Logické sekvenční obvody

Oproti LKO, kde výstupní hodnota vyjma přechodného děje, byla funkcí vstupních proměnných $x_1,x_2,...,x_n$ (vstupního vektoru), je u sekvenčních obvodů výstupní hodnota závislá i na předcházejících hodnotách vstupního vektoru. Tuto závislost můžeme matematicky vyjádřit rovnicí

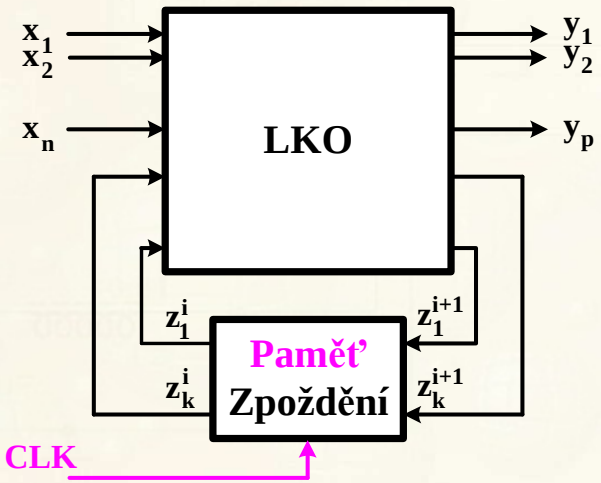
$$y_j^i = F\left(\vec{x}^i, \vec{x}^{i-1}, \vec{x}^{i-2}, ..., \vec{x}^1, \vec{z}^1\right) \quad \text{kde} \quad j = 1,2,...,m \quad \text{a} \quad \vec{x}^i = [x_1^i, x_2^i, ..., x_n^i]$$

Analýza i návrh obvodu založený na uvedeném vztahu je problematický. Chování obvodu rozdělíme do dvou částí se zavedením tzv. **vnitřních proměnných z_k^i**. Nepříjemná časová závislost je tak rozdělena do časových úseků, které souvisí s novým vstupním vektorem a následnými změnami v obvodu nebo s hodinovým synchronizačním kmitočtem obvodu. Předcházející rovnici rozdělíme do dvou rovnic - **výstupů** a **přechodů**.

$$y_j^i = f_j\left(\vec{x}^i, \vec{z}^i\right) = f_j\left(x_1^i, x_2^i, ..., x_n^i, z_1^i, z_2^i, ..., z_m^i\right)$$
$$z_k^{i+1} = g_k\left(\vec{x}^i, \vec{z}^i\right) = g_k\left(x_1^i, x_2^i, ..., x_n^i, z_1^i, z_2^i, ..., z_m^i\right)$$

kde x_r^i a z_k^i jsou vstupní a vnitřní proměnné v čase i. Postupným dosazováním obou rovnic do sebe získáme výše uvedený vztah, kde x^1 a z^1 jsou počáteční stavy obvodu.

Obecně můžeme LSO vytvořit ze dvou bloků - paměti a LKO. LKO realizuje funkce výstupů a přechodů. Proměnné označené i+1 odpovídají následujícímu stavu vnitřních proměnných v závislosti na současném stavu vstupních a vnitřních proměnných v čase i.



Paměť LSO může být realizována

- Zpožděním v logických členech
- Zpožďovacími obvody
- Paměťovými obvody (klopnými obvody).

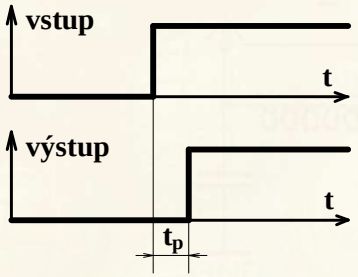
Podle funkcí výstupů můžeme LSO dělit:

- **Mealyho** typ a **Mooreův** typ

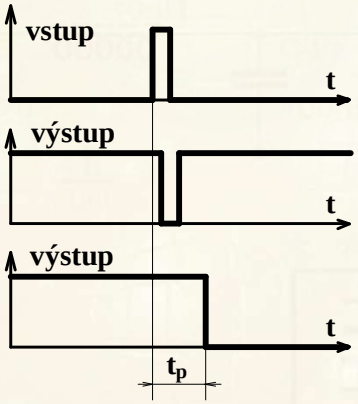
Podle vlastností paměťového bloku dělíme sekvenční obvody na:

- **synchronní obvody** - ke změnám vnitřních proměnných dochází téměř současně.
- **asynchronní obvody** - ke změnám vnitřních proměnných dochází postupně v závislosti na tom, jak se šíří podnět obvodem.

Asynchronní sekvenční obvody můžeme pak dále dělit na obvody s hladinovými vstupy i výstupy a na impulzní obvody s hladinovými nebo impulzními výstupy, kde t_p je doba reakce paměťových členů.



V synchronních sekvenčních obvodech je paměť tvořena synchronizovanými paměťovými členy (klopnými obvody). Změny souvisí s výkonnou hranou hodinového signálu – proto nemohou existovat po sobě následující nestabilní vnitřní stavy obvodu. Časový interval mezi hodinovými impulzy se volí tak, aby v obvodu odezněly všechny přechodné jevy.



U asynchronního obvodu musí být LKO bezhazardní, nežádoucí je současná změna dvou vstupních i vnitřních proměnných.

U synchronního obvodu nemusí být LKO bezhazardní, přípustné jsou současné změny vstupních i vnitřních proměnných. Změny vstupních proměnných nesmí způsobit nedodržení parametrů pro činnost PČ (metastabilní stav) .

Každý LSO je úplně specifikován šesti veličinami

$$LSO \equiv \left[\vec{x}, \vec{y}, \vec{z}, f, g, \vec{z}^{1}\right]$$

Při analýze a návrhu LSO se setkáváme s následujícími pojmy:

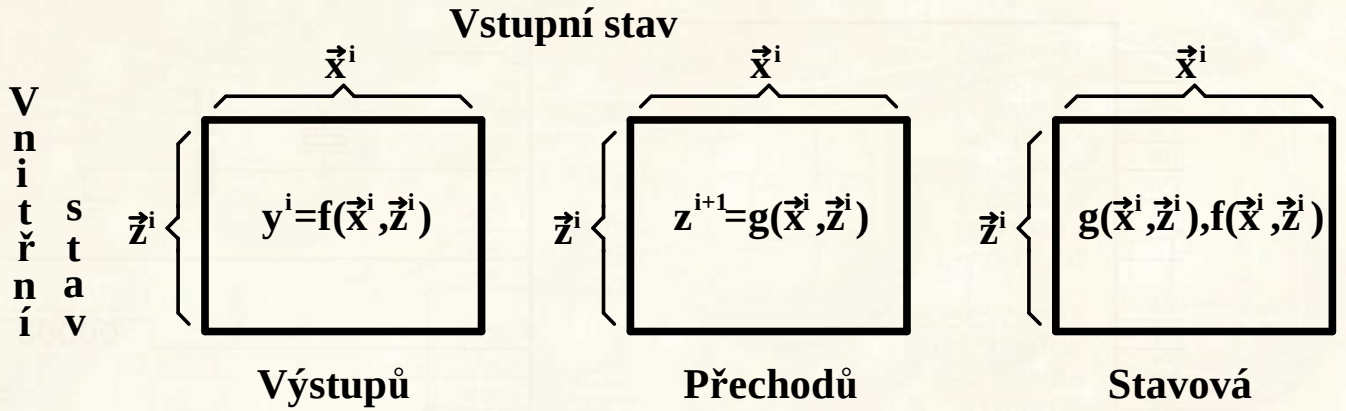
Vnitřní stav obvodu – dán kombinací vnitřních proměnných, (při analýze nebo konečné fázi návrhu obvodu) nebo obecným symbolem (při návrhu obvodu).

Stabilní stav asynchronního obvodu - stav, v němž obvod při konstantním vstupním vektoru může setrvávat neomezenou dobu. V tabulkách jej označujeme kroužkem a matematicky jej můžeme vyjádřit rovnicemi

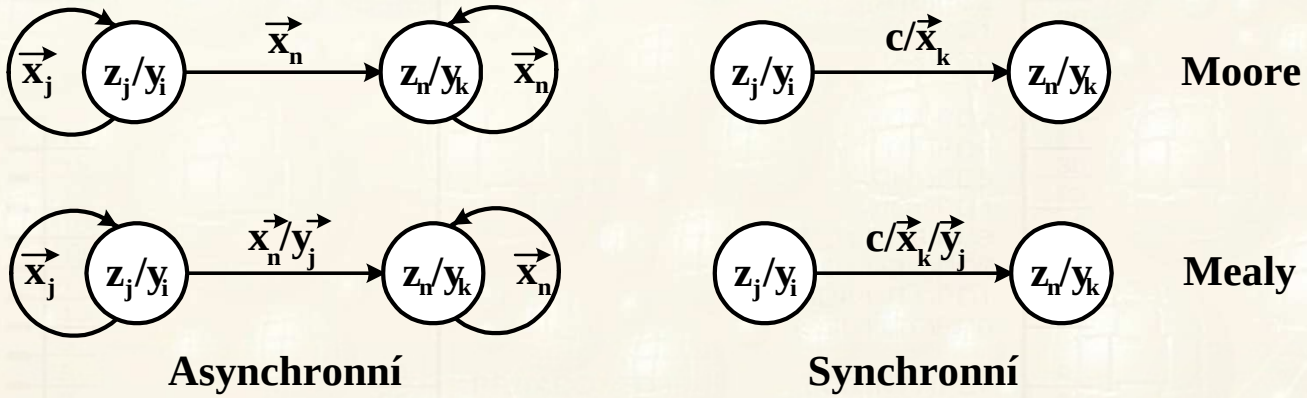
$$\vec{x}^{i+1} = \vec{x}^{i} \qquad \vec{z}^{i+1} = \vec{z}^{i}$$

Funkce přechodů a výstupů mohou být reprezentovány:

- Algebraickým výrazem - využívá se při analýze a v závěrečné fázi návrhu sekvenčního obvodu při přechodu od jeho struktury ke stavové tabulce a obráceně.
- Tabelárním vyjádřením - tabulkami z následujícího obrázku. Vývojová tabulka se shoduje se stavovou s tím, že vnitřní stavy jsou vyjádřeny obecně.

