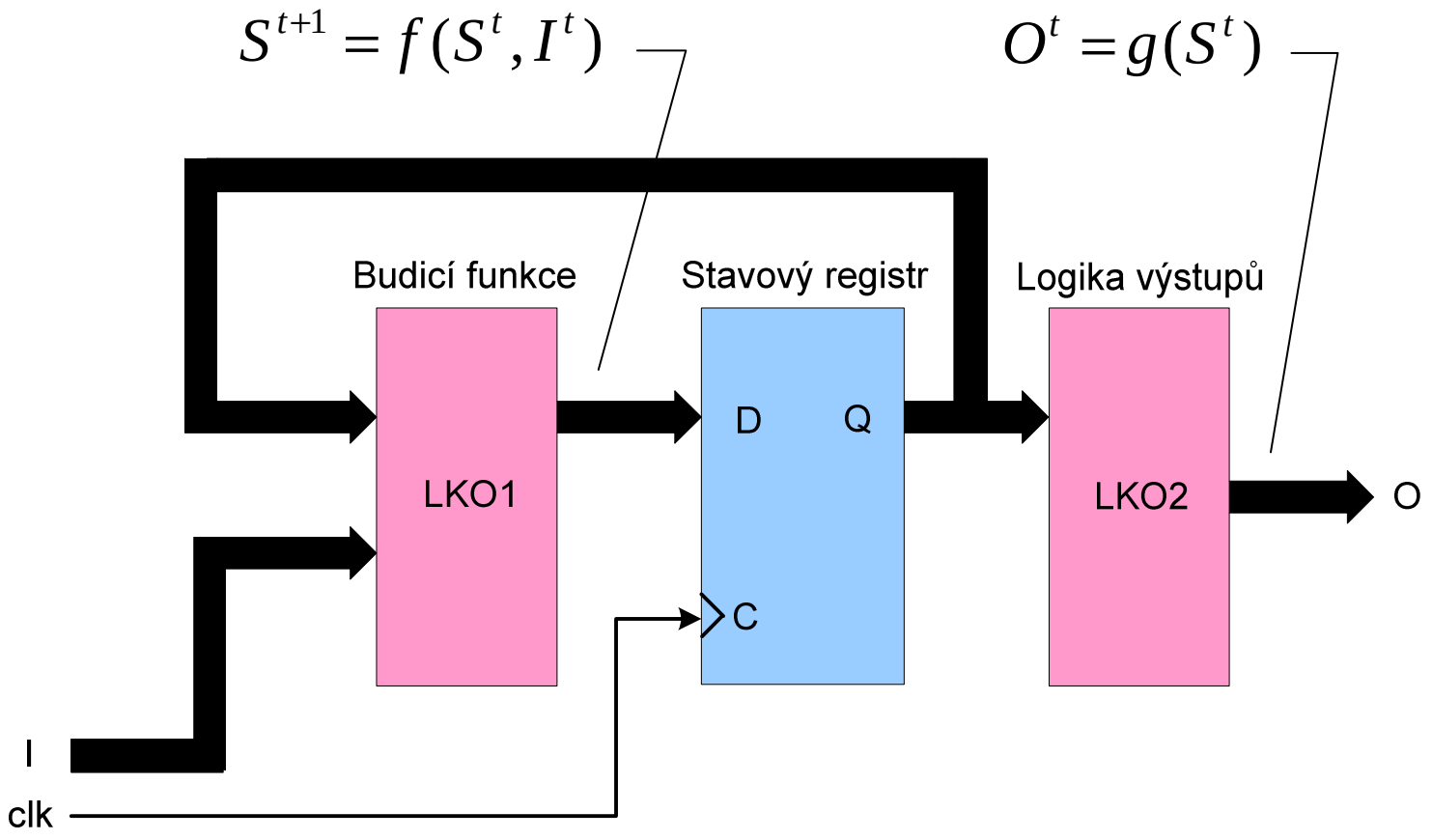
LKO

  - Automat typu Mealy → $S^{t+1} = f(S^t, I^t) \quad O^t = g(S^t, I^t)$
  - Automat typu Moore → $S^{t+1} = f(S^t, I^t) \quad O^t = g(S^t)$
  - Autonomní automat → $S^{t+1} = f(S^t) \quad O^t = g(S^t)$
    (např. čítače,… )

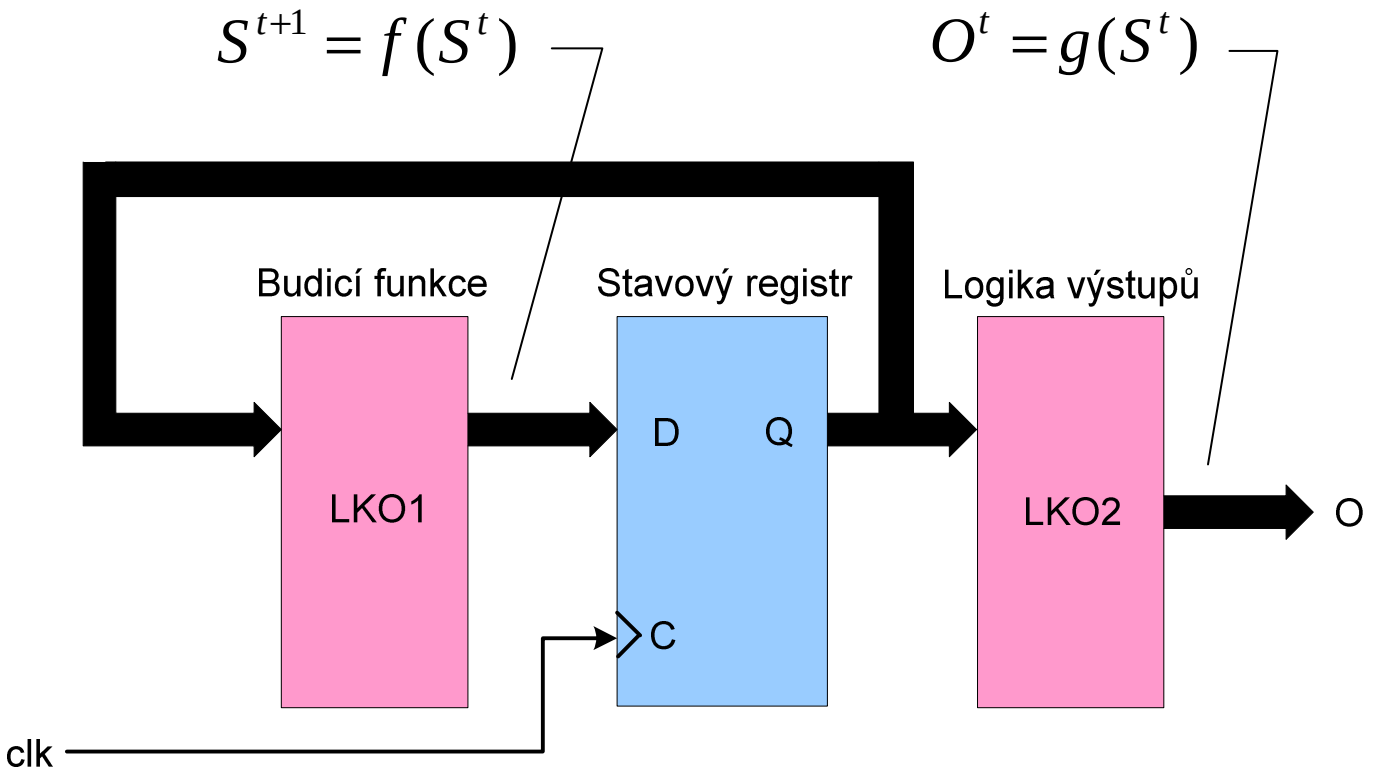
# Synchronní FSA – typu Mealy



# Synchronní FSA – typu Moore



# Synchronní FSA – autonomní

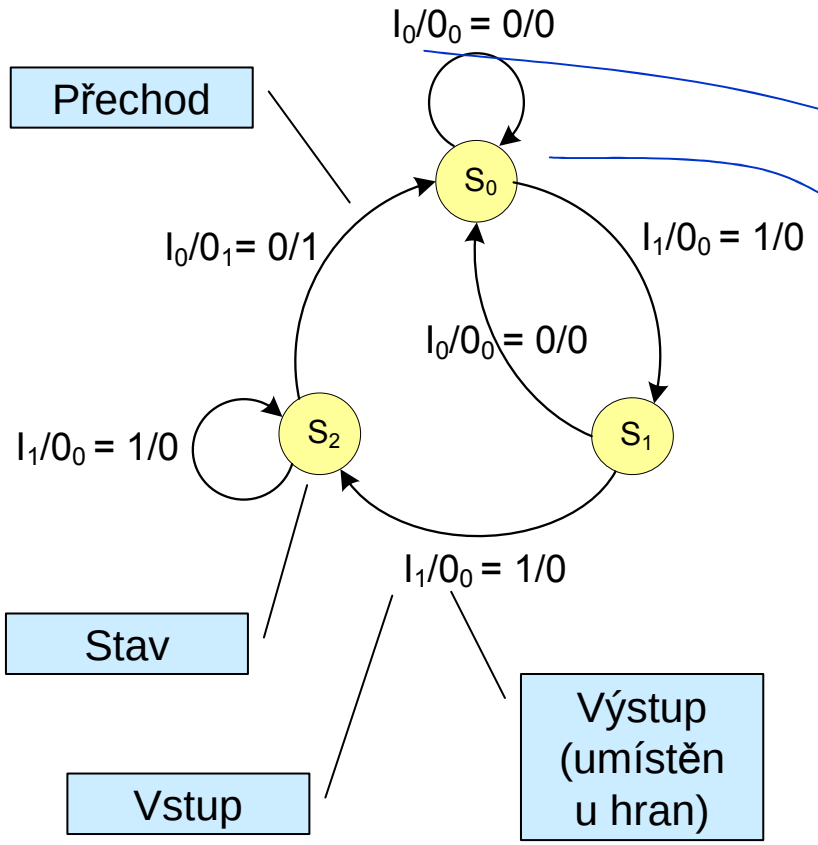


# Forma popisu LSO (FSA)

- Stavovým diagramem (State diagram) – forma orientovaného grafu
- Soustavou rovnic
- Tabulkami přechodů a výstupů
- Některým programovacím jazykem
- Vyjmenováním všech posloupností vstupů a výstupů (TO NE)
  nepraktické – nepoužívá se, vstupní posloupnost může být i nekonečné délky

# Formy popisu FSA – typu Mealy

## Stavový diagram




## Přechodová a výstupní funkce

Současný výskyt stavu a vstupu – „$\alpha$“

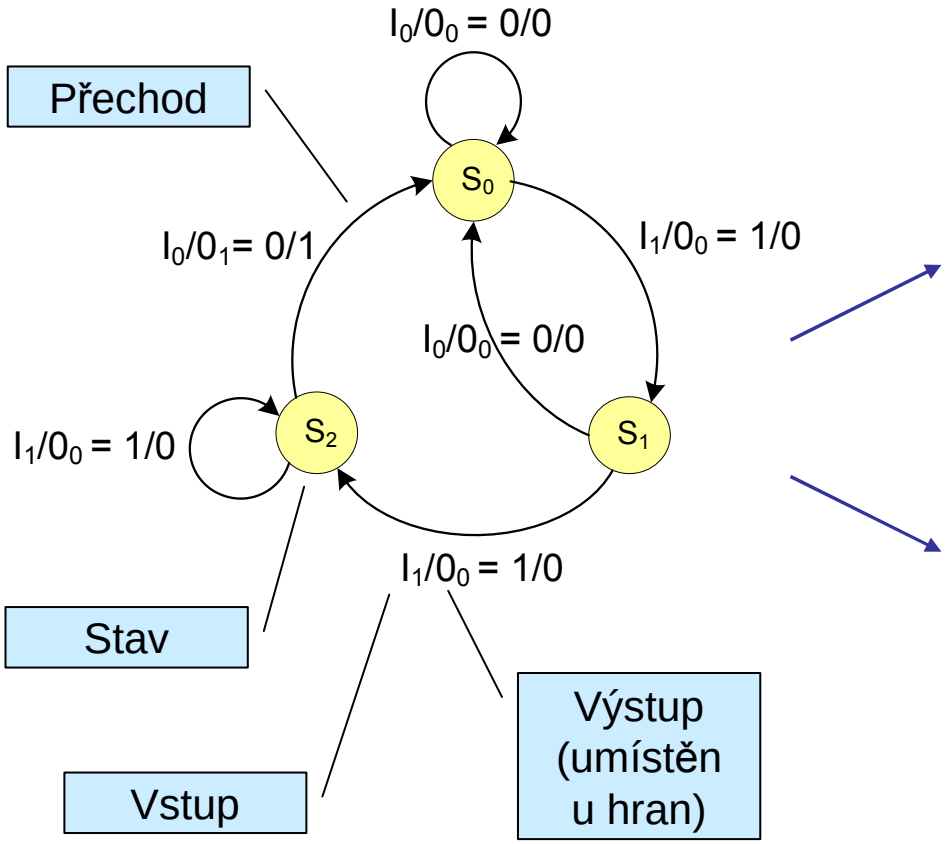
Přechod ze stavu v čase t do t+1 – „$\rightarrow$“

Okamžité vytvoření výstupu – „:“

| $S_i \alpha I_j \rightarrow S_k$ | $S_i \alpha I_j : O_k$ |
|---|---|
| $S_0 \alpha I_0 \rightarrow S_0$ | $S_0 \alpha I_0 : O_0$ |
| $S_0 \alpha I_1 \rightarrow S_1$ | $S_0 \alpha I_1 : O_0$ |
| $S_1 \alpha I_0 \rightarrow S_0$ | $S_1 \alpha I_0 : O_0$ |
| $S_1 \alpha I_1 \rightarrow S_2$ | $S_1 \alpha I_1 : O_0$ |
| $S_2 \alpha I_1 \rightarrow S_2$ | $S_2 \alpha I_1 : O_0$ |
| $S_2 \alpha I_0 \rightarrow S_0$ | $S_2 \alpha I_0 : O_1$ |

