Logické sekvenční obvody

Výstupy logického sekvenčního obvodu jsou závislé nejen na současném stavu vstupních proměnných $x_1,x_2,...,x_n$ (vstupního vektoru xi), ale i na předcházejících hodnotách a počátečním stavu po připojení napájení. Matematicky

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

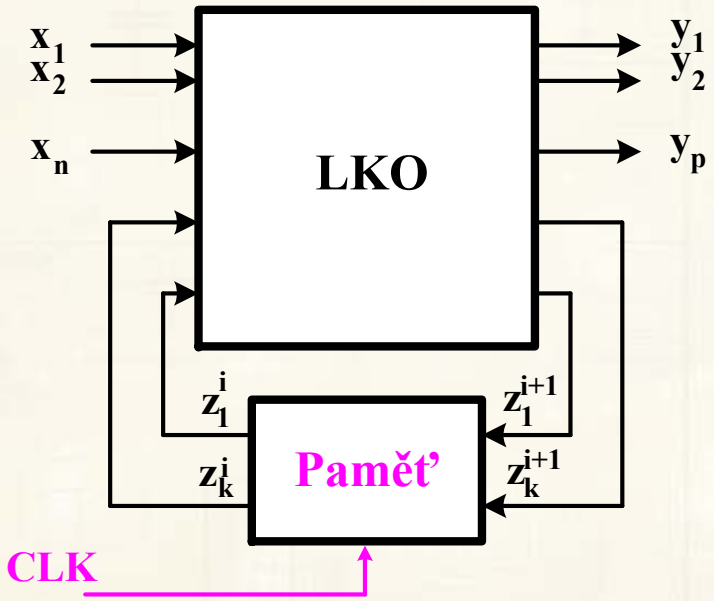
Analýza i návrh LSO je z uvedeném vztahu velmi problematický. Proto chování obvodu rozděluje do dvou částí tzv. rovnic – **přechodů a výstupů** zavedením **vnitřních proměnných z_k^i**.

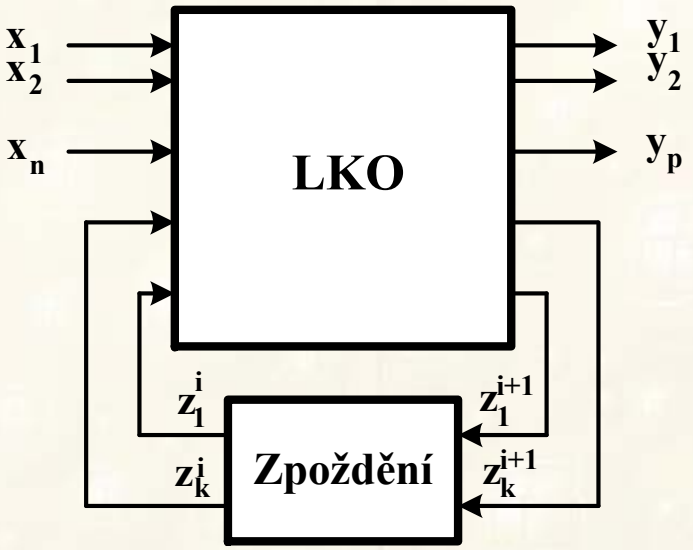
$$y_j^i = f_j\left(\vec{x}^i, \vec{z}^i\right) = f_j\left(x_1^i, x_2^i, ..., x_n^i, z_1^i, z_2^i, ..., z_m^i\right)$$
$$z_k^{i+1} = g_k\left(\vec{x}^i, \vec{z}^i\right) = g_k\left(x_1^i, x_2^i, ..., x_n^i, z_1^i, z_2^i, ..., z_m^i\right)$$

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 po připojení k napájení.

Obecně se LSO skládá ze dvou bloků - Paměti a LKO. Paměť LSO může být realizována

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





Synchronní obvod – ke změnám z_k^i dochází téměř současně.

Asynchronní obvod - ke změnám z_k^i dochází postupně v závislosti na tom, jak se šíří podnět obvodem.