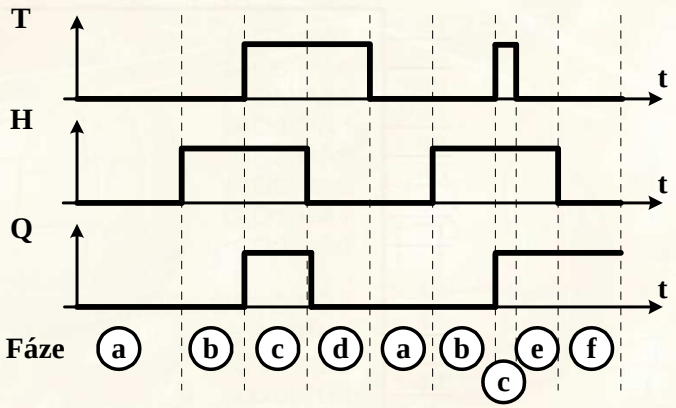


- Strukturou – Schéma je výchozím vyjádřením obvodu před jeho analýzou a cílovým stavem při syntéze sekvenčního obvodu.
- Grafem – jedním z možných způsobů zadání chování LSO nebo poslední fází v analýze funkce LSO.

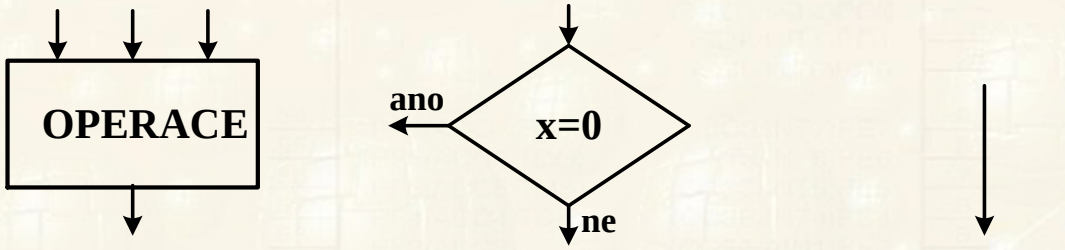


- Časovým nebo fázovým diagramem – vstupních a výstupních signálů obvodu viz následující obrázek.

x_1 x_2 0 0	0 1	1 1	1 0	y
①	5	7	4	0
②	6	8	4	1
1	6	7	③	0
1	6	8	④	1
1	5	⑦	3	0
2	5	⑧	4	1
1	⑤	8	3	0
2	⑥	8	3	1



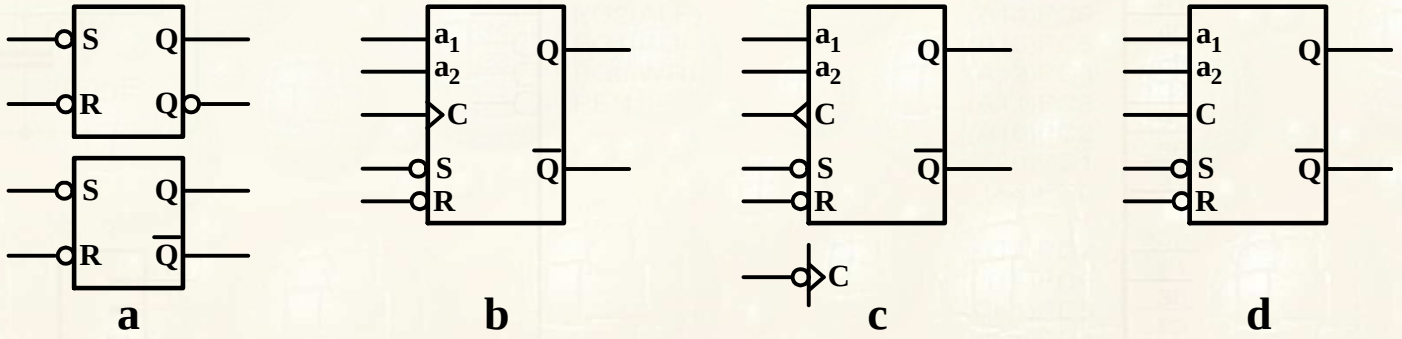
- Fázovou tabulkou – návrh i analýza asynchronního sekvenčního obvodu. V každém řádku je jeden stabilní stav
- Jazykem HDL – určeným pro návrh složitých logických obvodů. Firemní jazyky - AHDL, LHDL, ABEL, atd. vznikly s vývojem PLD u jednotlivých firem – **nepřenosné**. **Verilog** a **VHDL** jsou univerzálními jazyky pro návrh logických obvodů.
- Vývojovým (programovým) diagramem



PAMĚŤOVÉ ČLENY

Paměťové členy (klopné obvody) - asynchronní LSO, které mají dva různé výstupní stavy. Využívají se jako paměť hodnoty logické proměnné. Paměťové členy dělíme je podle jejich vlastností:

- asynchronně řízené
- synchronně řízené
  - hladinovým signálem
  - náběžnou hranou synchronizačního signálu
  - sestupnou hranou synchronizačního signálu

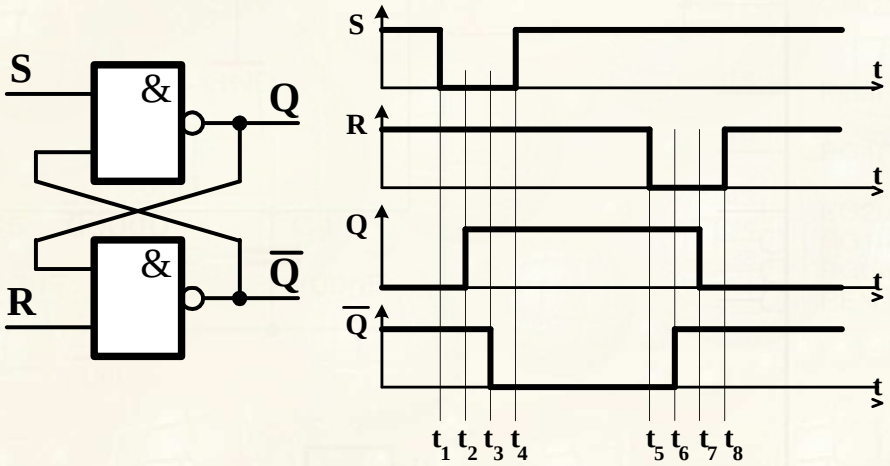


Schematické značení jednotlivých typů paměťových členů je na obrázku, kde a) značí asynchronní obvod s aktivní úrovní v log.0, b) obvod synchronizovaný náběžnou hranou, c) obvod synchronizovaný sestupnou hranou a d) obvod synchronizovaný úrovní hodinového signálu. PČ mohou mít asynchronní nastavovací vstupy (R,S).

Vstup nastavení se obvykle označuje S (set) a nulovací R (reset), jsou-li v aktivní úrovni PČ nereaguje na synchronizační signál.

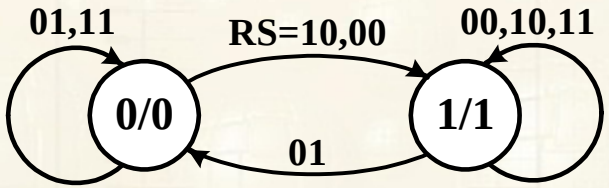
Paměťový člen RS

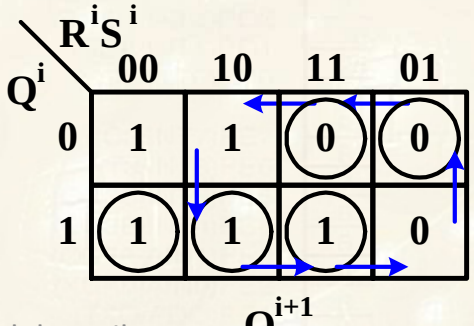
Paměťový člen RS je asynchronně řízený obvod se signály R (reset) a S (set). Pro obvod RS-NAND jsou vstupy S a R aktivní v log.0.



Z časových průběhů s respektováním zpoždění je zřejmé, že minimální šířka signálu musí být větší než doba $\Delta t = t_3 - t_1 \approx 2.t_{pd}$, kde t_{pd} je střední doba zpoždění signálu logického členu NAND.

Zobrazen je stavový diagram a stavová tabulka s označení běžných přechodů obvodu.