# Formy popisu FSA – typu Mealy

Stavový diagram




Tabulky přechodů a výstupů

Tabulka přechodů

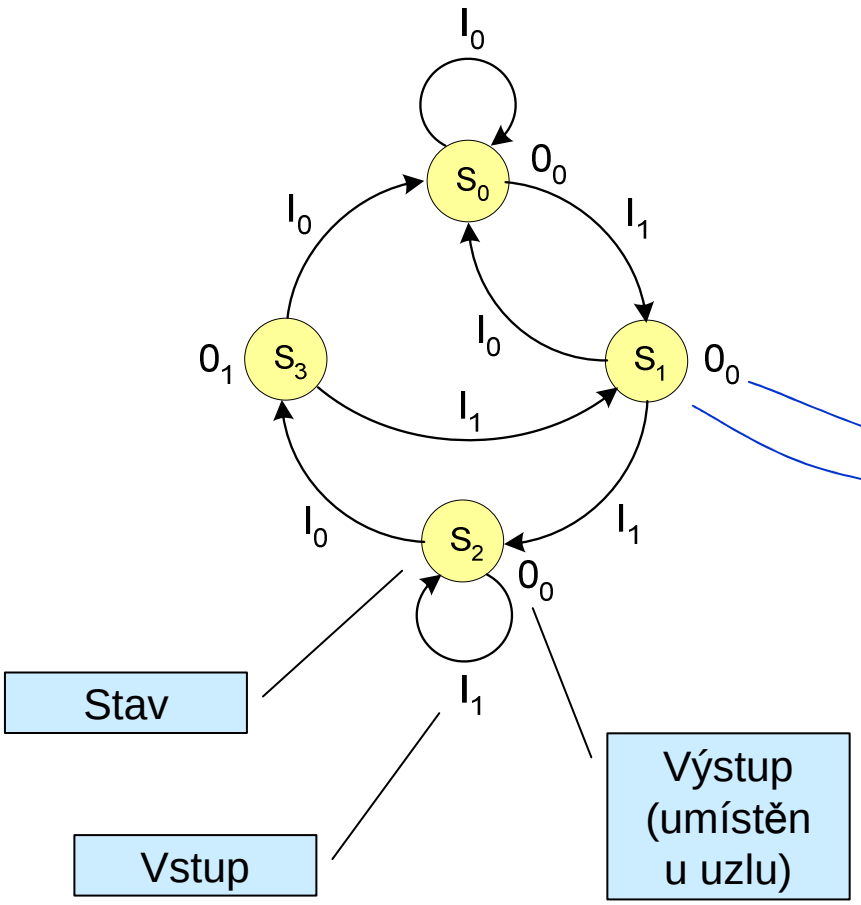
| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $S_0$ | $S_1$ |
| $S_1$ | $S_0$ | $S_2$ |
| $S_2$ | $S_0$ | $S_2$ |

Tabulka výstupů

| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $O_0$ | $O_0$ |
| $S_1$ | $O_0$ | $O_0$ |
| $S_2$ | $O_1$ | $O_0$ |

# Formy popisu FSA – typu Moore

**Stavový diagram**




**Přechodová a výstupní funkce**

Současný výstkyt stavu a vstupu – „$\alpha$“

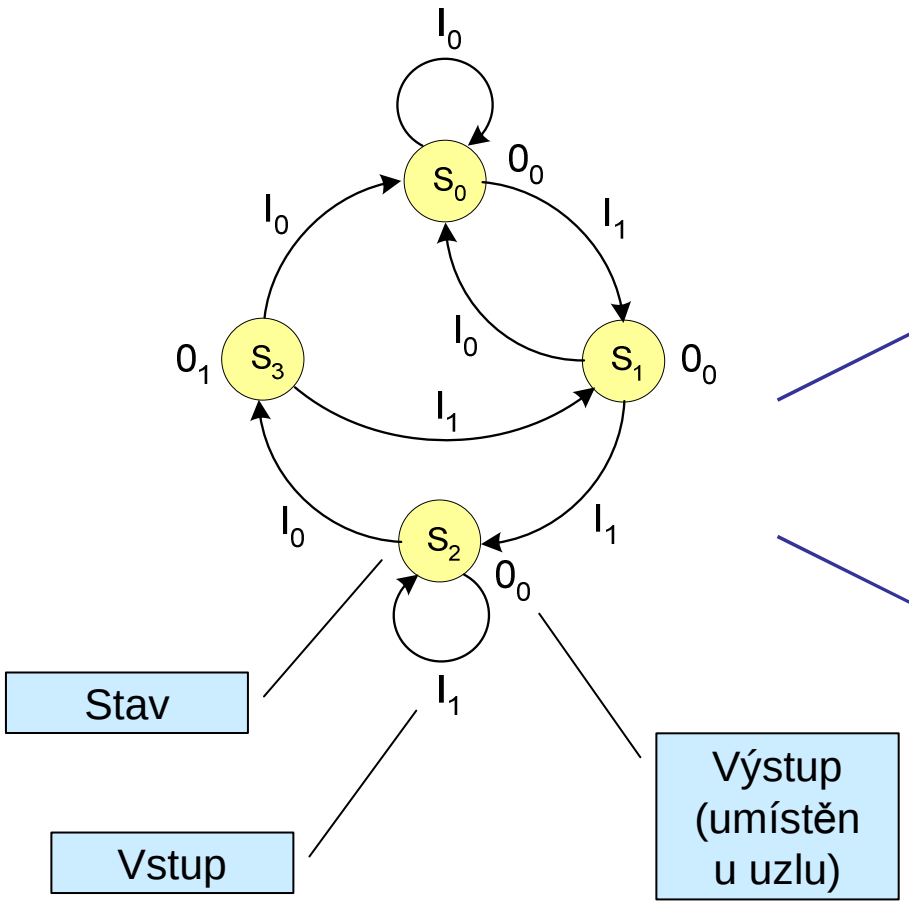
Přechod ze stavu v čase t do t+1 – „$\rightarrow$“

Okamžité vytvoření výstupu – „:“

| $S_i \, \alpha \, I_j \rightarrow S_k$ | $S_i : O_k$ |
|---|---|
| $S_0 \, \alpha \, I_0 \rightarrow S_0$ | $S_0 : O_0$ |
| $S_0 \, \alpha \, I_1 \rightarrow S_1$ | |
| $S_1 \, \alpha \, I_0 \rightarrow S_0$ | $S_1 : O_0$ |
| $S_1 \, \alpha \, I_1 \rightarrow S_2$ | |
| $S_2 \, \alpha \, I_1 \rightarrow S_2$ | $S_2 : O_0$ |
| $S_2 \, \alpha \, I_0 \rightarrow S_3$ | |
| $S_3 \, \alpha \, I_1 \rightarrow S_1$ | $S_3 : O_1$ |
| $S_3 \, \alpha \, I_0 \rightarrow S_0$ | |

# Formy popisu FSA – typu Moore

Stavový diagram




Tabulky přechodů a výstupů

Tabulka přechodů

| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $S_0$ | $S_1$ |
| $S_1$ | $S_0$ | $S_2$ |
| $S_2$ | $S_3$ | $S_2$ |
| $S_3$ | $S_0$ | $S_1$ |

Tabulka výstupů

| $S_i$ | $O_i$ |
|---|---|
| $S_0$ | $O_0$ |
| $S_1$ | $O_0$ |
| $S_2$ | $O_0$ |
| $S_3$ | $O_1$ |

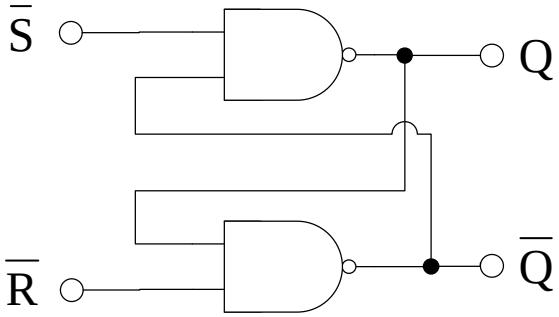
# Postup návrhu logického sekvenčního obvodu (schema)

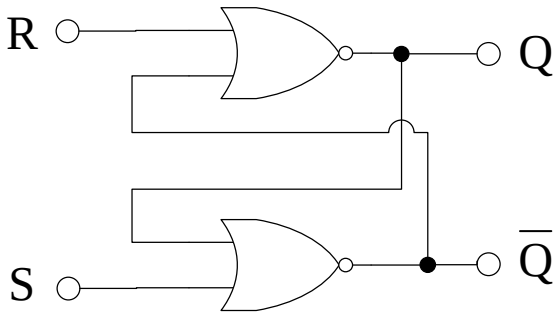
- Návrh obvodového řešení zápisem schematu
  - Formulace zadání – slovní popis
  - Stavový diagram (orientovaný graf přechodů a výstupů)
  - Tabulky přechodů a výstupů
  - Kódování vnitřních stavů a výstupů
  - Zakódované tabulky přechodů a výstupů
  - Budící funkce a funkce výstupů
  - Minimalizace budící funkce a funkce výstupů (K–mapy)
  - Návrh z hradel (z požadovaných typů) – schema
  - [Logická simulace]
  - Realizace z hradel
  - [Časová simulace po realizaci (umístění do hradlového pole)]
  - Výpočet (ověření) maximální povolené frekvence synchronizačního signálu (následující přednáška)
  - Ověření v aplikaci

# Postup návrhu logického sekvenčního obvodu (HDL)

- Návrh obvodového řešení zápisem v HDL (VHDL, Verilog)
  - Formulace zadání – slovní popis
  - Stavový diagram (orientovaný graf přechodů)
  - Zápis programu v HDL (Hardware Description Language)
  - Syntéza zapojení (překlad programu v HDL)
  - [Logická simulace]
  - Realizace z hradel (z prostředků hradlového pole)(Place & Route)
  - [Časová simulace po realizaci (umístění do hradlového pole)]
  - Výpočet (ověření) maximální povolené frekvence synchronizačního signálu (následující přednáška)
  - Ověření v aplikaci

# Paměťový člen – záchytný registr (Latch)




R–S Latch (NAND)

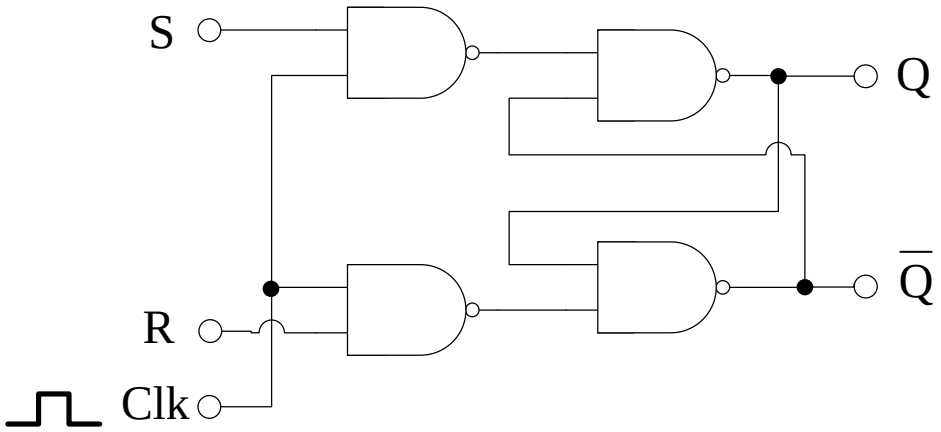
| $R^i$ | $S^i$ | $Q^{i+1}$ |
|---|---|---|
| 0 | 0 | !!! |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | $Q^i$ |

R–S Latch (NOR)

| $R^i$ | $S^i$ | $Q^{i+1}$ |
|---|---|---|
| 0 | 0 | $Q^i$ |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | !!! |

!!! – zakázaný stav

# Paměťový člen – záchytný registr (Latch)



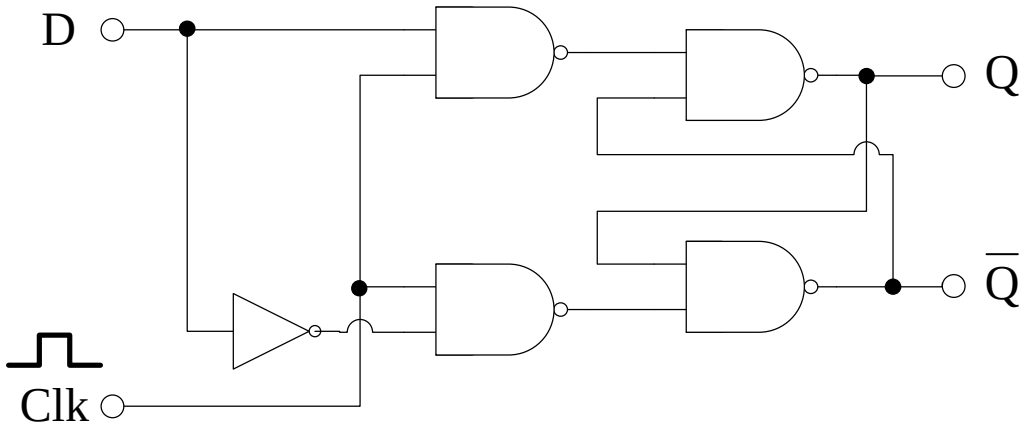

R–S Latch (Clock enable)

| $R^i$ | $S^i$ | Clk | $Q^{i+1}$ |
|---|---|---|---|
| 0 | 0 | 1 | $Q^i$ |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | !!! |
| X | X | 0 | $Q^i$ |