Podle funkcí výstupů můžeme LSO dělit:

- **Mealyho typ** $y_j^i = f_j\left(\vec{x}^i, \vec{z}^i\right) = f_j\left(x_1^i, x_2^i, \ldots, x_n^i, z_1^i, z_2^i, \ldots, z_m^i\right)$
- **Mooreův typ** $y_j^i = f_j\left(\vec{x}^i, \vec{z}^i\right) = f_j\left(z_1^i, z_2^i, \ldots, z_m^i\right)$

Paměť **synchronních obvodů** je tvořena synchronizovanými klopnými obvody. Změny v obvodu souvisí s výkonnou hranou hodinového signálu CLK. Interval mezi hranami CLK se volí tak, aby v obvodu odezněly všechny přechodné jevy. Kombinační část obvodu nemusí být bezhazardní, přípustné jsou **současné změny vstupních i vnitřních proměnných**. Změny vstupních proměnných **nesmí způsobit nedodržení parametrů pro činnost PČ** (metastabilní stav) .

U **asynchronního** obvodu musí být kombinační část **bezhazardní**, nežádoucí je současná změna dvou vstupních i vnitřních proměnných (možný vznik kritického souběhu = obvod skončí v jiném stavu, než bylo předepsáno).

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 – ♣ kombinace vnitřních proměnných (analýza a konečná fáze návrhu)

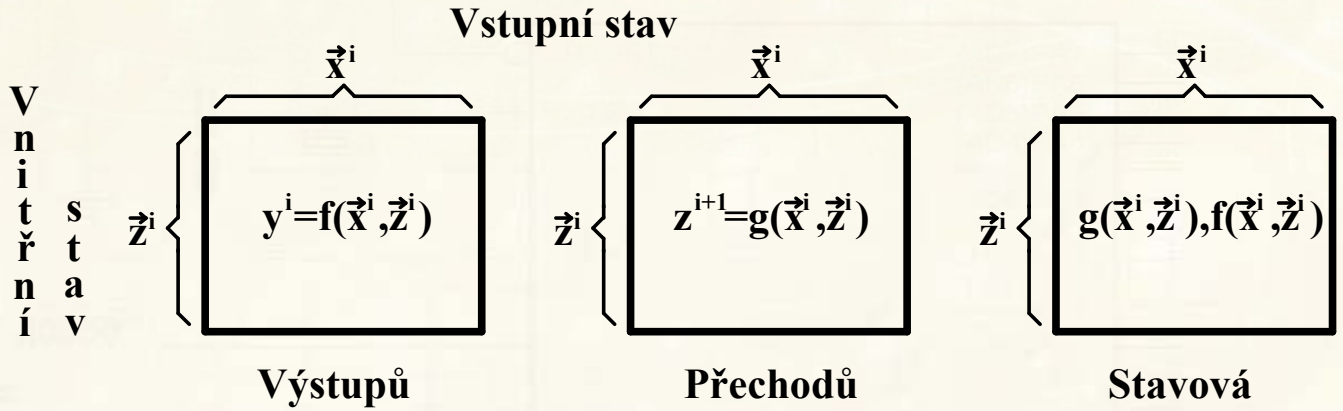
♣ obecným symbolem (při návrhu obvodu).

Stabilní stav asynchronního obvodu - stav, v němž obvod setrvává neomezenou dobu při konstantním vstupu. Označuje se kroužkem a matematicky jej můžeme vyjádřit rovnicemi