PAMĚŤOVÝ ČLEN – RS

R^i S^i	Q^{i+1}
0 0	zak. stav
0 1	0
1 0	1
1 1	Q^i

$Q^i \rightarrow Q^{i+1}$	R^i S^i
0 → 0	X 1
0 → 1	1 0
1 → 0	0 1
1 → 1	1 X

Zobrazena je pravdivostní tabulka a tabulka přechodů nezbytná pro návrh LSO s paměťovými členy RS. Operátor paměťového členu RS NAND je vyjádřen rovnicí

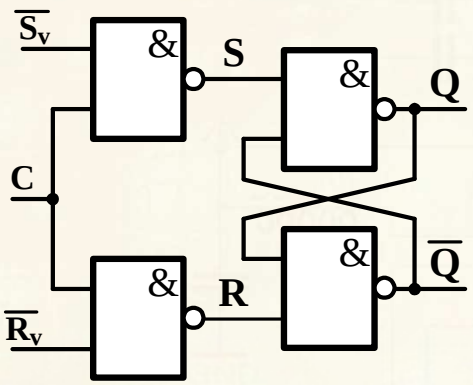
$$Q^{i+1} = \overline{S}^i + R^i.Q^i \quad pro \quad \overline{R}^i.\overline{S}^i = 0$$

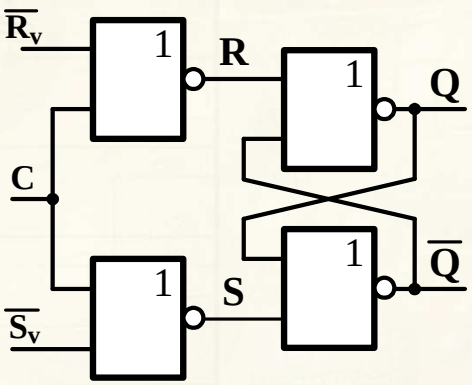
Stav R=S=0 se označuje jako zakázaný (nepřípustný), protože při něm nejsou výstupy obvodu vzájemně komplementární. Při současné změně vstupů R a S ze stavu 0 do stavu 1 není jisté zda výstup Q = 0 nebo Q=1.

Hladinově řízené paměťové členy

Paměťové člen RS byl následně upraven pro synchronní řízení hladinovým signálem. Výstupní stav determinovaný stavem vstupních proměnných se přenese na výstup pouze při splnění podmínky synchronizace. Pokud obvod není synchronizován, jeho výstupní stav se nemění a obvod je ve stabilním stavu.

Paměťový člen RST





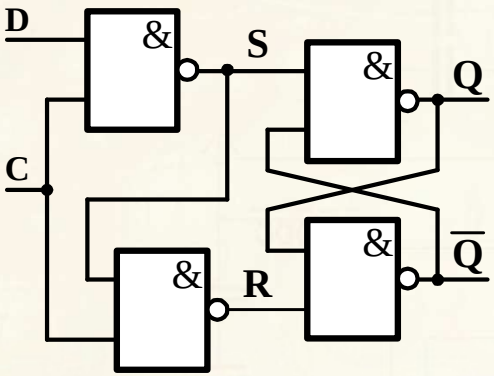
Zobrazeny obě varianty realizace paměťových členů RST hradly NAND i NOR. Obvody jsou synchronizovány úrovní vstupní proměnné C (Clock) nebo někdy označované H. Bude-li pro variantu NAND C=0 zůstává výstup obvodu Q nezávislý na vstupních hodnotách R_v a S_v, a pro C=1 se obvod chová jako paměťový člen RS. Chování obvodů můžeme popsat těmito vztahy

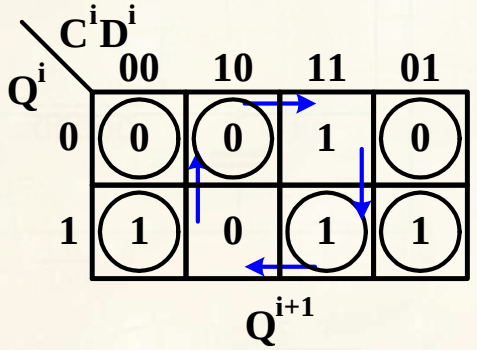
$$Q^{i+1} = C^i.\overline{S}_v^i + \left(R_v^i + \overline{C}^i\right).Q^i \quad \text{pro} \quad \text{RST} - \text{NAND}$$

$$Q^{i+1} = \left(\overline{C}^i + \overline{R}_v^i\right).\left(S_v^i.C^i + Q^i\right) \quad \text{pro} \quad \text{RST} - \text{NOR}$$

Proměnné R_v a S_v mají být v době, kdy je obvod synchronizován (C=1), konstantní. Při sériovém řazení paměťových členů tohoto typu se dá splnit jednofázovou impulzní synchronizací, při úzkém impulzu signálu C. Šířka impulzu C je ovšem kritická a proto se častěji používá dvou a vícefázové řízení.

Paměťový člen D - Delay





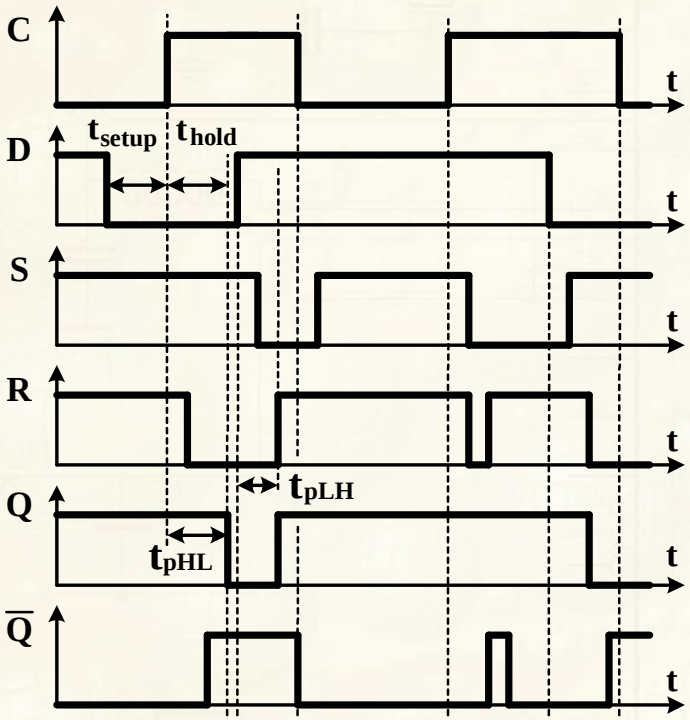
Základem paměťového členu D je opět paměťový člen RS, který pro C=0 má na svých vstupech R a S neaktivní úrovně (log.1). Výstup si zachovává poslední stav nezávisle na vstupní proměnné D. V okamžiku, kdy hodinový signál C=1, je na vstup R přivedena vstupní proměnná D a na vstup S její negace. Výstup Q potom kopíruje stav vstupní proměnné D, jak vyplývá i ze stavové tabulky. Operátor paměťového členu D je dán touto rovnicí

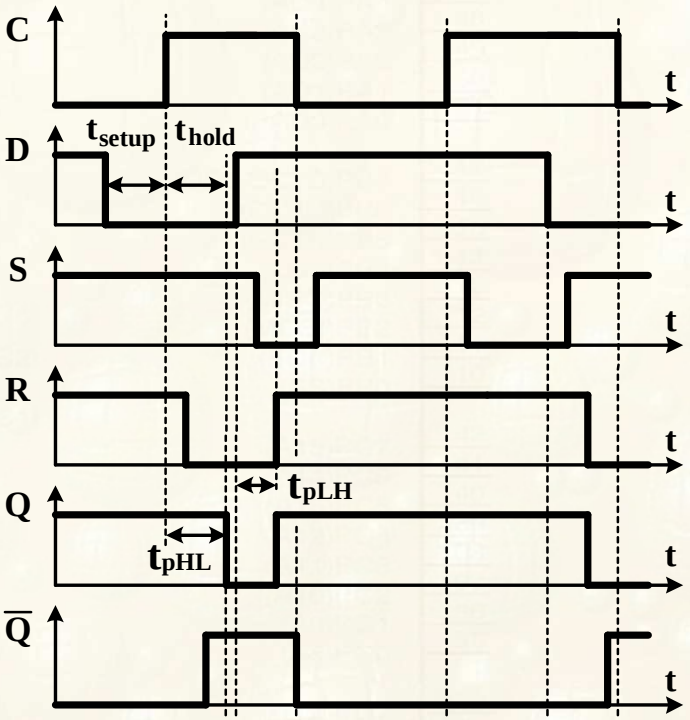
$$Q^{i+1} = \overline{C}^i . Q^i + C^i . D^i$$

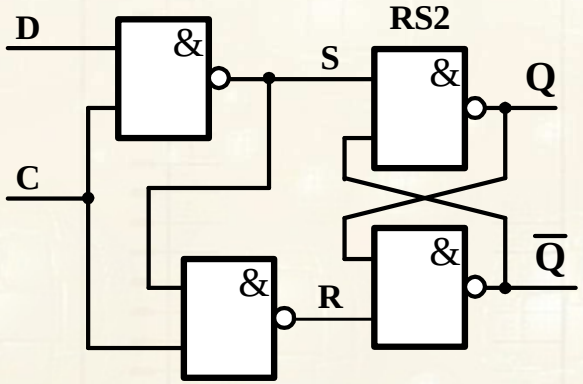
Clock	Data	Vstupy RS	Q^{i+1}	$nonQ^{i+1}$	Poznámka
0	0 or 1	1 1	Q^i	$nonQ^i$	žádná změna
1	0	0 1	0	1	Nulování Q
1	1	1 0	1	0	Nastavení Q

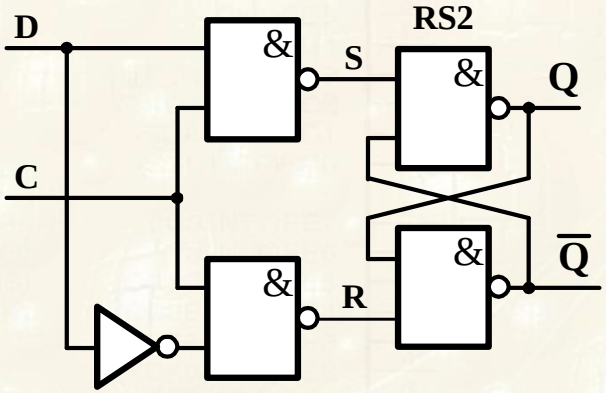
PAMĚŤOVÝ ČLEN – D (LATCH)

Při uvažování zpoždění log.členů je situace v chování PČ D složitější. Varianta s pěti hradly má lepší chování při změně D při C=1.