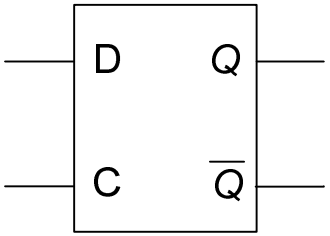
X – nezáleží

!!! – zakázaný stav

# Paměťový člen – záchytný registr (D – Latch)




Symbol

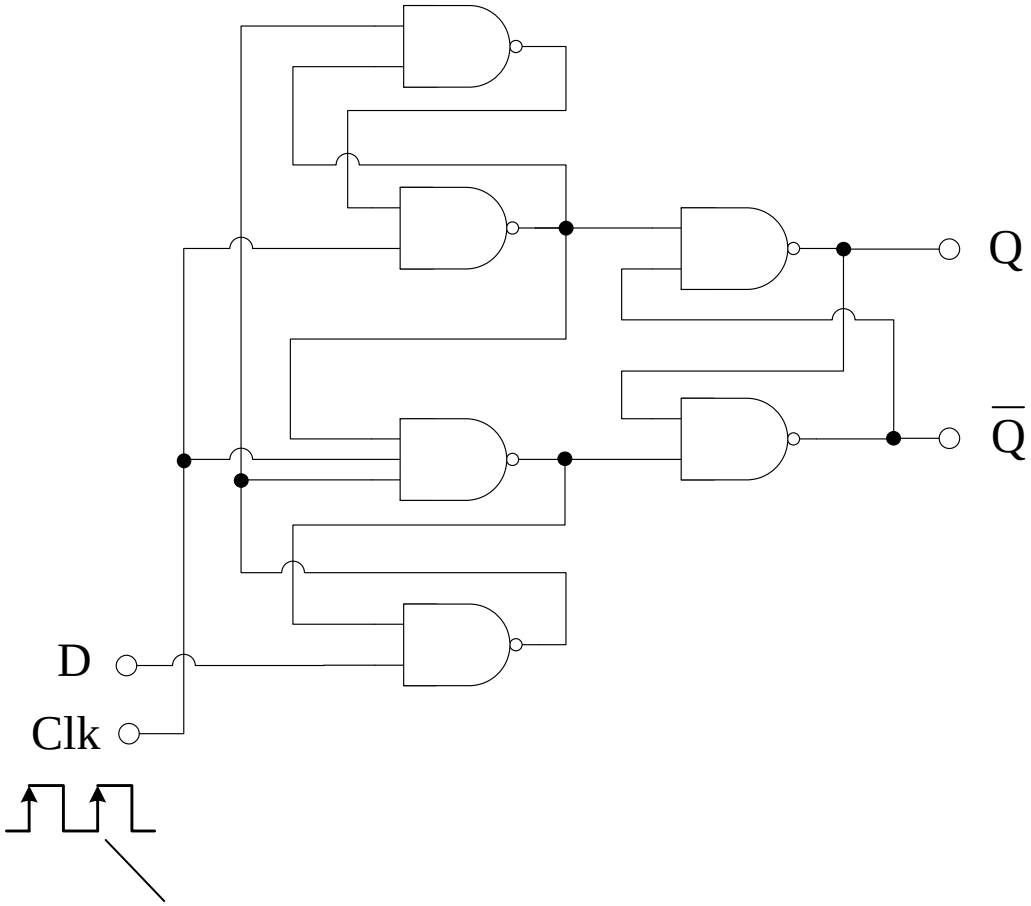



D Latch (Clock enable)

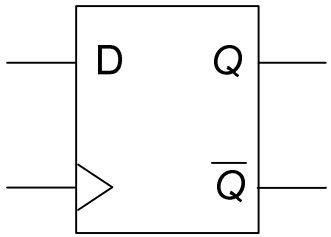
| $D^i$ | Clk | $Q^{i+1}$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| X | 0 | $Q^i$ |

X – nezáleží

# Paměťový člen – D klopný obvod (D – Flip-Flop)




Symbol

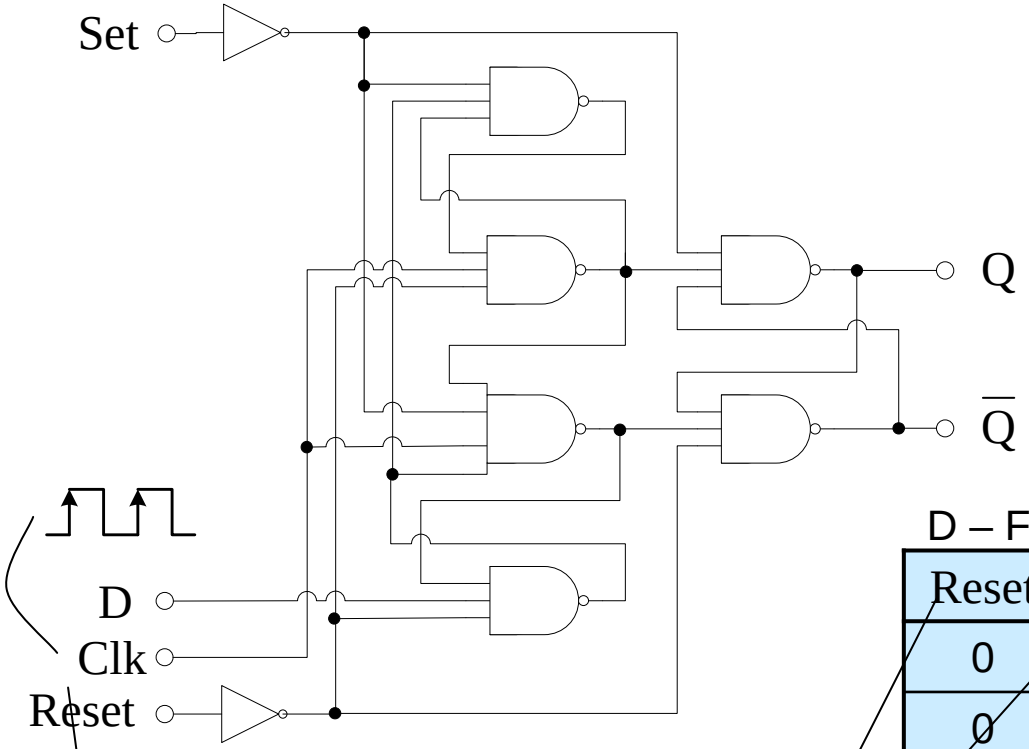



D Flip-Flop

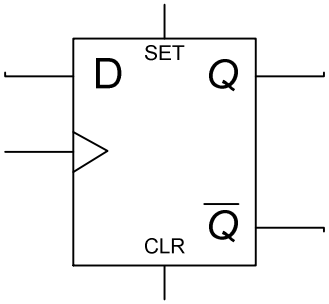
| $D^i$ | Clk | $Q^{i+1}$ |
|---|---|---|
| 0 | ↑ | 0 |
| 1 | ↑ | 1 |
| X | 0 | $Q^i$ |
| X | 1 | $Q^i$ |

↑ – zápis řízený náběžnou hranou

# Paměťový člen – D klopný obvod (D – Flip-Flop)




Symbol




↑ – zápis řízený náběžnou hranou

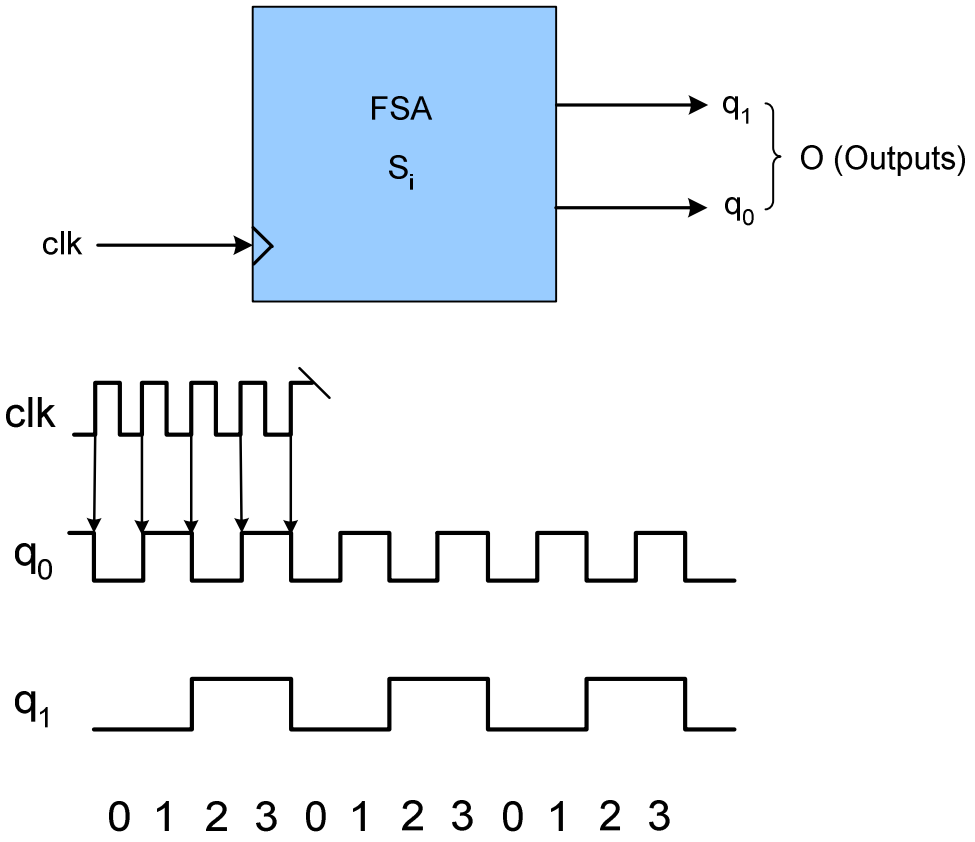
asynchronní nulování/nastavení

D – Flip-Flop

| $Reset^i$ | $Set^i$ | $D^i$ | Clk | $Q^{i+1}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | ↑ | 0 |
| 0 | 0 | 1 | ↑ | 1 |
| 0 | 0 | X | 0 | $Q^i$ |
| 0 | 0 | X | 1 | $Q^i$ |
| 0 | 1 | X | X | 1 |
| 1 | 0 | X | X | 0 |

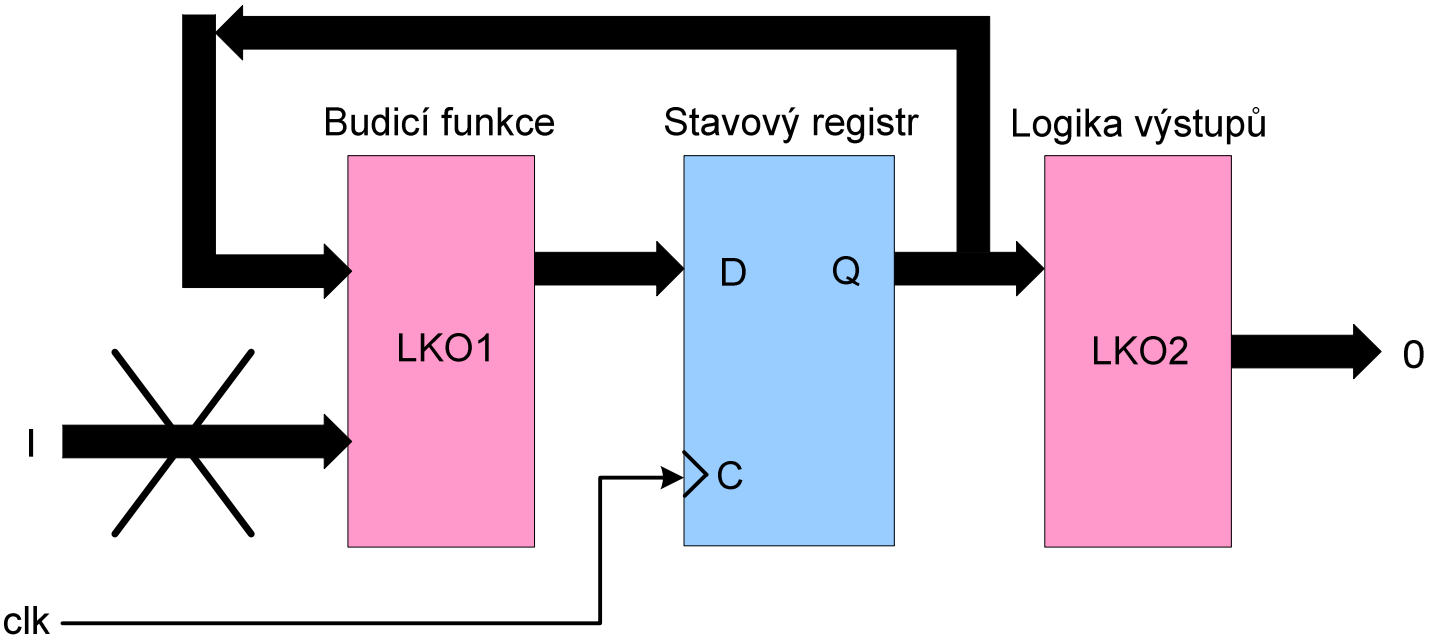
# Synchronní 2bitový binární čítač

Navrhněte synchronní konečný automat (FSA – Finite State Automaton) typu čítač. Čítač čítá v binárním kódu a je 2bitový. Automat navrhněte s asynchronním nulováním.



# Synchronní 2bitový binární čítač

Co máme navrhnout?



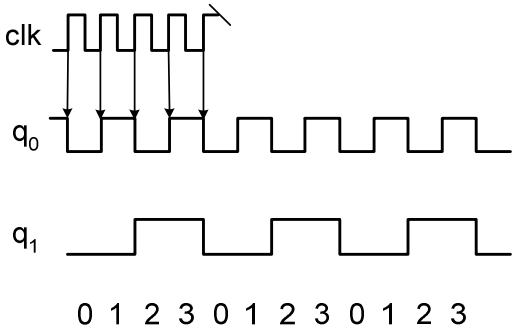
# Synchronní 2bitový binární čítač

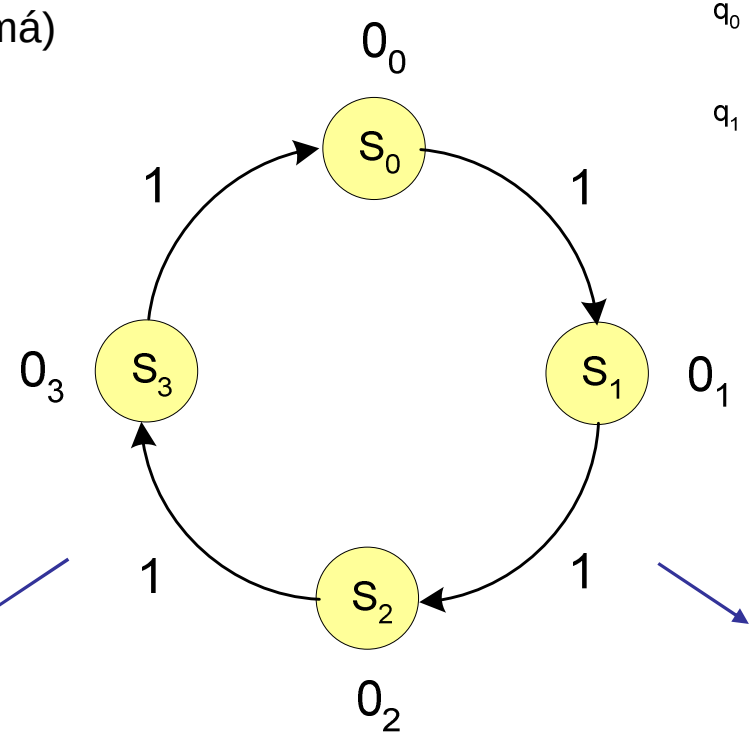
**Stavový diagram**

I – Vstupy (Inputs) (nemá)

O – Výstupy (Outputs)