PAMĚŤOVÝ ČLEN – D (LATCH) IMPLEMENTACE VE VHDL

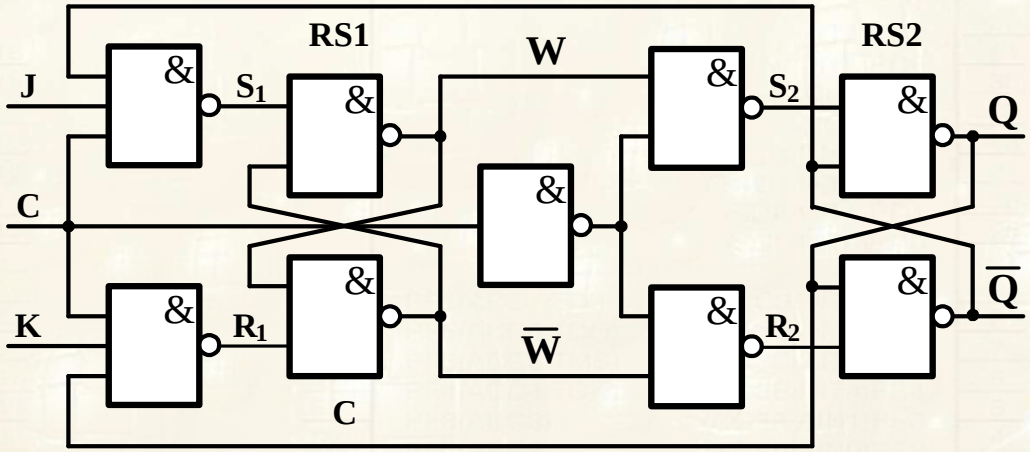
```
library ieee;
use ieee.std_logic_1164.all;

entity d_latch is
     port (H, D: in std_logic;
               Q, nQ: out std_logic);
end entity d_latch;

architecture behavior of d_latch is
     begin
p1:    process (H, D)
               begin
              if  H = '1'  then
               Q <= D;
               nQ <= not D;
               end if;
          end process;
end behavior;
```

PČ synchronizované hranami hodinového signálu

Hranově synchronizované PČ ovlivňují výstup hodnotami vstupních signálů v okamžicích změny synchronizačního (hodinového) signálu. Jsou obvykle tvořeny několika paměťovými členy mezi nimiž je přenos signálu řízen alespoň dvoufázově.

Paměťový člen JK

Paměťový člen JK (Master-Slave) je příkladem členu s dvoufázovým přenosem dat odvozeným od jednoho hodinového signálu. Náběžnou hranou je vstupní informace zapsána do RS1 a se závěrnou hranou se přepíše do členu RS2. Zjednodušená realizace starší varianty tohoto obvodu je zobrazena.



PAMĚŤOVÝ ČLEN – JK (MASTER-SLAVE)

Při analýze JK ztotožníme výstupy RS s vnitřními proměnnými LSO. Obvod má vstupní proměnné J,K,C, výstupní proměnnou Q a vnitřní proměnné W a Q. Pro funkce buzení vstupů paměťových členů můžeme psát

$$R_1 = \overline{C} + \overline{K} + \overline{Q} \qquad S_1 = \overline{C} + \overline{J} + Q$$

$$R_2 = C + W \qquad S_2 = C + \overline{W}$$

W^iQ^i \ $J^iK^iC^i$	000	100	110	010	001	111	101	001
00	(00)	(00)	(00)	(00)	(00)	10	10	(00)
10	11	11	11	11	(10)	(10)	(10)	(10)
01	(11)	(11)	(11)	(11)	01	01	(11)	(11)
11	00	00	00	00	01	(01)	(01)	(01)

$W^{i+1}Q^{i+1}$

Po dosazení do operátoru paměťového $Q^{i+1} = \overline{S}^i + R^i.Q^i$ členu dostáváme tyto funkce přechodů pro vnitřní proměnné W a Q

$$W^{i+1} = C^i.J^i.\overline{Q}^i + \left(\overline{C}^i + \overline{K}^i + \overline{Q}^i\right).W^i$$

$$Q^{i+1} = \overline{C}^i.W^i + \left(C^i + W^i\right).Q^i$$

Z rovnic přechodů snadno vytvoříme stavovou tabulku obvodu, v které zakroužkujeme stabilní stavy. Obvyklá činnost obvodu předpokládá, že s náběžnou hranou signálu C se podle stavu vstupů JK nastaví paměťový člen RS1 a se závěrnou hranou C se informace z členu RS1 přepíše do členu RS2.

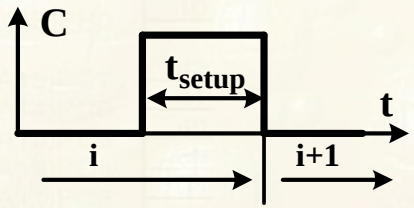
PAMĚŤOVÝ ČLEN – JK (MASTER-SLAVE)

Činnost obvodu popíšeme pravdivostní tabulkou, operátorem a tabulkou přechodů (jedna z možností při návrhu obvodu s PČ JK. Čas **i** se počítá do sestupné hrany signálu C a doba předstihu je rovna šířce hodinového impulzu v log.1.

J^i K^i	Q^{i+1}
0 0	Q^i
0 1	0
1 0	1
1 1	$\overline{Q^i}$

$Q^i \rightarrow Q^{i+1}$	J^i K^i
$0 \rightarrow 0$	0 X
$0 \rightarrow 1$	1 X
$1 \rightarrow 0$	X 1
$1 \rightarrow 1$	X 0

$$Q^{i+1} = J^i.\overline{Q}^i + \overline{K}^i.Q^i$$



Paměťový člen JK se dnes vyrábí ve variantě čistě hranově řízeného obvodu, reagujícího na sestupnou hranu hodinového signálu a dobou předstihu rovnou několika ns před sestupnou hranou.

Na následujícím obrázku je zobrazena implementace paměťového členu JK v jazyce VHDL, reagujícího na náběžnou hranu signálu CLK.

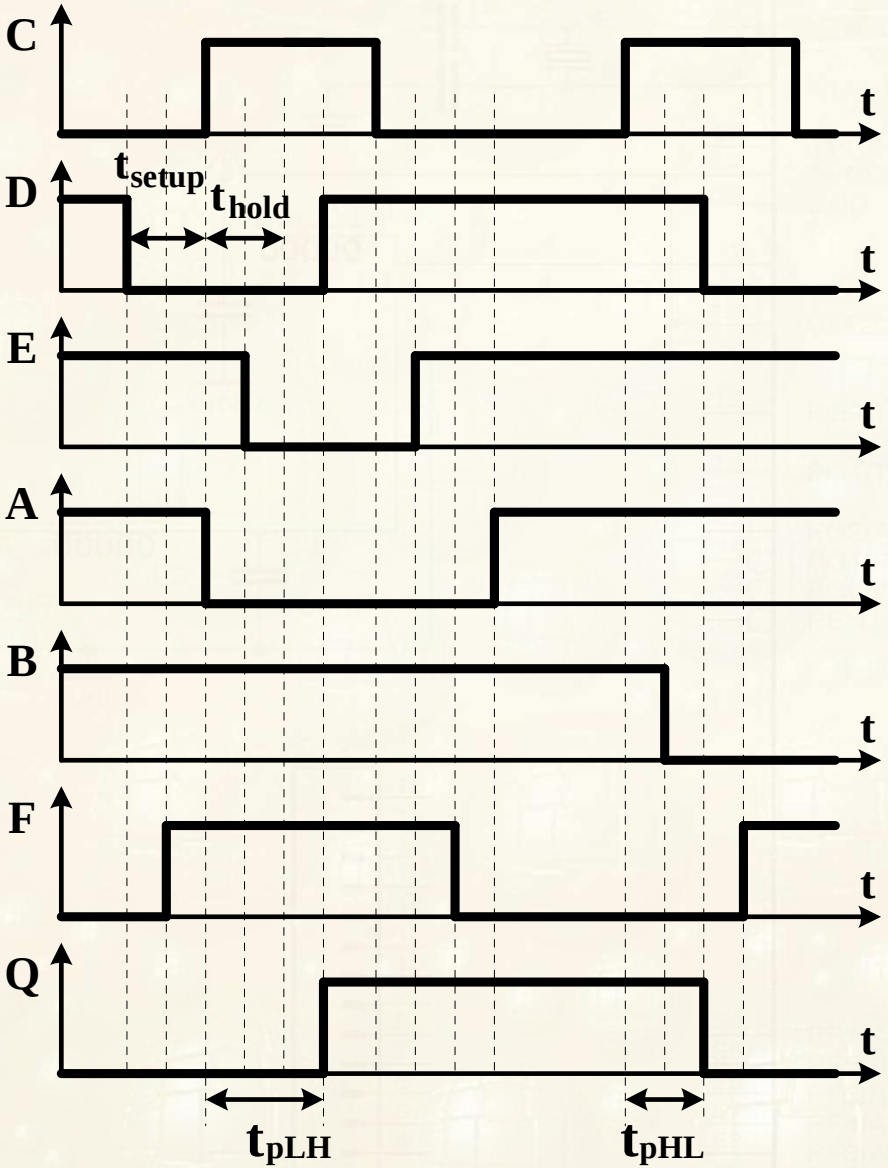
IMPLEMENTACE PAMĚŤOVÉHO ČLENU JK VE VHDL

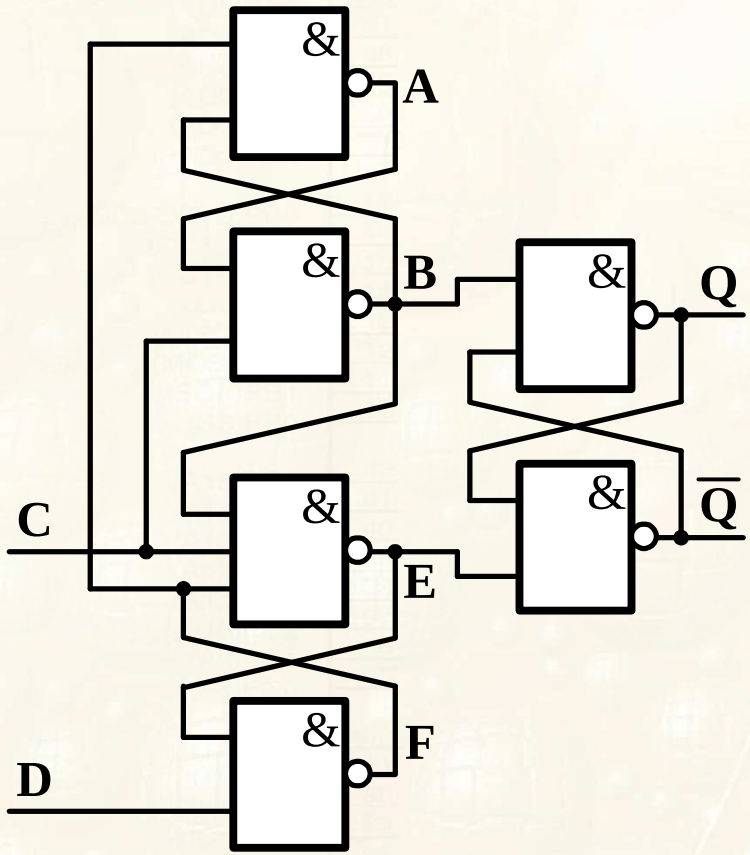
```
architecture behavior of JK_FF is
signal temp: std_logic;
begin
process (clk, reset) is
begin
if (Reset = '1') then
temp <= '0';
elsif (clk'event and clk = '1') then     -- Náběžná hrana
if  (J = '1' and K = '1') then
temp <= not temp;
elsif  (J = '1' and K = '0') then
temp <= '1';
elsif  (J ='0' and K = '1') then
temp <= '0';
else
temp <= temp;
end if;
end if;
end process;
Q <= temp;
end behavior;
```

Paměťový člen D hranově řízený

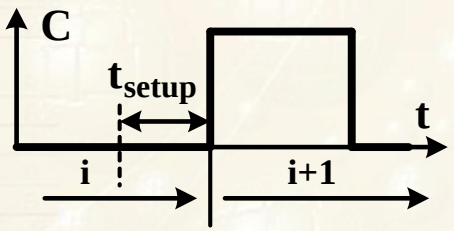
U hranově řízeného PČ D se využívá zpětná vazba, která brání průchodu vstupního signálu D strukturou obvodu bezprostředně po náběžné hraně synchronizačního signálu C. Na obrázku je zobrazeno zjednodušené zapojení paměťového členu D (bez asynchronních vstupů R a S) spolu s časovými průběhy vstupních, vnitřních a výstupních signálů. Při D=0 je další přenos případných změn signálu D blokován v logickém členu signálem E=0 a při D=1 se blokování realizuje v logickém členu signálem B=0. Na obrázku jsou zobrazeny zpoždění v jednotlivých částech obvodu doby v nichž probíhá zápis vstupní proměnné D do vnitřních paměťových členů a dále přenos stavu vnitřního paměťového členu na výstup Q. Při malé strmosti náběžné hrany hodinového signálu dochází k prodloužení těchto dob. Z činnosti paměťového členu vyplývá, že v okolí aktivní (náběžné) hrany synchronizačního signálu musí být vstupní signál D konstantní. Doby předstihu a přesahu signálu D vůči náběžné hraně synchronizačního signálu jsou znázorněny a jejich nedodržení způsobuje u PČ **metastabilní** stav.

PAMĚŤOVÝ ČLEN – D HRANOVĚ ŘÍZENÝ





$$Q^{i+1} = D^{i}$$



PAMĚŤOVÝ ČLEN – D HRANOVĚ ŘÍZENÝ

Některé paměťové členy jsou vybaveny dalšími vstupy R a S, které umožňují asynchronní nebo synchronní nulování nebo nastavení výstupu obvodu. Tyto vstupy mají **vyšší prioritu** a nastavují výstup obvodu bez ohledu na stav hodinového signálu a vstupu D nebo JK, jak ukazuje následující tabulka.

C	D	Vstupy RS	Q^{i+1}	$nonQ^{i+1}$	Poznámka
↓	0 nebo 1	1 1	Q^i	$nonQ^i$	Žádná změna výstupu Q
↑	0	1 1	0	1	Synchronní nulování výstupu Q
↑	1	1 1	1	0	Synchronní nastavení výstupu Q
↑	0 nebo 1	0 1	0	1	Asynchronní nulování výstupu Q
↑	0 nebo 1	1 0	1	0	Asynchronní nastavení výstupu Q

IMPLEMENTACE PAMĚŤOVÉHO ČLENU D ŘÍZENÉHO HRANOU VE VHDL

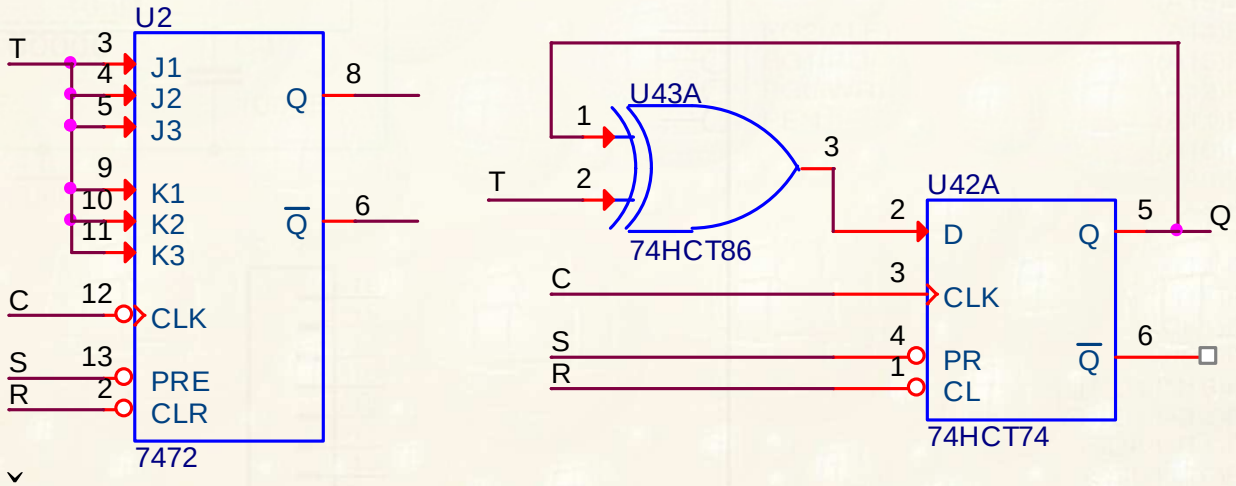
```
library ieee;
use ieee.std_logic_1164.all;

entity D_FF is
port (D, clk: in std_logic;
Q: out std_logic);
end entity D_FF;

architecture behavior of D_FF is
begin
process (clk)
begin
if clk' event and clk = '1' then
Q <= D;
end if;
end process;
end behavior;
```

Paměťový člen T

Paměťový člen T (synchronní invertor) je obvod, který při vstupním signálu T=1 vždy po příchodu synchronizačního signálu změní hodnotu svého výstupu (Q). Pro vstupní signál T=0 se hodnota výstupu nemění.

$$Q^{i+1} = T^i \oplus Q^i = \overline{T}^i . Q^i + T^i . \overline{Q}^i$$

Paměťový člen T není v integrované podobě běžně vyráběn, ale lze jej realizovat pomocí členu JK (J=K=T) nebo členu D (D=Q.T).



Paměťový člen E

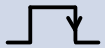
J^i K^i	Q^{i+1}
0 0	Q^i
0 1	1
1 0	0
1 1	Q^i

PČ E je dalším paměťovým členem, který se nevyrábí. Jeho pravdivostní tabulka je kromě kombinace JK=11 shodná s pravdivostní tabulkou paměťového členu JK.

IMPLEMENTACE PAMĚŤOVÉHO ČLENU T VE VHDL

```
entity T_FF is
port (T, clk, reset: in std_logic;
Q, negQ: out std_logic);
end T_FF;

architecture behavior of T_FF is
signal temp: std_logic;
begin
process (clk, reset) is
begin
if  (reset = '1') then
temp <= '0';
elsif  (clk'event and clk = '1') then
if  T = '1' then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
negQ <= not temp;
end behavior;
```

PAMĚŤOVÉ ČLENY – PŘEHLED NEJBĚŽNĚJŠÍCH OBVODŮ

Obvod	Typ	Počet	Q	neg.Q	R	S	Synch.	Výstup
74xxx73	JK	2	ano	ano	ano	ne		2 stav.
74xxx74	D	2	ano	ano	ano	ano		2 stav.
74xxx75	D	4	ano	ano	ne	ne		2 stav.
74xxx108	JK	2	ano	ano	ano	ano		2 stav.
74xxx174	D	6	ano	ne	ano	ne		2 stav.
74xxx175	D	4	ano	ano	ano	ne		2 stav.
74xxx173	D	4	ano	ne	ano	ne		3 stav.
74xxx279	RS_{NAND}	4	ano	ne	ano	ano		2 stav.
74xxx373	D	8	ano	ne	ne	ne		3 stav.
74xxx374	D	8	ano	ne	ne	ne		3 stav.
74xxx573	D	8	ano	ne	ne	ne		3 stav.
74xxx574	D	8	ano	ne	ne	ne		3 stav.