$S_i$ – i-tý stav




Tabulka přechodů

| $S_i$ | $S_{i+1}$ |
|---|---|
| $S_0$ | $S_1$ |
| $S_1$ | $S_2$ |
| $S_2$ | $S_3$ |
| $S_3$ | $S_0$ |

Tabulka výstupů

| $S_i$ | $O_i$ |
|---|---|
| $S_0$ | $O_0$ |
| $S_1$ | $O_1$ |
| $S_2$ | $O_2$ |
| $S_3$ | $O_3$ |

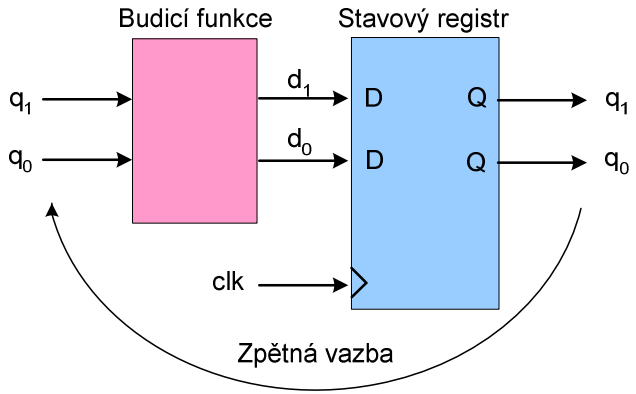
# Synchronní 2bitový binární čítač

Tabulka přechodů

| $S_i$ | $S_{i+1}$ |
|---|---|
| $S_0$ | $S_1$ |
| $S_1$ | $S_2$ |
| $S_2$ | $S_3$ |
| $S_3$ | $S_0$ |

Kódování stavů

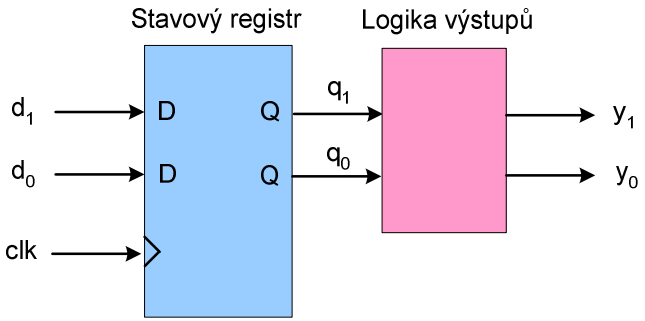
| $S_i$ | $q_1$ | $q_0$ | $d_1$ | $d_0$ | $S_{i+1}$ |
|---|---|---|---|---|---|
| $S_0$ | 0 | 0 | 0 | 1 | $S_1$ |
| $S_1$ | 0 | 1 | 1 | 0 | $S_2$ |
| $S_2$ | 1 | 0 | 1 | 1 | $S_3$ |
| $S_3$ | 1 | 1 | 0 | 0 | $S_0$ |



# Synchronní 2bitový binární čítač

Tabulka výstupů

| $S_i$ | $O_I$ |
|---|---|
| $S_0$ | $O_0$ |
| $S_1$ | $O_1$ |
| $S_2$ | $O_2$ |
| $S_3$ | $O_3$ |




$S_i$ $O_i$

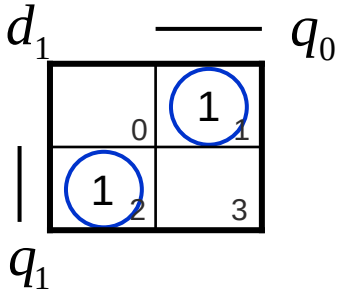
Kódování výstupů

| $S_i$ | $q_1$ | $q_0$ | $y_1$ | $y_0$ | $O_i$ |
|---|---|---|---|---|---|
| $S_0$ | 0 | 0 | 0 | 0 | $O_0$ |
| $S_1$ | 0 | 1 | 0 | 1 | $O_1$ |
| $S_2$ | 1 | 0 | 1 | 0 | $O_2$ |
| $S_3$ | 1 | 1 | 1 | 1 | $O_3$ |

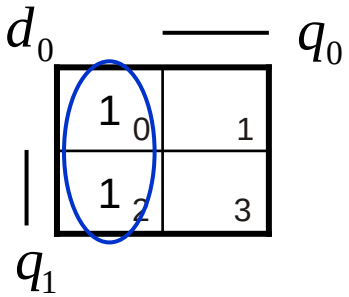
$S_i = O_i$

# Synchronní 2bitový binární čítač

Minimalizace




$$d_1 = q_1\overline{q_0} + \overline{q_1}\,q_0 = \text{XOR}$$



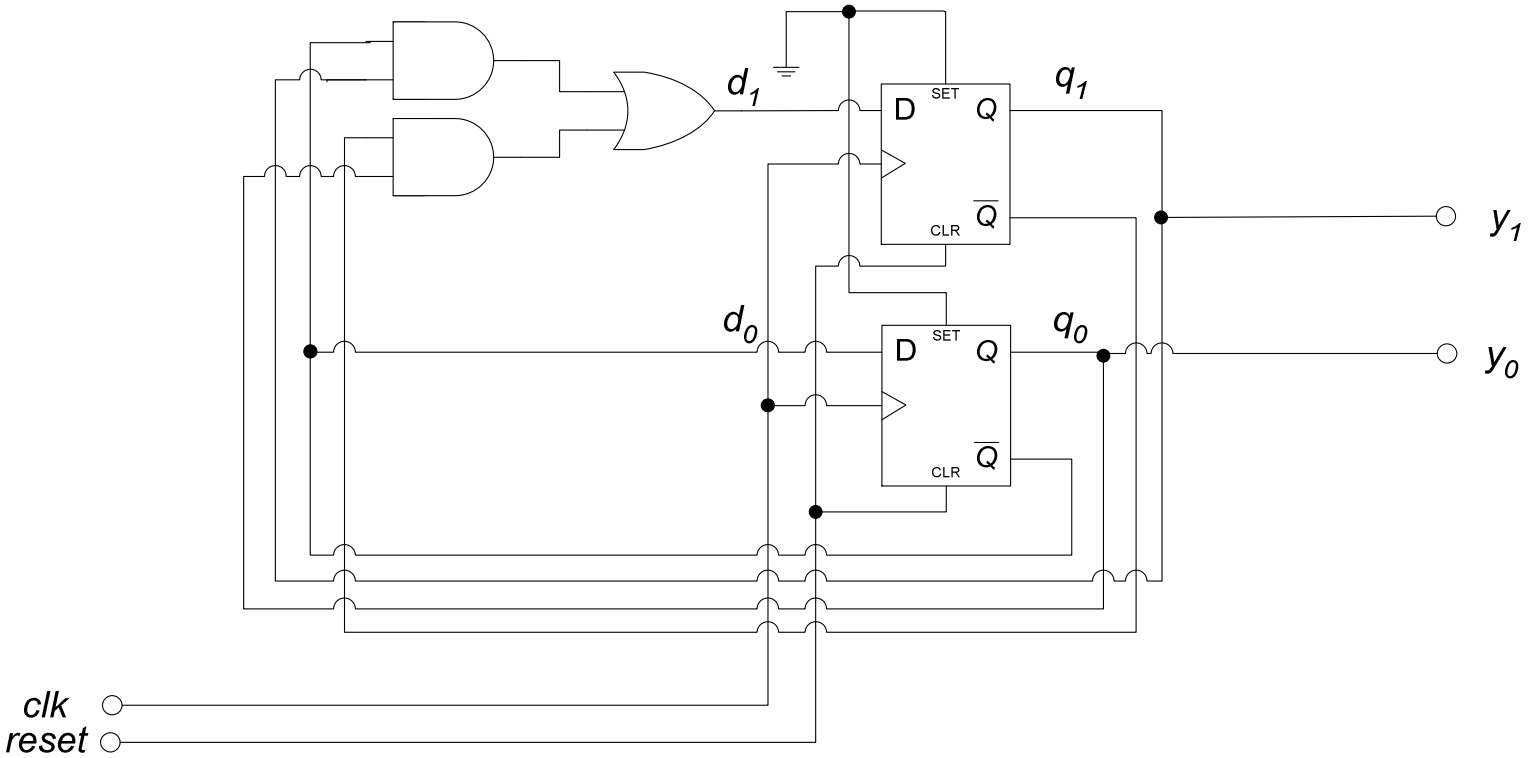

$$d_0 = \overline{q_0}$$

$$y_1 = q_1$$

$$y_0 = q_0$$

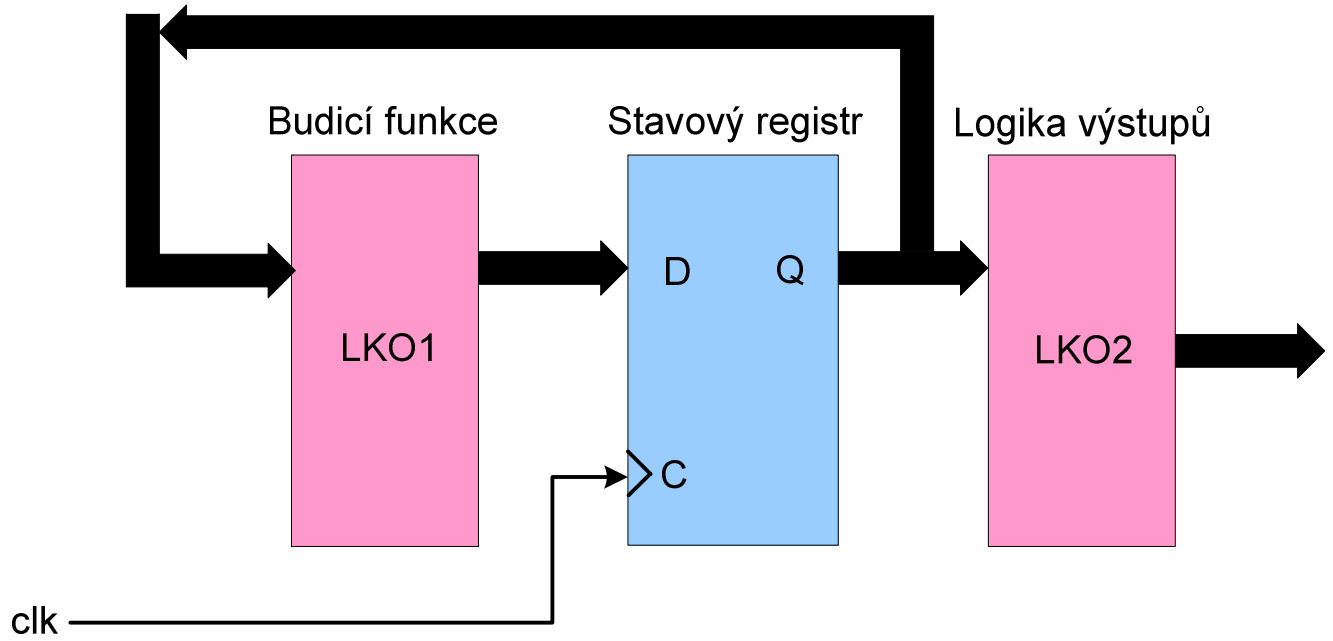
# Synchronní 2bitový binární čítač

Realizace



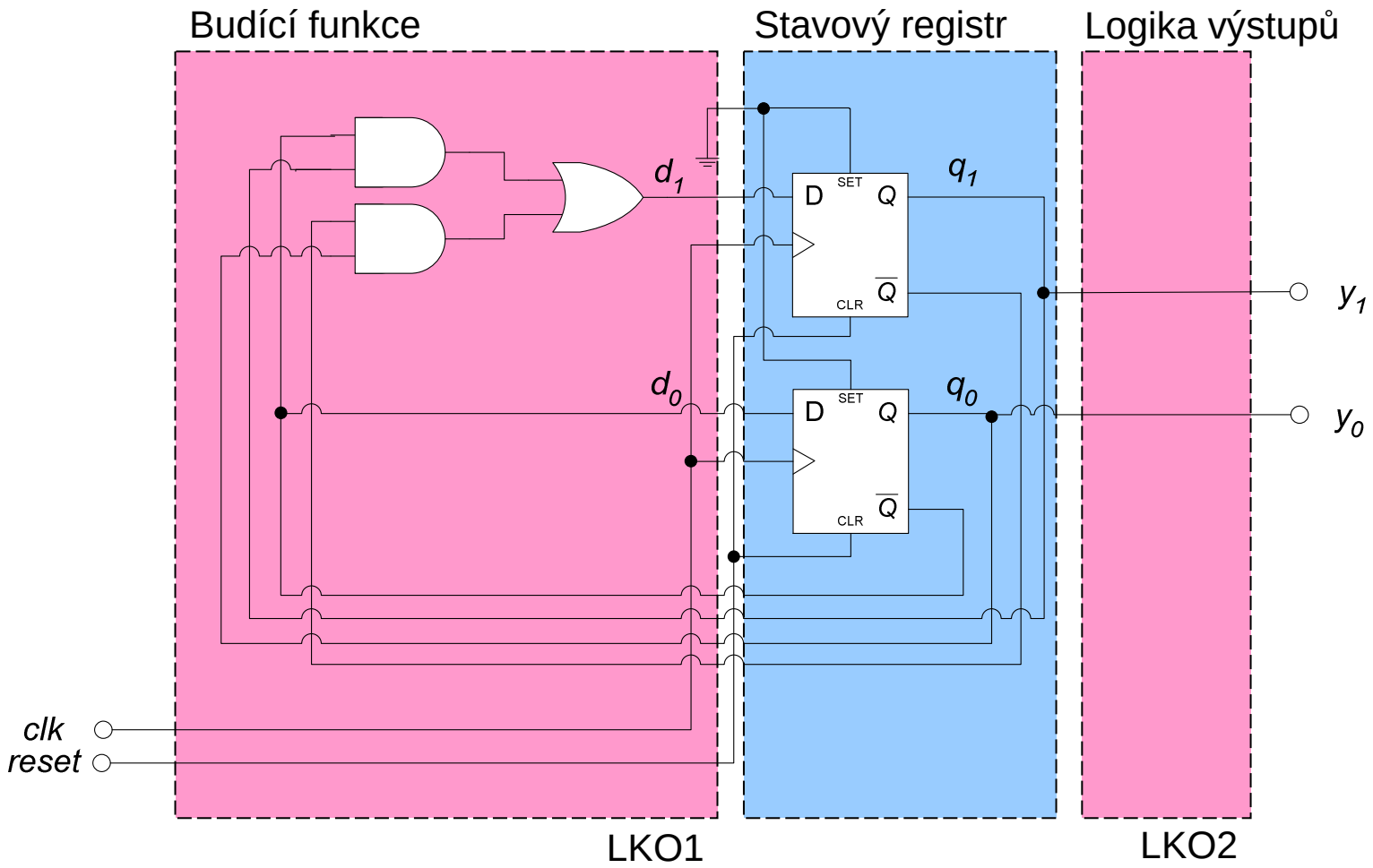
# Synchronní 2bitový binární čítač

Co jsme navrhli?