Návrh synchronního LSO může vycházet z požadovaných časových průběhů, stavového diagramu, slovního zadání, atd. Ukážeme si klasický návrh s využitím logických rovnic. **Příklad**: Navrhněte Johansonův čítač modulo 6.

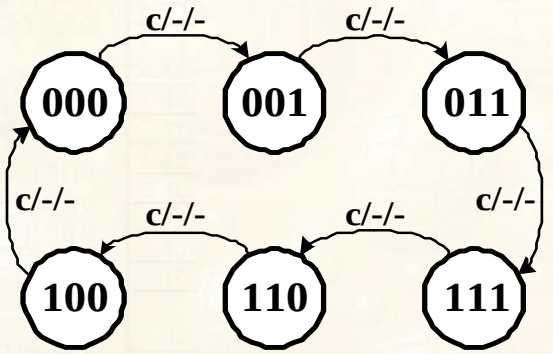
1) Vytvoříme stavový diagram.
2) Ze stavového diagramu vytvoříme stavovou tabulku
3) Ze stavové tabulky odvodíme rovnice přechodů pro Q3, Q2, Q1

$$Q_1^{i+1} = Q_1^i.\overline{Q}_2^i + \overline{Q}_3^i \qquad Q_2^{i+1} = Q_1^i \qquad Q_3^{i+1} = Q_2^i$$

4) Porovnáním rovnic přechodů s operátorem použitých PČ odvodíme rovnice buzení vstupů PČ. Pro paměťový člen D

$$Q_j^{i+1} = D_j^i$$

$$D_1 = Q_1.\overline{Q}_2 + \overline{Q}_3 \qquad D_2 = Q_1 \qquad D_3 = Q_2$$



Q_3^i \ $Q_1^iQ_2^i$	00	10	11	01
0	001	011	111	xxx
1	000	xxx	110	100

$Q_3^{i+1}Q_2^{i+1}Q_1^{i+1}$

Q_3^i \ $Q_1^iQ_2^i$	00	10	11	01
0	001	011	111	101
1	000	011	110	100

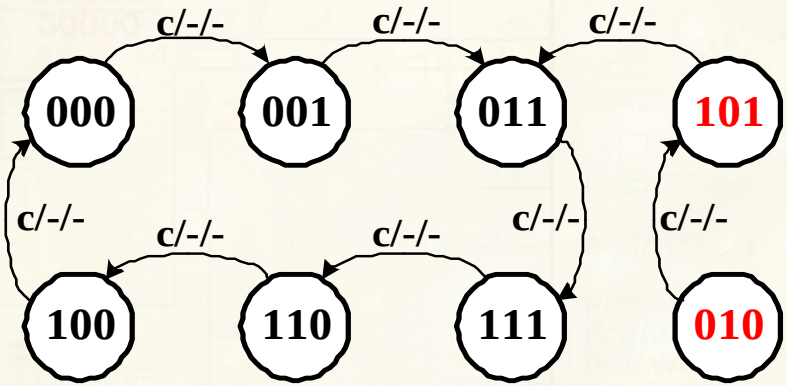
$Q_3^{i+1}Q_2^{i+1}Q_1^{i+1}$

Při odvození rovnic přechodů využijeme neurčitých stavů u nevyužitých stavů automatu (označeny červeně).

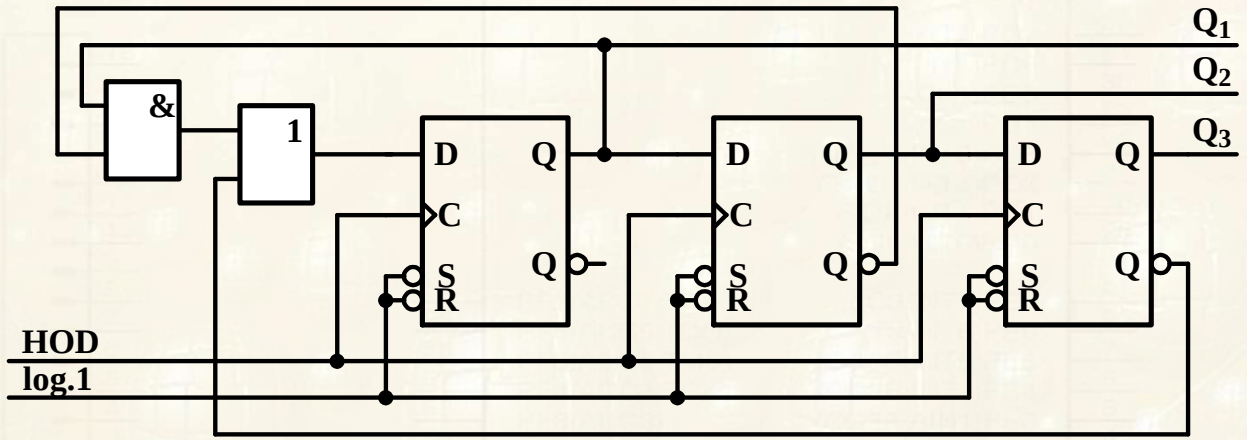
Q_3^i \ $Q_1^i Q_2^i$	00	10	11	01
0	001	011	111	101
1	000	011	110	100

$Q_3^{i+1} Q_2^{i+1} Q_1^{i+1}$

5) Před vytvoření schématu ověříme přechody z nevyužitých stavů.



6) Nakreslíme navržené schéma obvodu

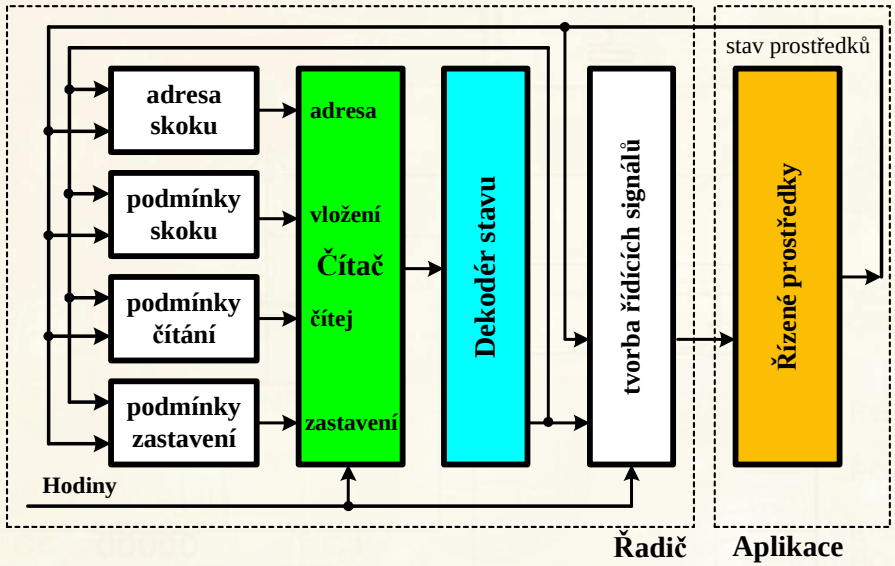


IMPLEMENTACE BINÁRNÍHO ČÍTAČE

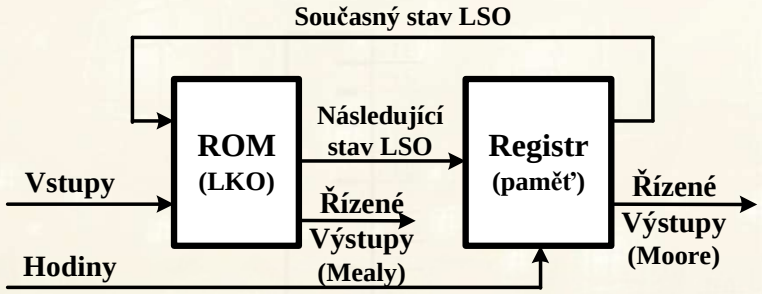
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity upcounter is
port (clk, reset: in std_logic;
Q: out std_logic_vector (3 downto 0));
end upcounter;

architecture behavior of upcounter is
signal count: std_logic_vector (3 downto 0);
begin
    process (clk, reset)
    begin
        if reset = '1' then
        count <= (others => '0');
        elsif clk' event and clk = '1' then
        count <= count + 1;
        end if;
    end process;
    Q <= count;
end behavior;
```

ŘADIČ – KLASICKÉ OBVODOVÉ ZAPOJENÍ

Obvodový řadič se skládá jako jakýkoliv jiný sekvenční obvod (LSO) z paměťových členů (vnitřní proměnné), generovaných signálů (výstupy Mealy, Moore) , funkcí přechodů určující následující stav obvodu.

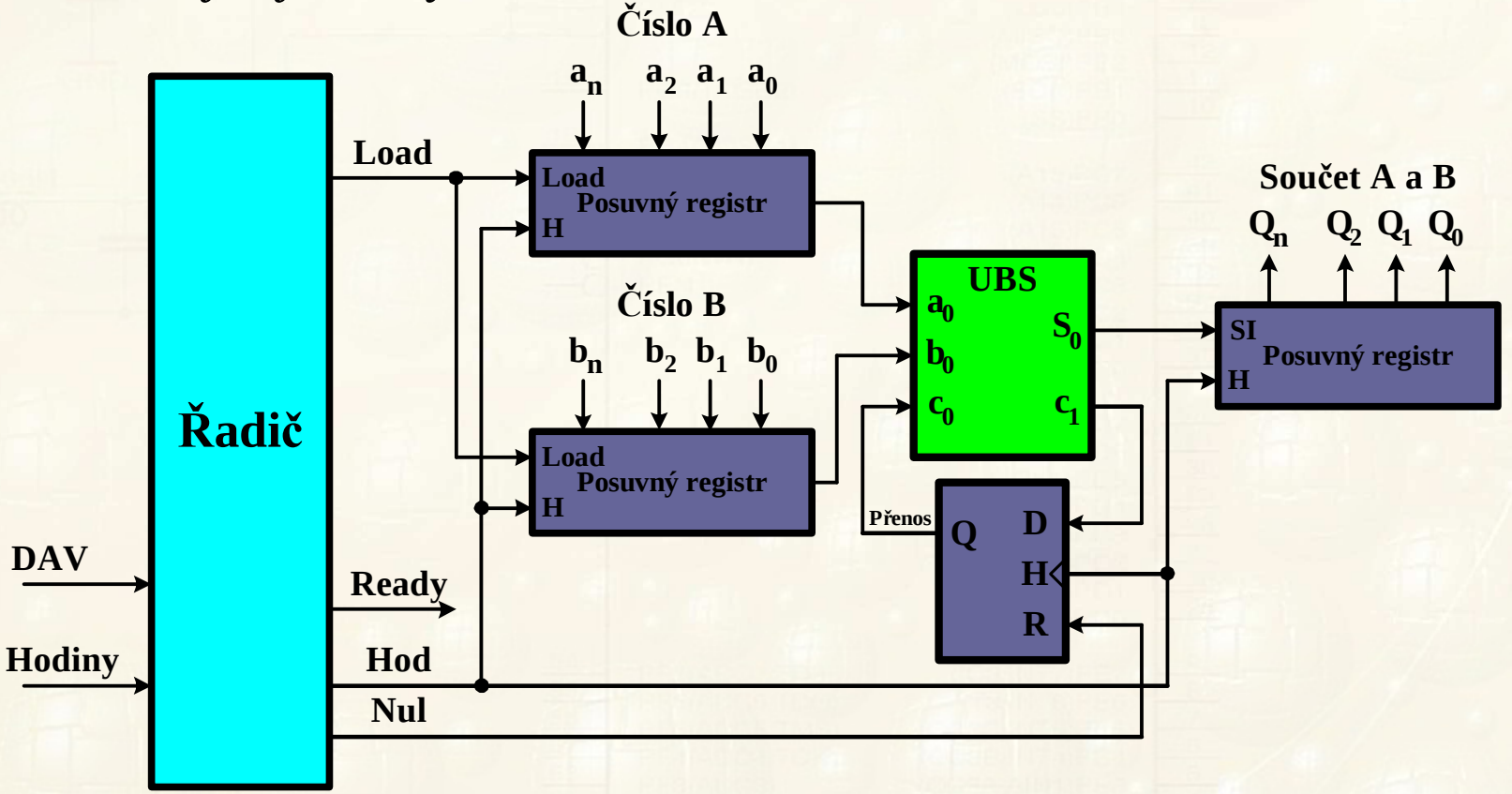


- **Složité řadiče** - čítač, posuvný registr dekódovaný na řídící signály.
- Pro snazší změnu řídících signálů byl dekodér nahrazen **pamětí**
- **Jednodušší řadič** může být vytvořen ze dvou obvodů tzv. **automodifikace registru.**
- Paměťovou funkci realizuje **registr** a paměť **ROM** (PROM, RAM nebo LKO v PLD) vytváří kombinační část obvodu (funkce přechodů a výstupů).



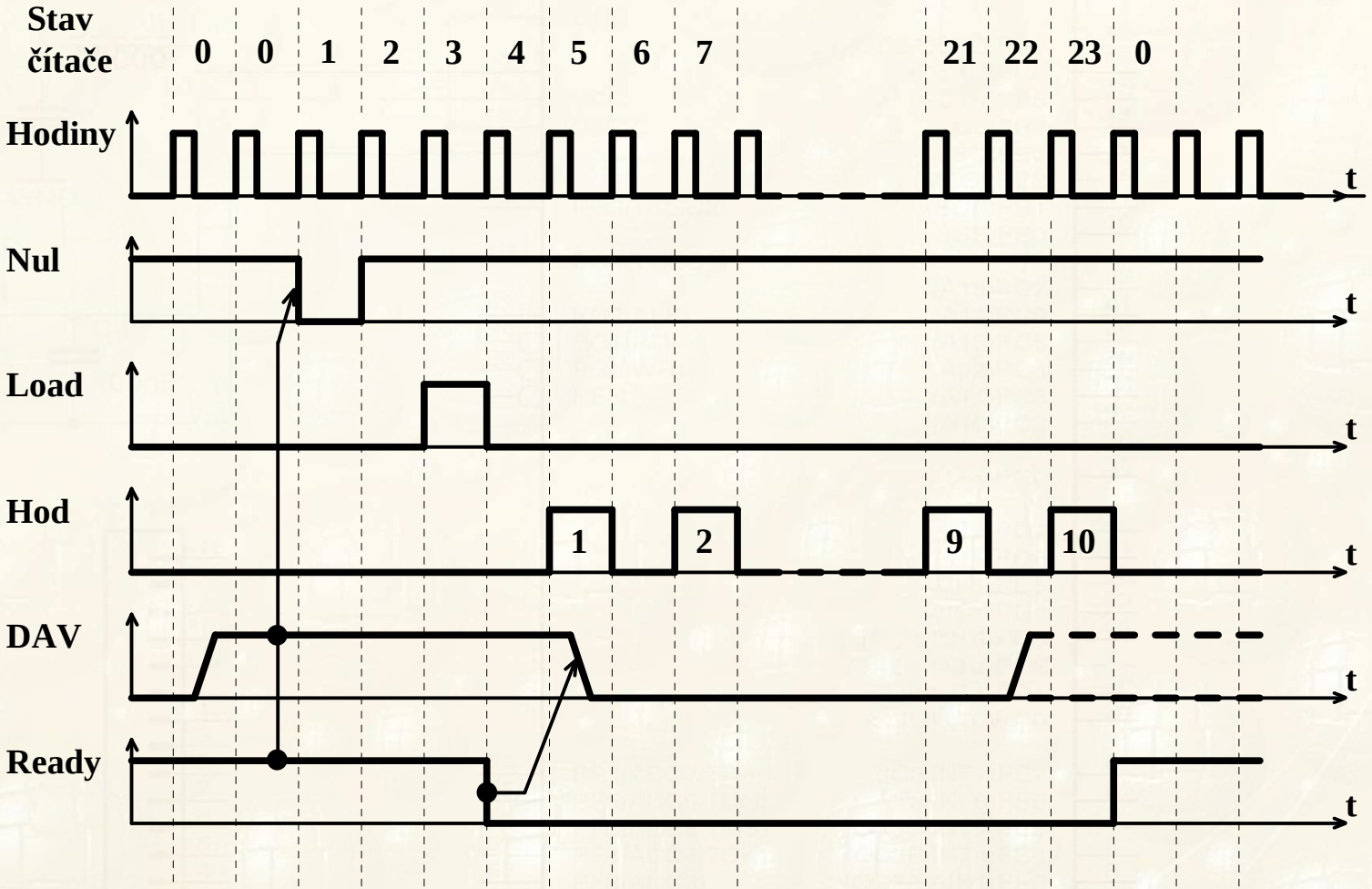
PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

Příklad: Navrhněte obvodový řadič pro sériovou sčítačku dvou desetibitových dvojkových čísel X a Y.
Řízenými prostředky je sériová sčítačka z obrázku. Řízení bude direktivní (do řadiče nebude přiveden žádný zpětný signál o stavu) a řídící signály musí mít takové časové parametry, které jsou pro řízené obvody vyhovující.



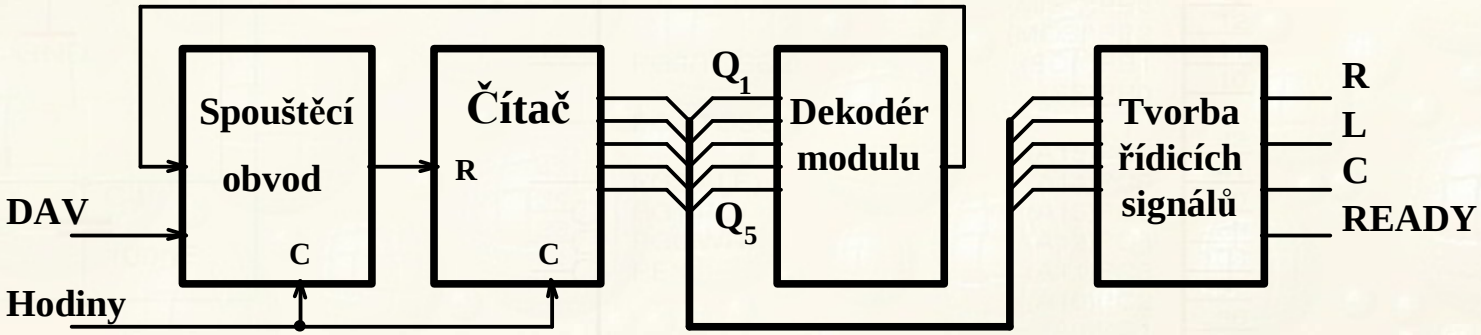
PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

Časové průběhy řídících signálů zajišťující správnou činnost sčítačky mohou vypadat dle obrázku. Jednotlivé fáze řadiče můžeme označit stavem řídícího čítače, jehož stav bude dekódován dekodérem stavu na potřebné signály.



PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

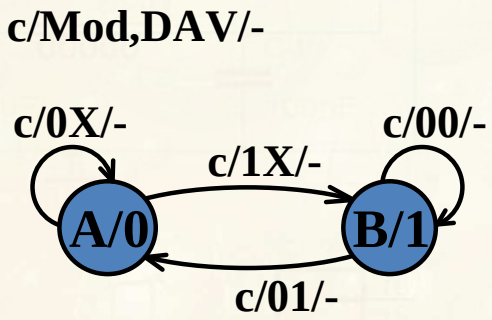
Z časového diagramu vyplývá, že po zápisu zpracovávaných hodnot je zrušen signál Ready (log.0), z kterého periferie zjistí převzetí hodnot sčítačkou. Po dokončení součtu je nastaven signál Ready=1 (připravenost dalšího zpracování). Jsou-li ve stejné době připraveny další hodnoty (Ready=DAV=1), pak jednotka pokračuje ve své činnosti. Pro (DAV=0) setrvává v počátečním stavu.



Řídicí signály (Nul, Load, Hod a Ready) budeme generovat dekódovacími obvody stavu čítače. Oproti obecnému zapojení použijeme jenom obvod pro vyhodnocení čítání a skoku. Nepředpokládáme zpětnou informaci ⇒ jedná se o řadič s **direktivním řízením**. Čítač bude mít modul 24, po dosažení hodnoty (23) je synchronně nastaven do počátečního stavu (0) spouštěcím obvodem.

PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

Spouštěcí obvod bude synchronizován ⇒ musí dekodér pro zkrácení cyklu indikovat hodnotu modulu, tj.$23_{10}=10111_2$. Na obrázku je vývojový diagram spouštěcího a nulovacího obvodu, kde Mod=1 indikuje poslední stav čítače Q4Q3Q2Q1Q0 = 10111. Stav **A** odpovídá realizovanému sčítání, stav **B** odpovídá nulování čítače a čekání na nová data (vnitřní proměnná=1). Stavová tabulka popisuje chování spouštěcího obvodu, druhá tabulka stav vnitřní proměnné.



z^i	Mod,DAV			
	00	01	11	10
A	A	A	B	B
B	B	A	---	---

z^i	Mod,DAV			
	00	01	11	10
0	0	0	1	1
1	1	0	---	---