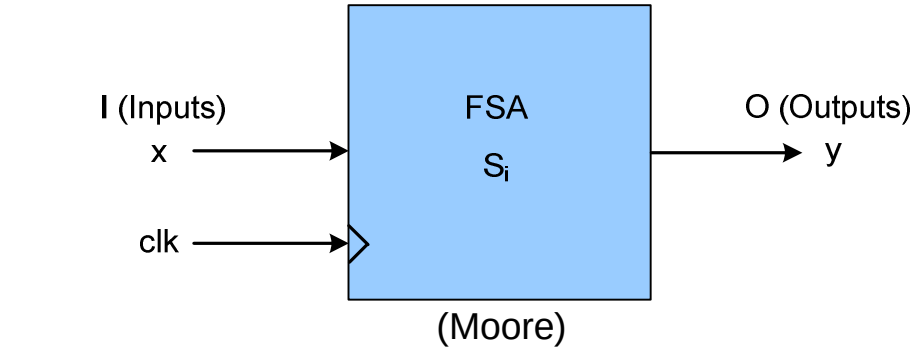
# Synchronní 2bitový binární čítač

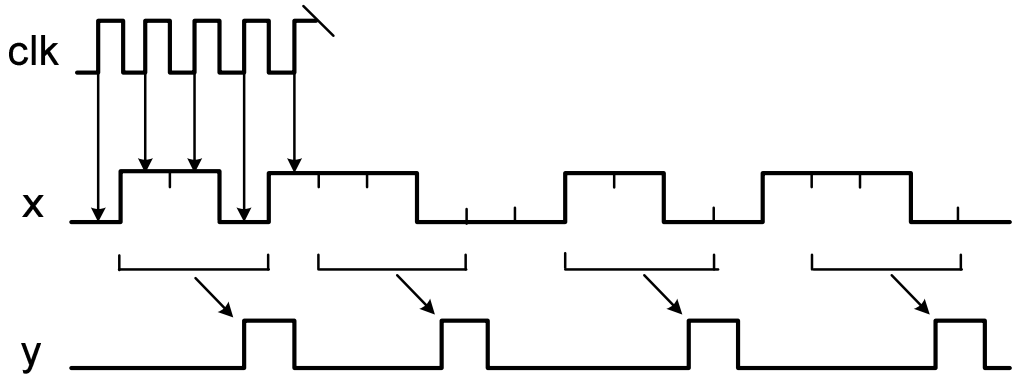
Co jsme navrhli?



# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

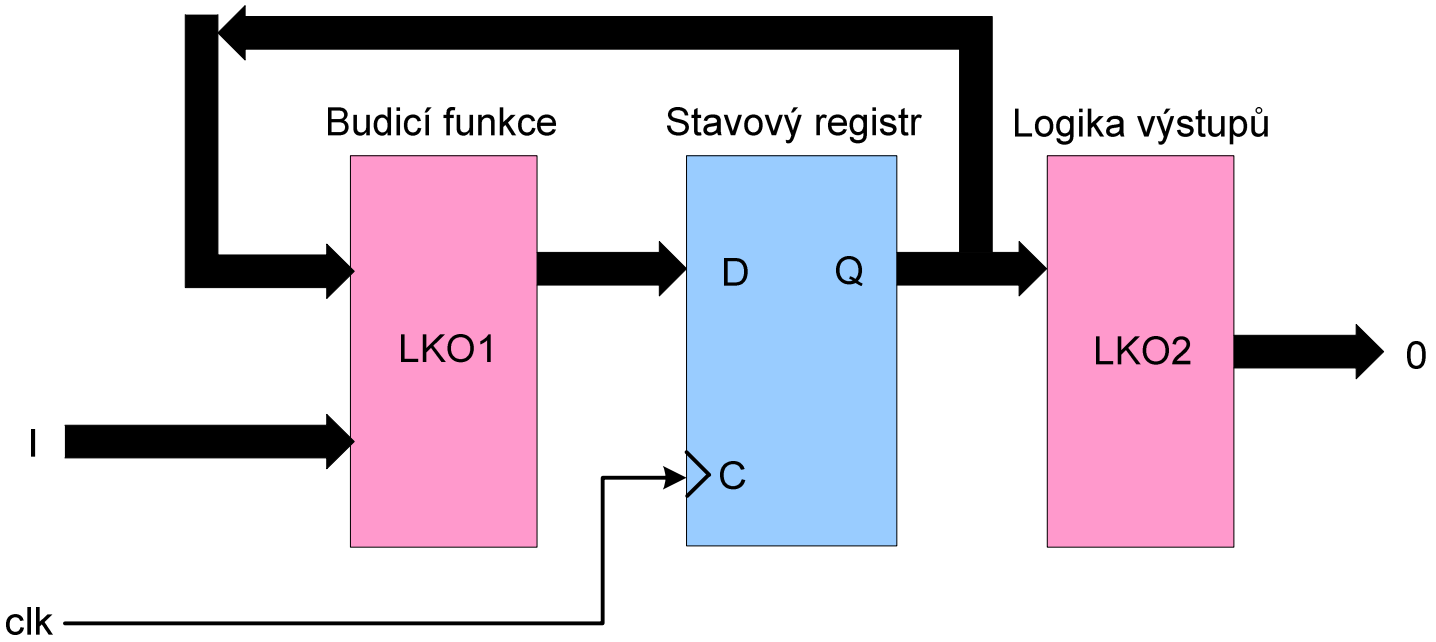
Navrhněte synchronní konečný automat (FSA – Finite State Automaton), který v proudu vstupních bitů detekuje posloupnost ‘110’. Při detekci každé takové posloupnosti automat vyšle na výstupu impuls. Automat navrhněte s asynchronním nulováním.



# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

Co máme navrhnout?



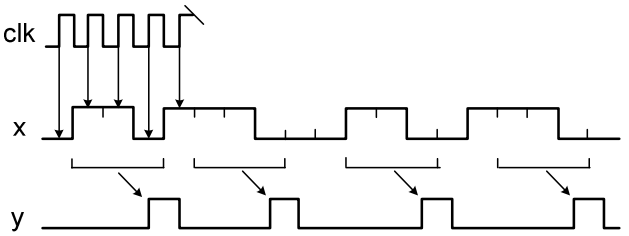
# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

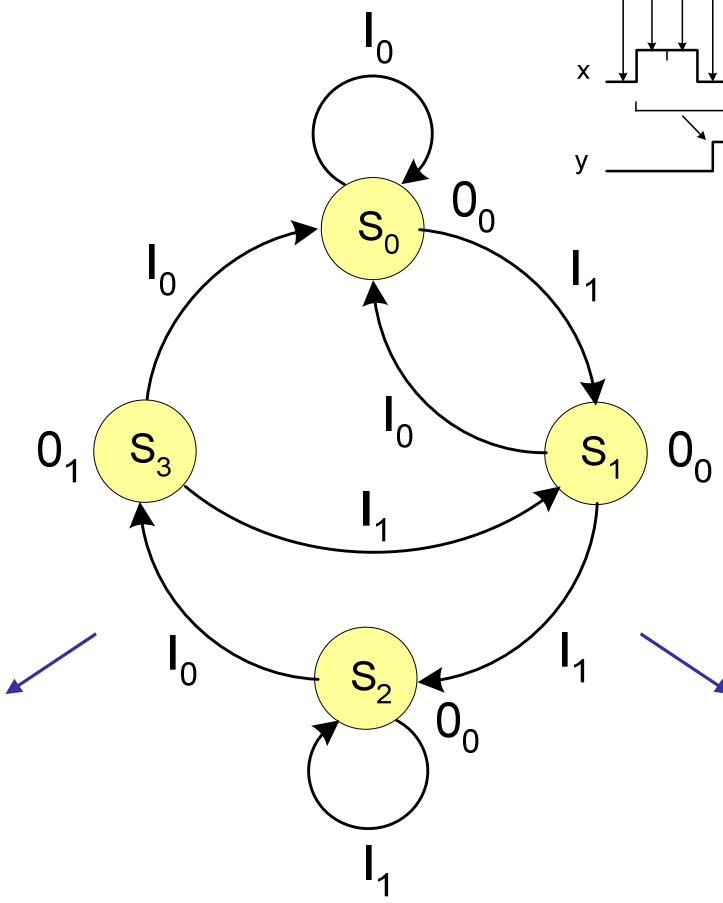
Stavový diagram

I – Vstupy (Inputs)

O – Výstupy (Outputs)

$S_i$ – i-tý stav




Tabulka přechodů

| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $S_0$ | $S_1$ |
| $S_1$ | $S_0$ | $S_2$ |
| $S_2$ | $S_3$ | $S_2$ |
| $S_3$ | $S_0$ | $S_1$ |

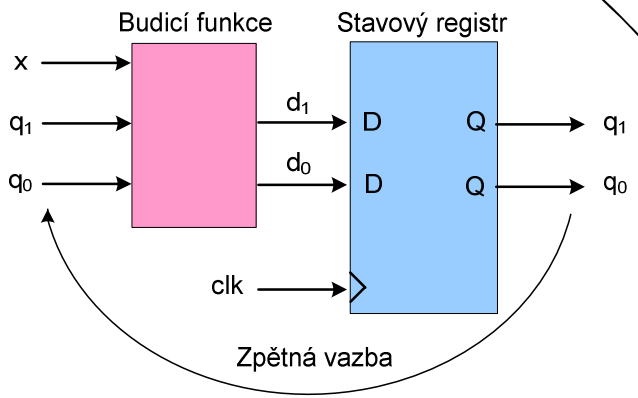
Tabulka výstupů

| $S_i$ | $O_i$ |
|---|---|
| $S_0$ | $O_0$ |
| $S_1$ | $O_0$ |
| $S_2$ | $O_0$ |
| $S_3$ | $O_1$ |

# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

Tabulka přechodů

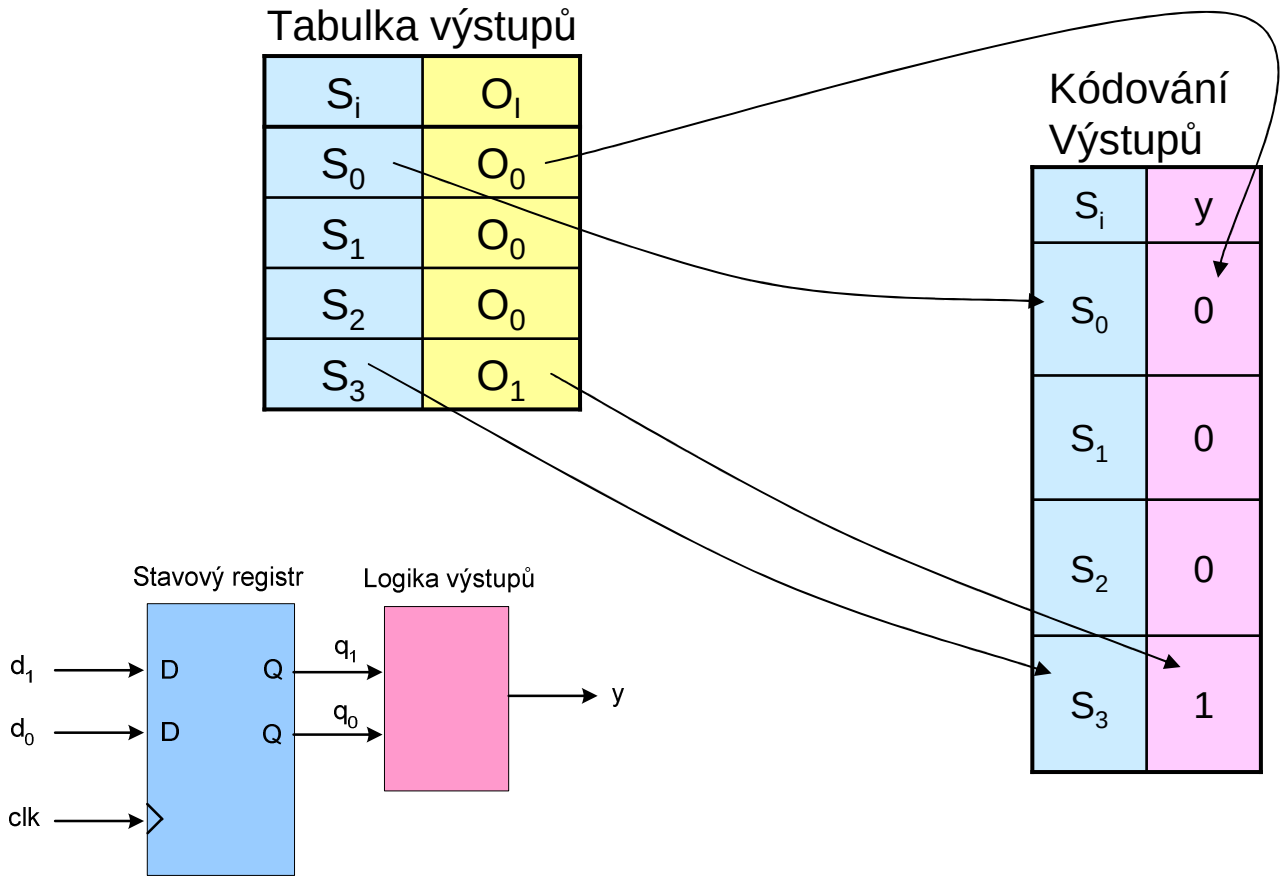
| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $S_0$ | $S_1$ |
| $S_1$ | $S_0$ | $S_2$ |
| $S_2$ | $S_3$ | $S_2$ |
| $S_3$ | $S_0$ | $S_1$ |




Kódování stavů

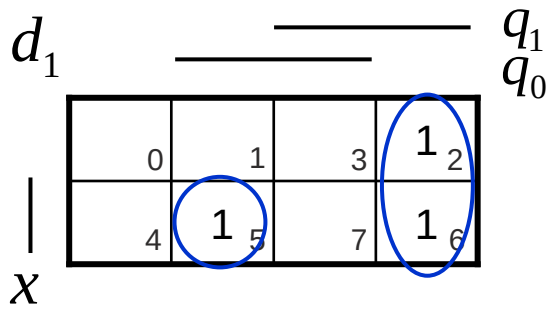
| $S_i$ | $q_1$ | $q_0$ | x | $d_1$ | $d_0$ | $S_{i+1}$ |
|---|---|---|---|---|---|---|
| $S_0$ | 0 | 0 | 0 | 0 | 0 | $S_0$ |
| | 0 | 0 | 1 | 0 | 1 | $S_1$ |
| $S_1$ | 0 | 1 | 0 | 0 | 0 | $S_0$ |
| | 0 | 1 | 1 | 1 | 0 | $S_2$ |
| $S_2$ | 1 | 0 | 0 | 1 | 1 | $S_3$ |
| | 1 | 0 | 1 | 1 | 0 | $S_2$ |
| $S_3$ | 1 | 1 | 0 | 0 | 0 | $S_0$ |
| | 1 | 1 | 1 | 0 | 1 | $S_1$ |

# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

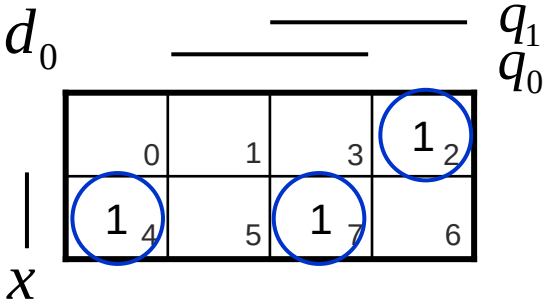


# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

Minimalizace




$$d_1 = q_1\,\overline{q_0} + \overline{q_1}\,q_0\,x$$

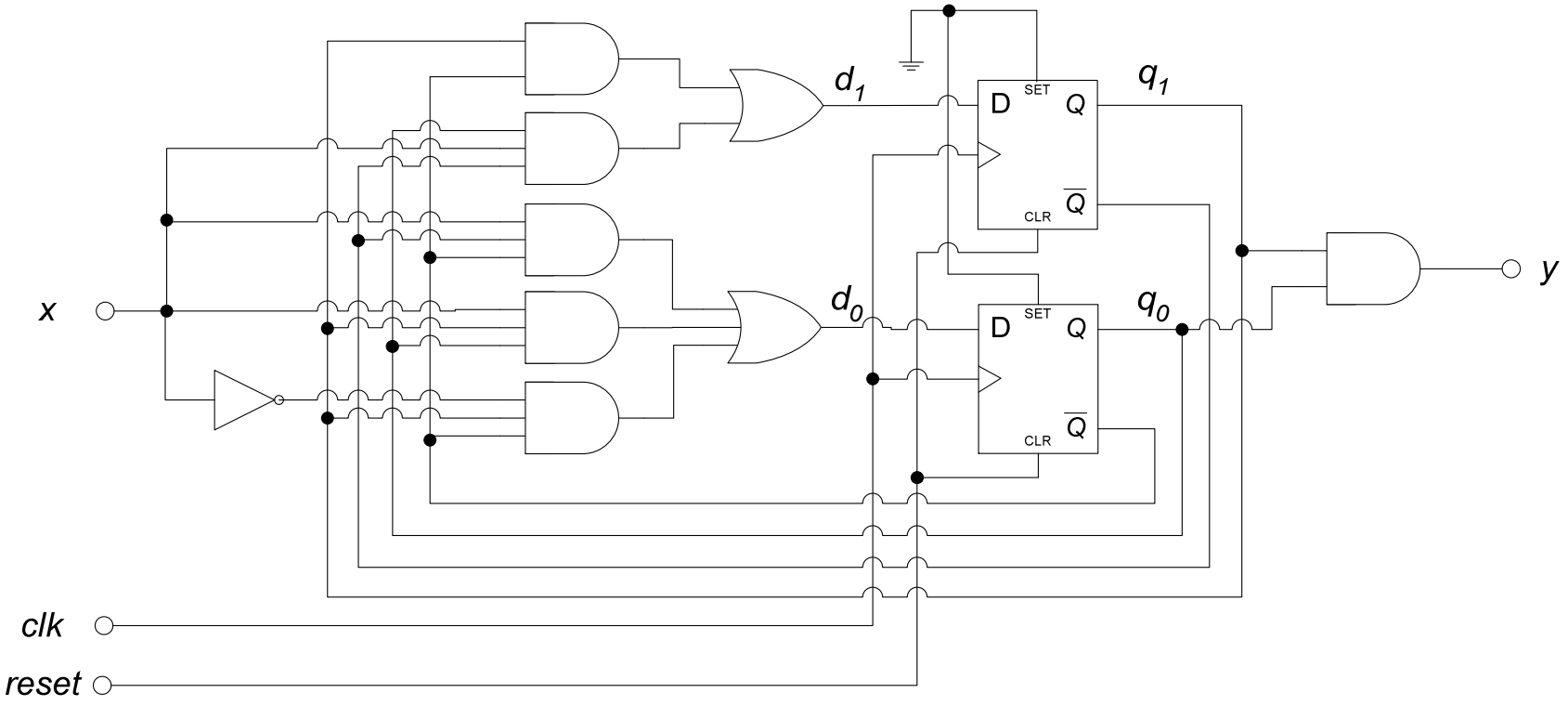



$$d_0 = \overline{q_1}\,\overline{q_0}\,x + q_1\,q_0\,x + q_1\,\overline{q_0}\,\overline{x} =$$
$$= x(\overline{q_1}\,\overline{q_0} + q_1\,q_0) + q_1\,\overline{q_0}\,\overline{x}$$

$$y = q_1\,q_0$$

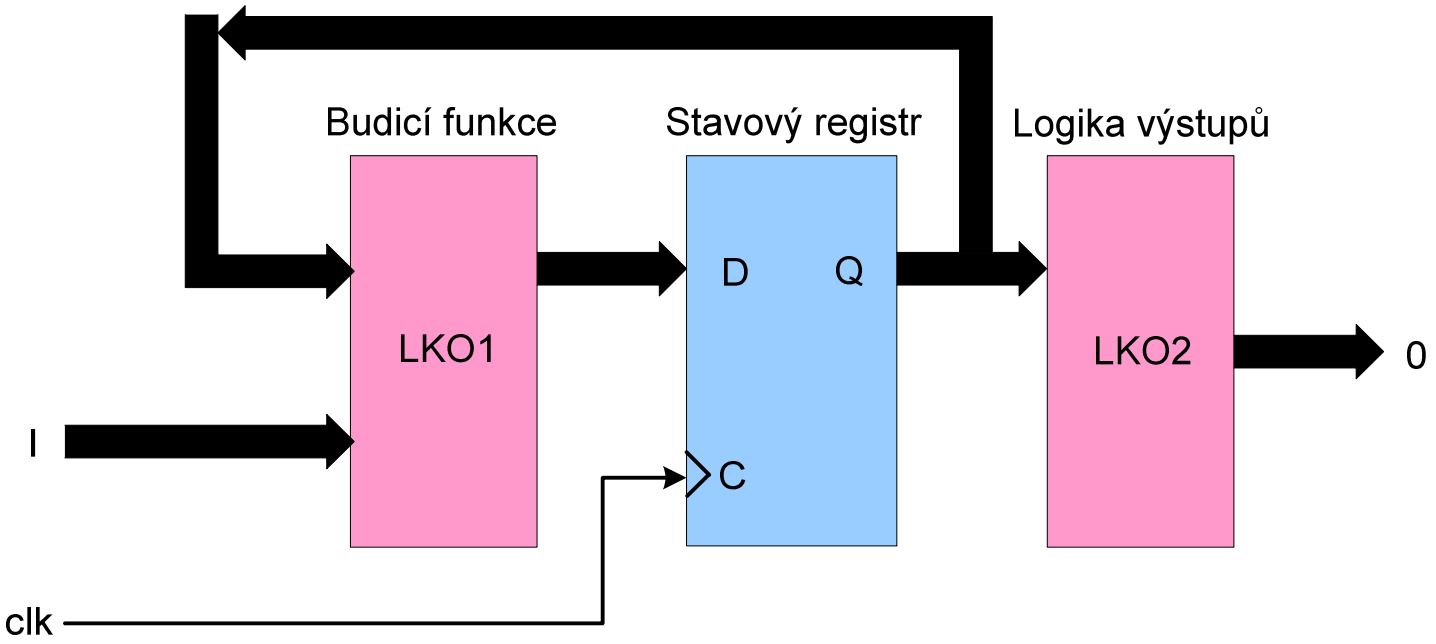
# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

Realizace



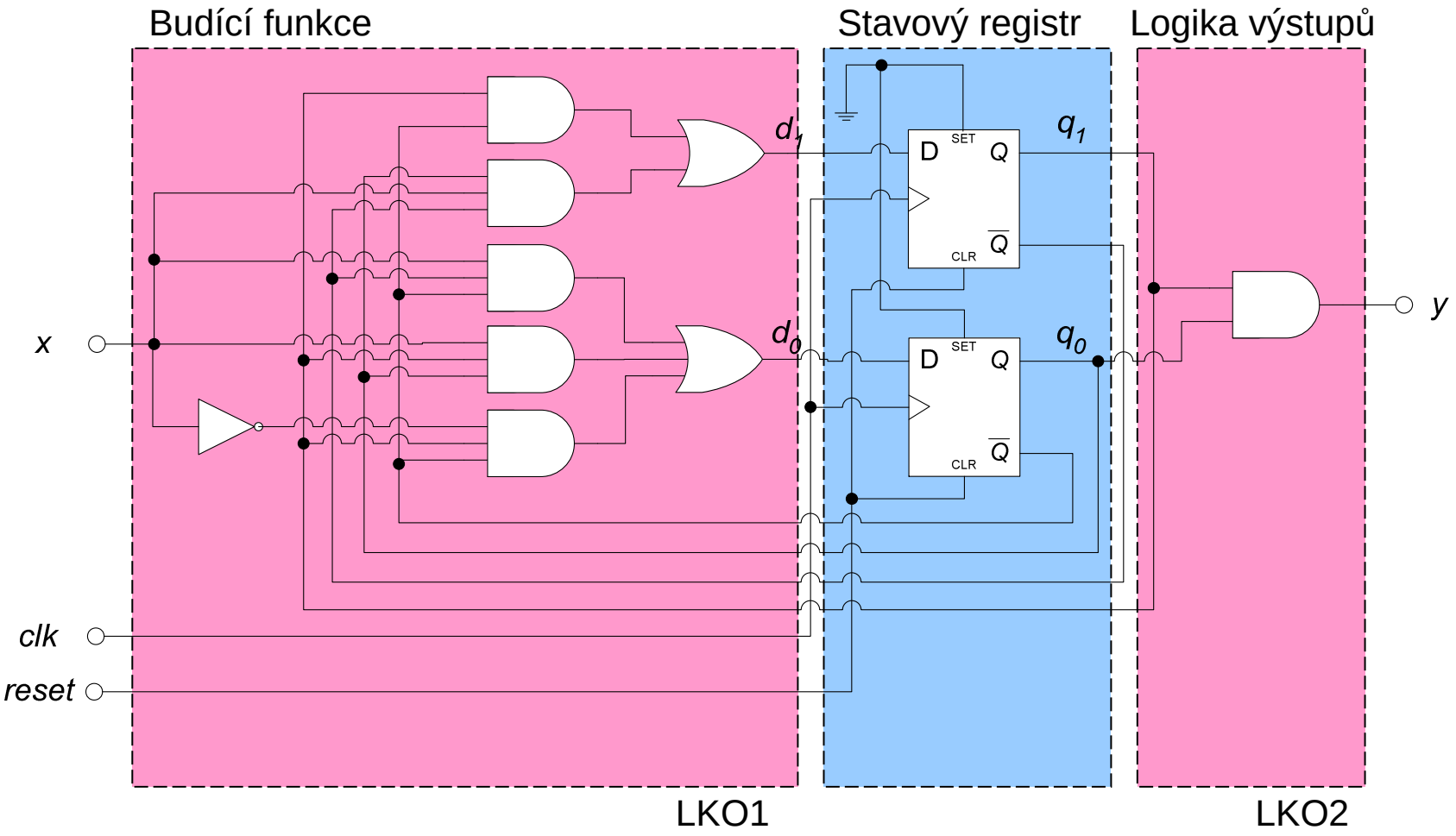
# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

Co jsme navrhli?



# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

Co jsme navrhli?



# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

“C”

```
int cBitStream3Decoder(int x_in, int reset){
   // Moore type FSA,
   // Inputs: x_in, reset, Outputs: y_out
   enum {s0,s1,s2,s3};
   static int stateReg=s0, nextState=s0, y_out;
   if(reset == TRUE){
      stateReg = s0; nextState = s0; x_in = 0;
    }
   y_out = 0;
   stateReg = nextState;
   switch(stateReg){
      case s0:
         if(x_in == 0);
         if(x_in == 1)
            nextState = s1;
          break;
      case s1:
         if(x_in == 0)
            nextState = s0;
         if(x_in == 1)
            nextState = s2;
         break;
      case s2:
         if(x_in == 0)
            nextState = s3;
         if(x_in == 1);
         break;
      case s3:
         y_out = 1;
         if(x_in == 0)
            nextState = s0;
         if(x_in == 1)
            nextState = s1;
         break;
      default: // Error section
         y_out = 0;
         nextState = s0;
   }// switch() END
   return(y_out);
}// cBitStream3Decoder() END
```

# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

Java

```
class JBitStream3Decoder {
   final int s0 = 0, s1 = 1, s2 = 2, s3 = 3;
   int stateReg = s0, nextState = s0;
   int yOut = 0;

   public JBitStream3Decoder() {// Constructor
      // empty
   }

   void setFsaReset (boolean reset){
      stateReg = s0;
      nextState = s0;
      yOut = 0;
   }

   int jBitStream3Decoder(int xIn) {
       // Moore type FSA
       // Inputs: xIn, reset,  Outputs: y_out
      yOut = 0;
      stateReg = nextState;
      switch (stateReg) {
         case s0:
            if (xIn == 0);
            if (xIn == 1)
               nextState = s1;
            break;
         case s1:
            if (xIn == 0)
               nextState = s0;
            if (xIn == 1)
               nextState = s2;
            break;
         case s2:
            if (xIn == 0)
               nextState = s3;
            if (xIn == 1);
            break;
         case s3:
            yOut = 1;
            if (xIn == 0)
               nextState = s0;
            if (xIn == 1)
               nextState = s1;
            break;
         default: // Error section
            yOut = 0;
            nextState = s0;
      }// switch() END
      return (yOut);
   }// jBitStream3Decoder() END

}// JBitStream3Decoder class END
```