$$z^{i+1} = Mod + \overline{DAV}.z^i$$

$$Nul = \overline{Q}_1 + Q_2 + Q_3 + Q_4 + Q_5 \qquad Load = Q_1.Q_2.\overline{Q}_3.\overline{Q}_4.\overline{Q}$$

$$Ready = \overline{Q}_3.\overline{Q}_4.\overline{Q}_5 \qquad Hod = Q_1.Q_3 + Q_1.Q_4 + Q_1.Q_5$$

PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

Častou chybou je připojení výstupu dekodéru přímo na synchronní čítač. Existují malé **nepředvídatelné rozdíly**, které mohou být rozdílnými cestami dekodérem znásobeny, a ty mohou způsobit vznik parazitních impulzů. Jejich odstranění je možné pomocí vyrovnávacího registru, který je aktivován v dostatečném odstupu od hodinových impulzů, kdy je stav čítače stabilní.

Z navrženého časování vyplývá, že nulovací a nastavovací impulz bude trvat jeden hodinový impulz a perioda generovaných hodinových impulzů C bude dva hodinové impulzy. Tyto hodnoty by měly být porovnány s časovými parametry sčítačky, aby byla zajištěna její správná činnost při direktivním řízení řadičem. Hodinové impulzy **Hod** generované řadičem mohou být realizovány **hradlováním** hodinového kmitočtu **Hodiny** pro dosažení vyššího kmitočtu Hod. Toto řešení má svá úskalí v zúženi signálu Hodiny v úrovni log.1 (**riskantní**).

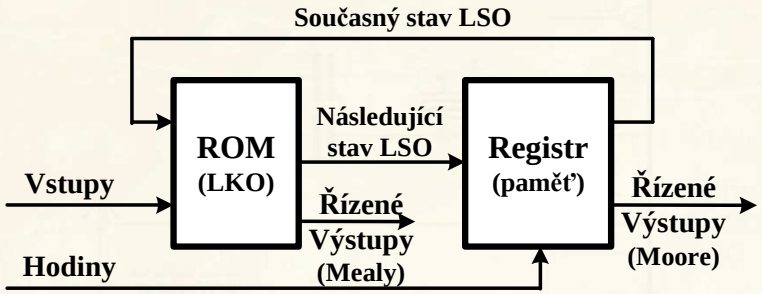
PŘECHOD OD OBVODOVÉHO ŘADIČE K MIKROPROGRAMOVÉMU

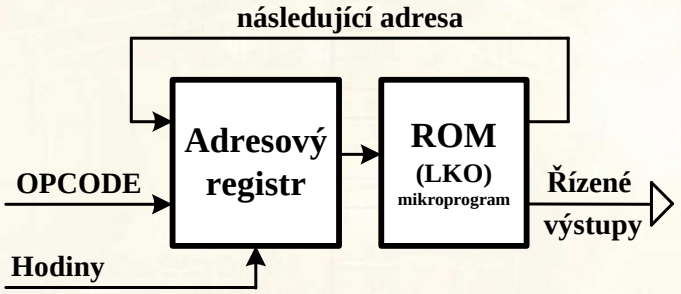
Mikroprogramování logická návrhová technika (M.V.Wilkes, 1950) ⇒ přináší řadu výhod oproti řešením Boolovskými rovnicemi a stavovými diagramy.

Od **programování** se liší jen **velmi málo** (obdobný formát instrukcí i programových struktur).

- **Mikroprogramování** - podporuje návrh obvodového řešení (hardware), s každou instrukcí může být generován nový stav řídících signálů. Dnes využíváno k realizaci jednotlivých instrukcí procesoru. Dříve mikroprogramovatelné řadiče využívány pro ovládání složitých sběrnic.
- **Programování** - JSA (jazyce symbolických adres, assembler), jazyk C, atd. je spojováno s manipulací s abstraktními daty a **řídící signály jsou generovány až po vykonání několika instrukcí**.

PŘECHOD OD OBVODOVÉHO ŘADIČE K MIKROPROGRAMOVÉMU





Obrácené pořadí registru a ROM ⇒ možnost jiného pohledu na LSO.

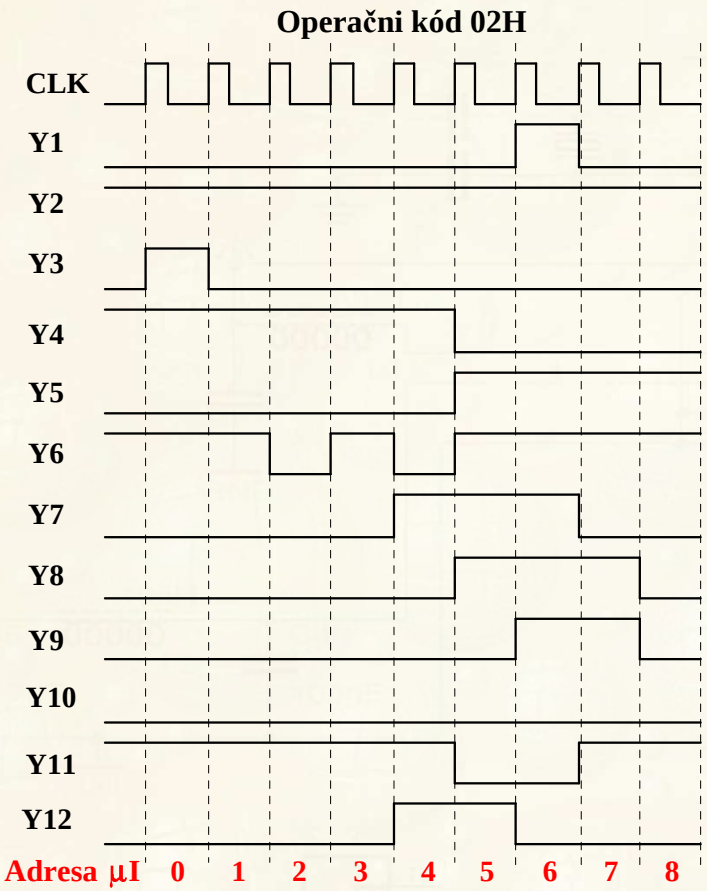
Zpětná vazba (vracející se vodiče) představuje

- vlevo - **současný stav LSO** (stav vnitřních proměnných)
- vpravo - **následující adresa** (adresa budoucího stavu a výstupních signálů).

Nahradíme-li vstupní proměnné **operačním kódem** ⇒ vhodný obvod pro realizaci řadiče (obvodu realizujícího instrukce μP).

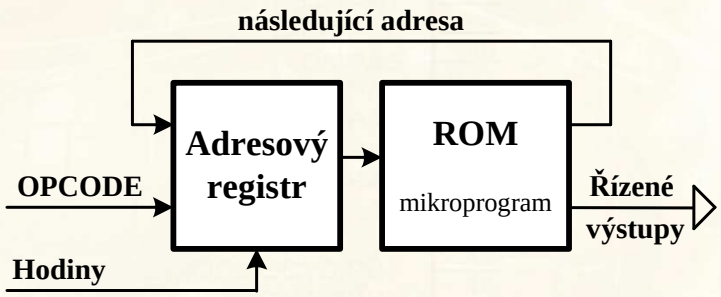
Část výstupů ROM tvoří generované signály, zbývající následující adresu mikroprogramu.

MIKROPROGRAMOVATELNÝ ŘADIČ



Adresa	Obsah ROM	Adresa	Obsah ROM
020h	42E1h	025h	8F26h
021h	42A2h	026h	1F37h
022h	40A3h	027h	5B28h
023h	42A4h	028h	4329h
024h	C4A5h		

Adresa mikroinstrukce
Adresa následující mikroinstrukce



Adresa ROM = Operační kód + následující adresa.

Výstup ROM = **Generovaná sekvence** (mikroinstrukce) + následující adresa mikroprogramu.

Pro obrázek vlevo:

❖ Výstupy ROM = Q15÷Q4 signály Y12÷Y1, Q3÷Q0 následující adresa,.

❖ Adresa ROM = An÷A4 operační kód (zde 02h), A3÷A0 následující adresa.