# Detektor posloupnosti bitů ‘110’ (FSA typu Moore)

VHDL

```
entity vBitStream3Decoder is
    Port ( clk : in  STD_LOGIC;
        x_in : in  STD_LOGIC;
        y_out : out  STD_LOGIC;
        reset : in  STD_LOGIC;
        q : out std_logic_vector(1 downto 0)
        );
end vBitStream3Decoder;

architecture Behavioral of vBitStream3Decoder is
    type states is (s1,s2,s3,s4);
    signal stateReg, nextState: states:= s1;
begin  -- FSA - Finite State Machine
    process(clk, reset)
        begin
            if reset = '1' then
                stateReg <= s1;
            elsif
                clk'event and clk = '1' then
                stateReg <= nextState;
            end if;
        end process;
```

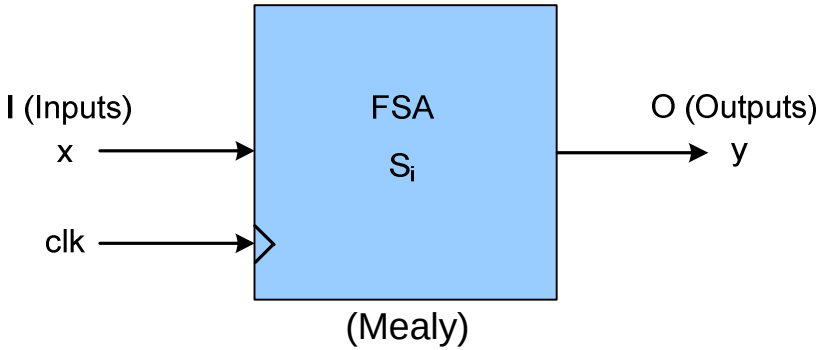
```
process(stateReg, x_in) -- State diagram
        definition
         begin
            nextState <= stateReg;
               case stateReg is
                  when s1 =>
                     if x_in = '1' then
                        nextState <= s2;
                     end if;
                  when s2 =>
                     if x_in = '0' then
                        nextState <= s1;
                     elsif x_in = '1' then
                        nextState <= s3;
                     end if;
                  when s3 =>
                      if x_in = '1' then
                         nextState <= s3;
                      elsif x_in = '0' then
                         nextState <= s4;
                      end if;
                   when s4 =>
                      if x_in = '1' then
                         nextState <= s2;
                      elsif x_in = '0' then
                         nextState <= s1;
                      end if;
```

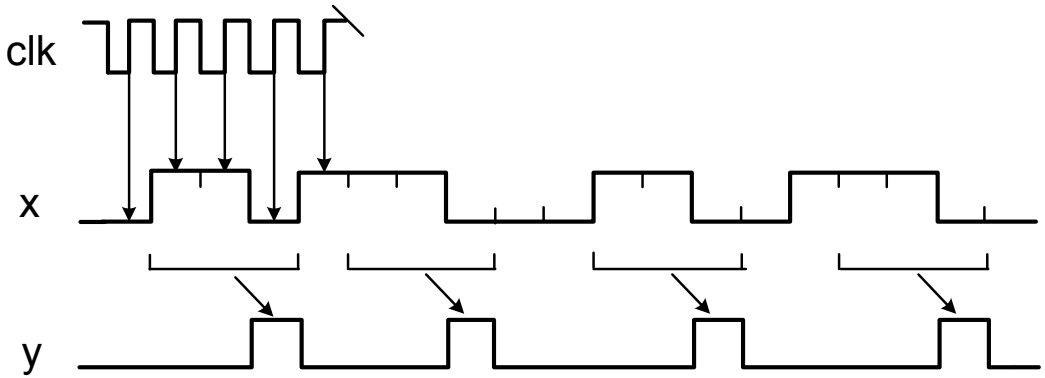
```
                  when others =>
                        nextState <= stateReg;
               end case;
            end process;

            process(stateReg) -- Output
          function
               begin
                  case stateReg is
                     when s1 =>
                        y_out <= '0';
                     when s2 =>
                        y_out <= '0';
                     when s3 =>
                        y_out <= '0';
                     when s4 =>
                        y_out <= '1';
                     when others => null;
                  end case;
               end process;
end Behavioral;
```

# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

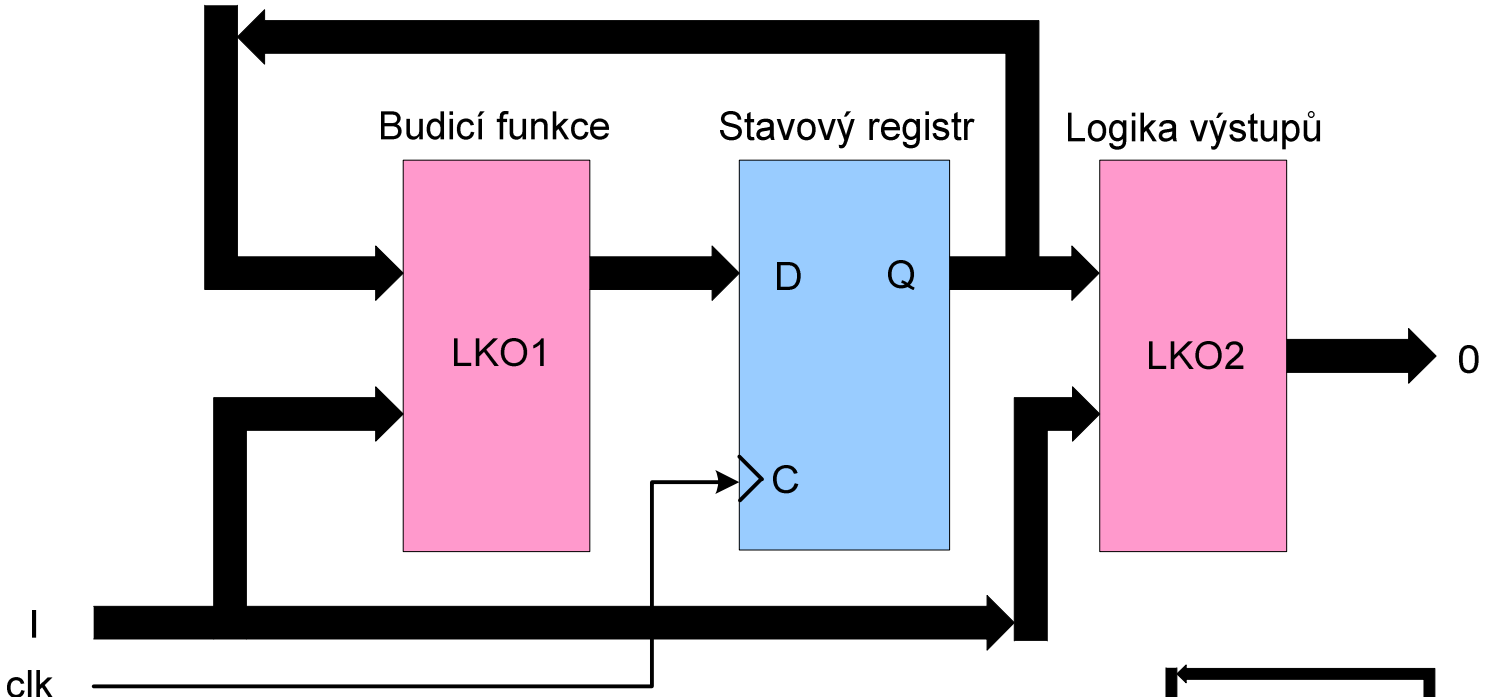
Navrhněte synchronní konečný automat (FSA – Finite State Automaton), který v proudu vstupních bitů detekuje posloupnost ‘110’. Při detekci každé takové posloupnosti automat vyšle na výstupu impuls. Automat navrhněte s asynchronním nulováním.



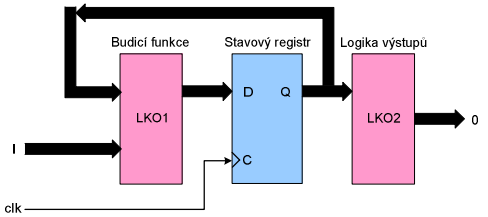
# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

Co máme navrhnout?




FSA Mealy

Srovnej vs FSA Moore



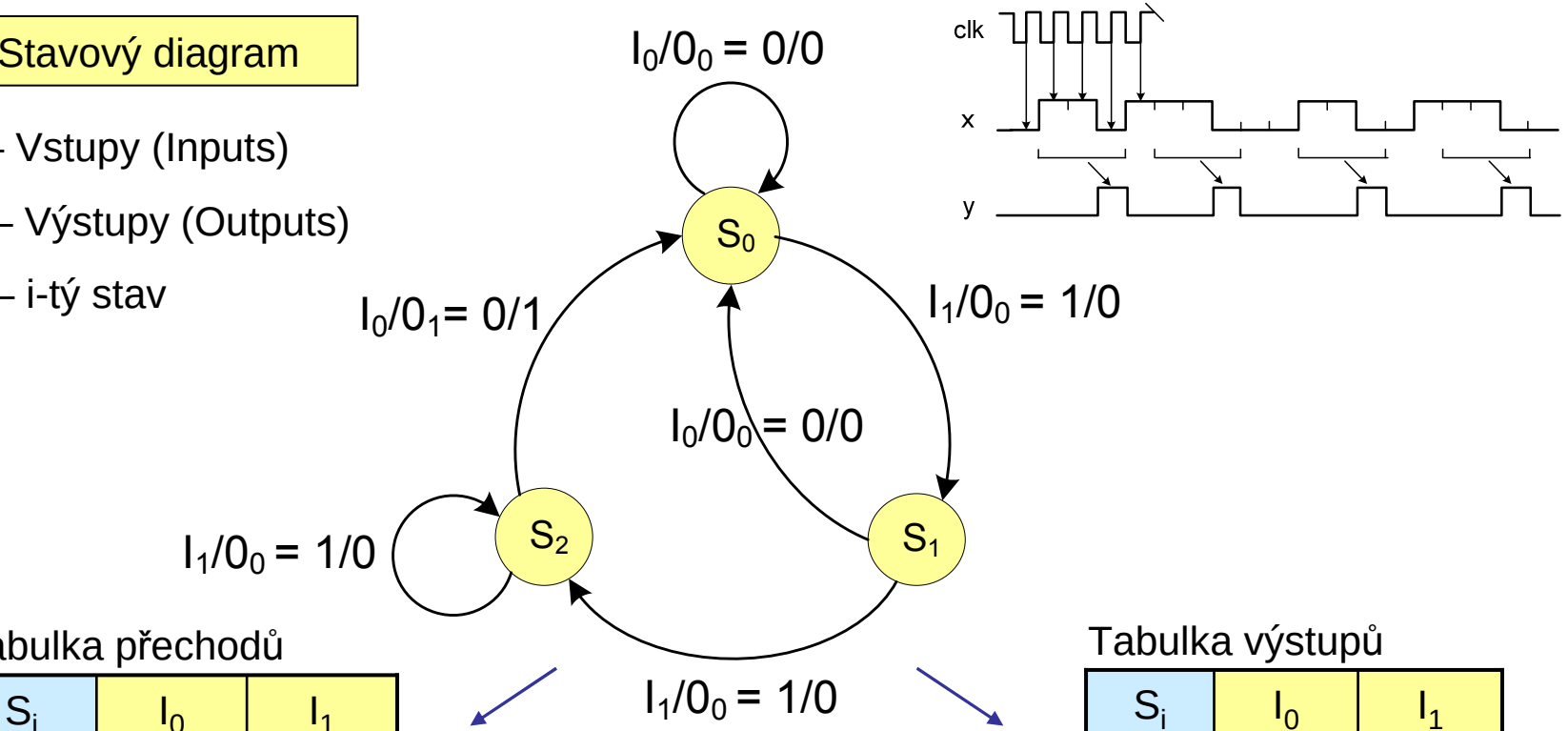
# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

Stavový diagram

I – Vstupy (Inputs)

O – Výstupy (Outputs)

$S_i$ – i-tý stav







Tabulka přechodů

| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $S_0$ | $S_1$ |
| $S_1$ | $S_0$ | $S_2$ |
| $S_2$ | $S_0$ | $S_2$ |

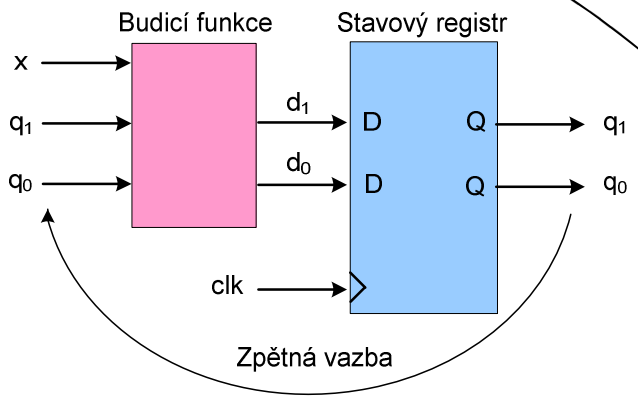
Tabulka výstupů

| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $O_0$ | $O_0$ |
| $S_1$ | $O_0$ | $O_0$ |
| $S_2$ | $O_1$ | $O_0$ |

# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

Tabulka přechodů

| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $S_0$ | $S_1$ |
| $S_1$ | $S_0$ | $S_2$ |
| $S_2$ | $S_0$ | $S_2$ |



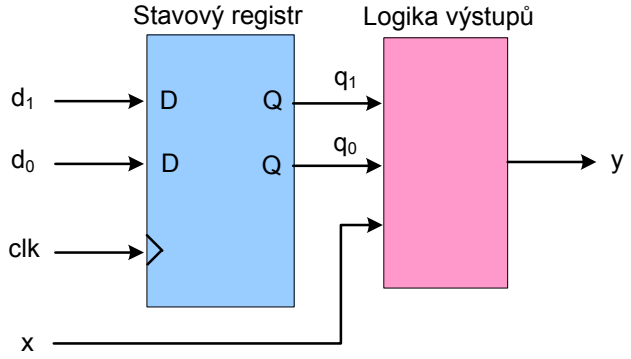

Kódování stavů

| $S_i$ | $q_1$ | $q_0$ | x | $d_1$ | $d_0$ | $S_{i+1}$ |
|---|---|---|---|---|---|---|
| $S_0$ | 0 | 0 | 0 | 0 | 0 | $S_0$ |
| | 0 | 0 | 1 | 0 | 1 | $S_1$ |
| $S_1$ | 0 | 1 | 0 | 0 | 0 | $S_0$ |
| | 0 | 1 | 1 | 1 | 0 | $S_2$ |
| $S_2$ | 1 | 0 | 0 | 0 | 0 | $S_0$ |
| | 1 | 0 | 1 | 1 | 0 | $S_2$ |

# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

Tabulka výstupů

| $S_i$ | $I_0$ | $I_1$ |
|---|---|---|
| $S_0$ | $O_0$ | $O_0$ |
| $S_1$ | $O_0$ | $O_0$ |
| $S_2$ | $O_1$ | $O_0$ |

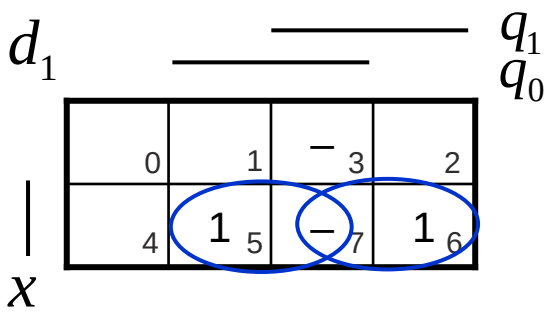



Kódování výstupů

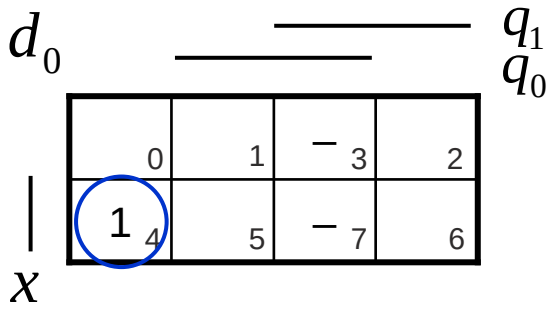
| $S_i$ | $q_1$ | $q_0$ | x | y |
|---|---|---|---|---|
| $S_0$ | 0 | 0 | 0 | 0 |
| | 0 | 0 | 1 | 0 |
| $S_1$ | 0 | 1 | 0 | 0 |
| | 0 | 1 | 1 | 0 |
| $S_2$ | 1 | 0 | 0 | 1 |
| | 1 | 0 | 1 | 0 |

# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

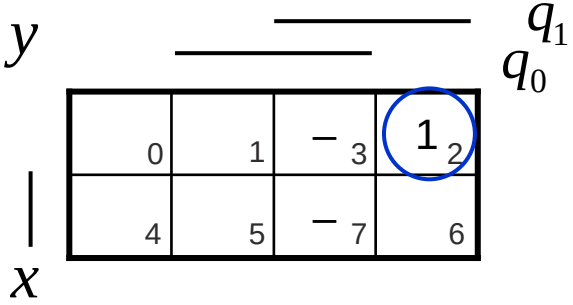
Minimalizace




$$d_1 = q_1 x + q_0 x$$



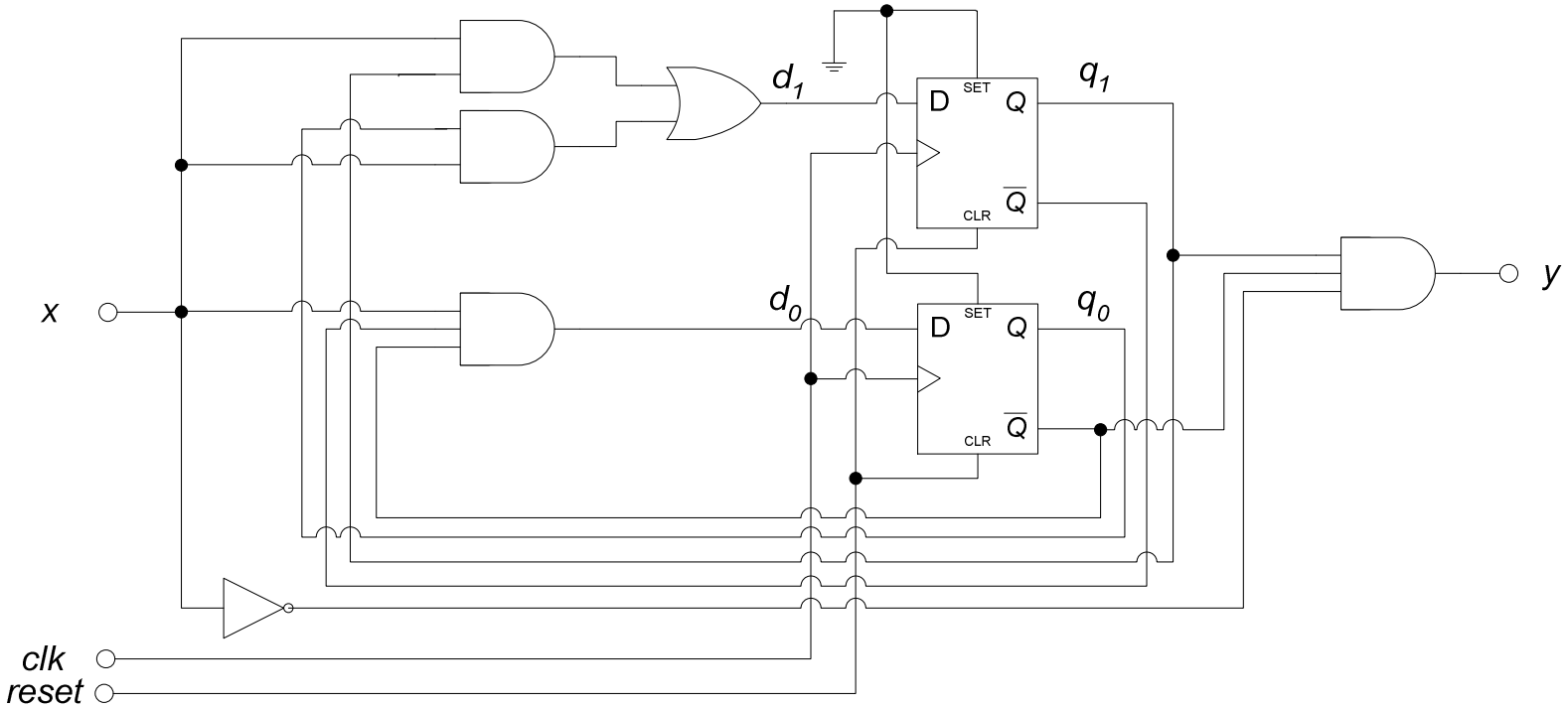

$$d_0 = \overline{q_1}\,\overline{q_0}\,x$$




$$y = q_1\,\overline{q_0}\,\overline{x}$$

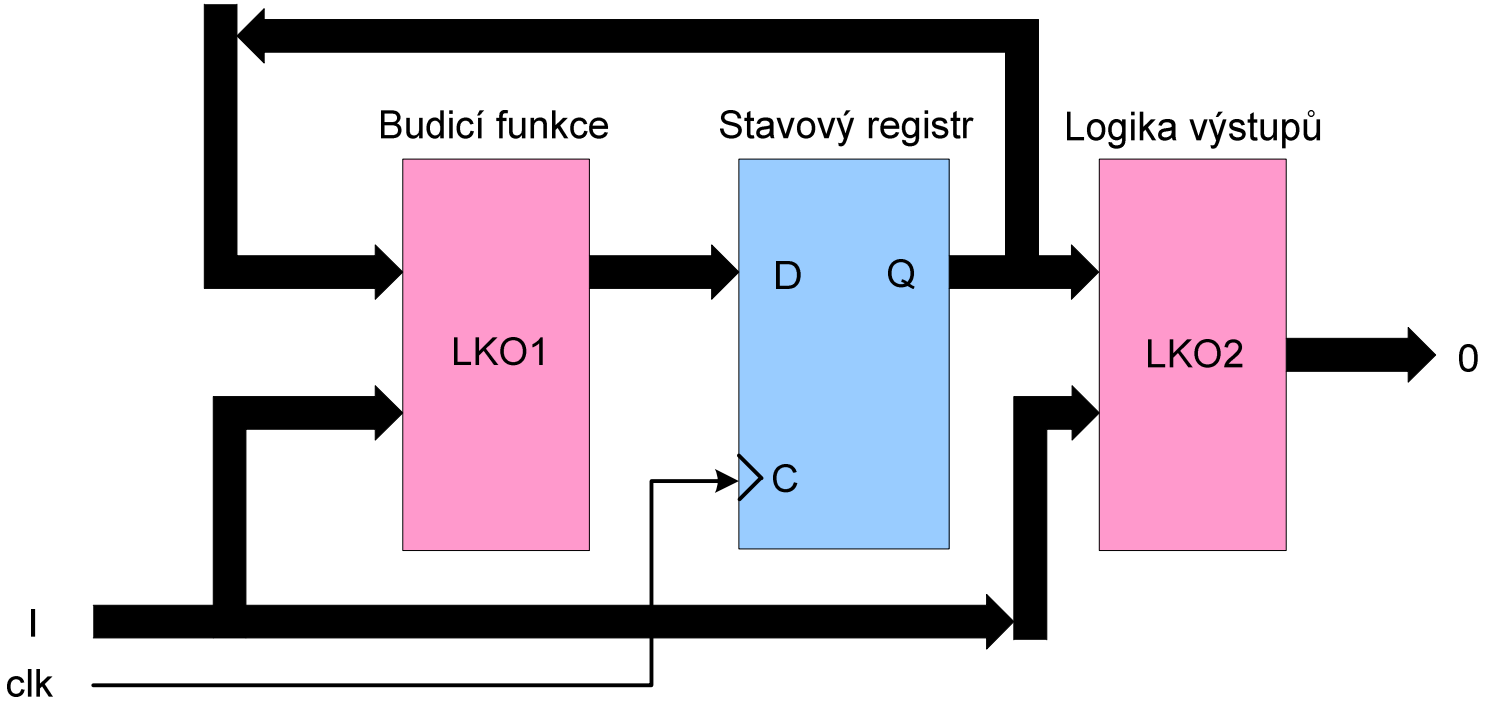
# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

Realizace



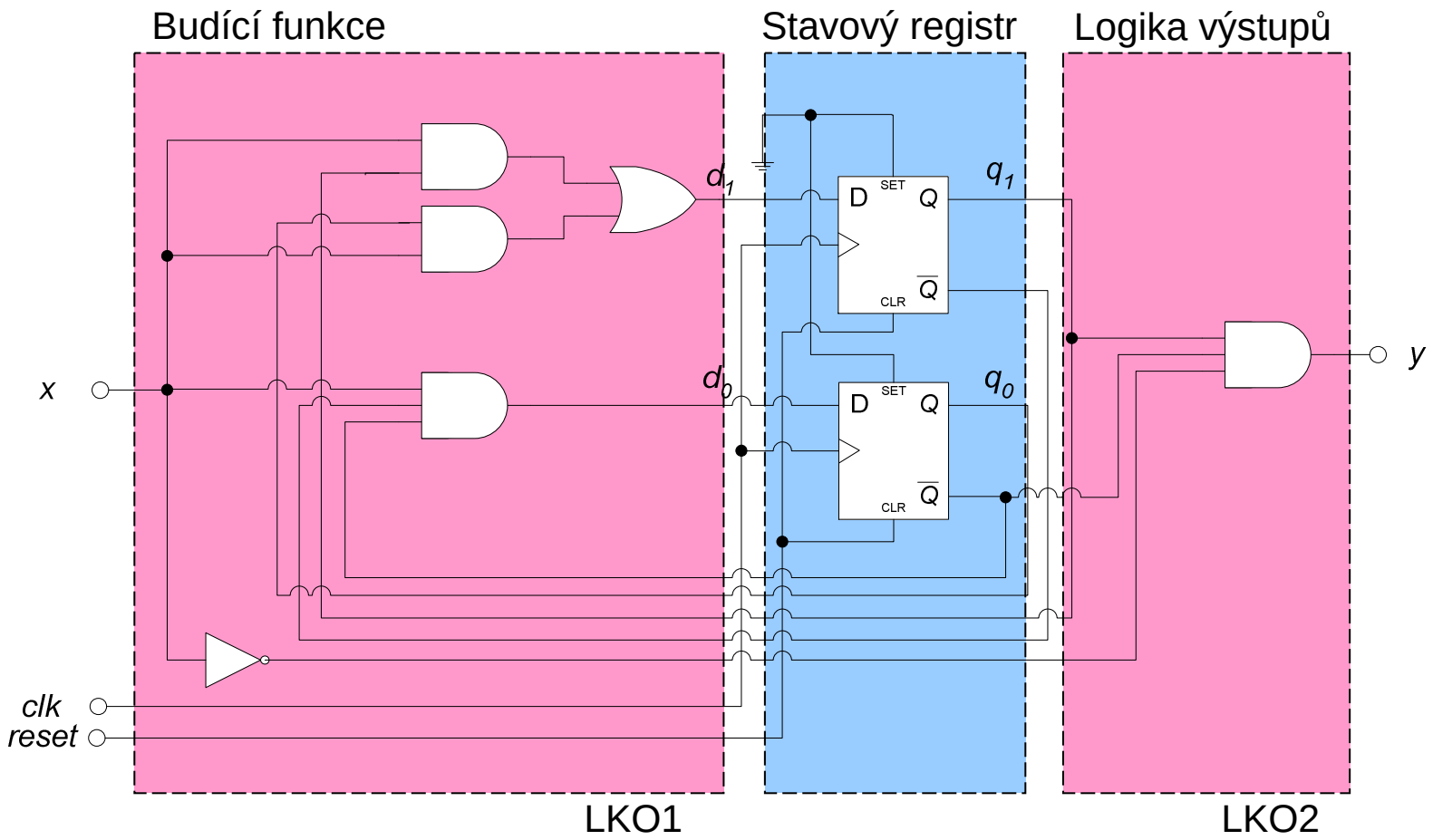
# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

Co jsme navrhli?



# Detektor posloupnosti bitů ‘110’ (FSA typu Mealy)

Co jsme navrhli?



# Časování – výpočet maximální hodinové frekvence

- Ovlivněno:
  - Technologií
  - Typy hradel
  - Počtem vstupů u hradel
  - Zatížením výstupů hradel (větvením)
  - Typem klopných obvodů
  - Délkou propojovacích vodičů (na plošném spoji,…)
  - Vzájemnou polohou vodičů (kvalita návrhu plošného spoje)
  - Rozmístěním součástek
  - Počtem zemnících a napájecích vrstev
  - Způsobem rozvodu napájení
  - Rozmístěním blokovacích kondezátorů
  - Dalšími vlivy …..

Podrobnosti další přednášku

# MIKROPROCESORY PRO VÝKONOVÉ SYSTÉMY

Logické obvody - sekvenční
Formy popisu, konečný automat
Příklady návrhu

KONEC

České vysoké učení technické  Fakulta elektrotechnická