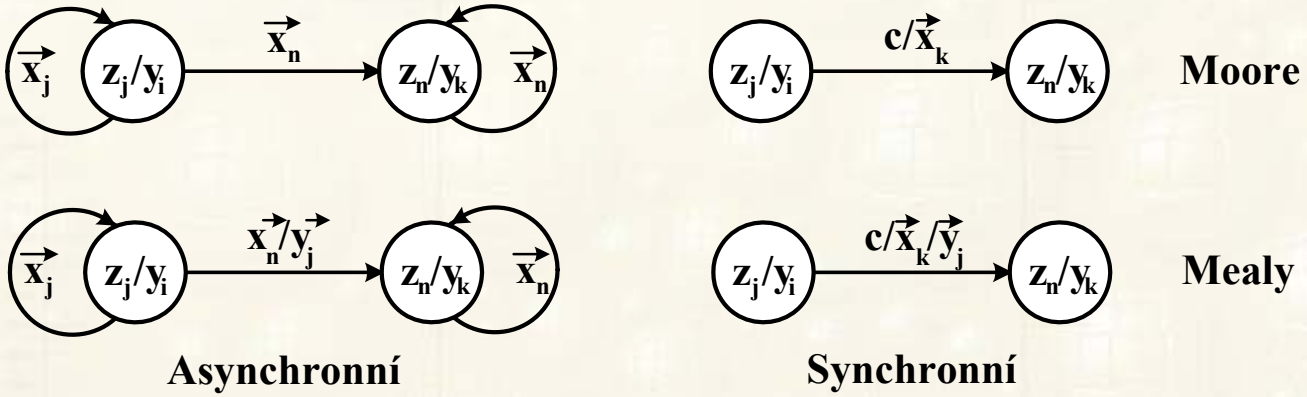
$$\vec{x}^{i+1} = \vec{x}^{i} \qquad \vec{z}^{i+1} = \vec{z}^{i}$$

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.

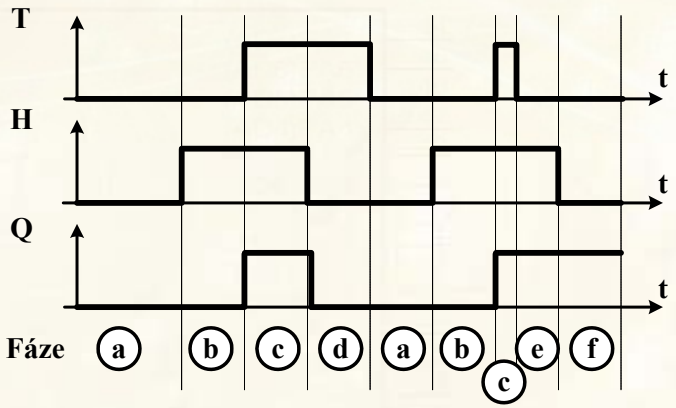


- **Strukturou** – Schéma je výchozím vyjádřením obvodu před jeho analýzou a cílovým stavem při syntéze sekvenčního obvodu.
- **Grafem** – možný způsob zadání chování LSO nebo poslední fáze při analýze LSO.



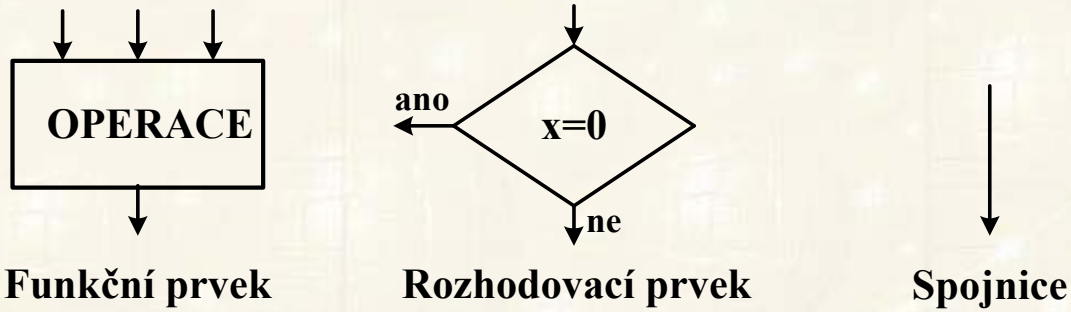
- **Časovým nebo fázovým diagramem** – vstupních a výstupních signálů obvodu viz následující obrázek.

$x_1 x_2$ 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



- **Fázová tabulka** – návrh i analýza asynchronního SO. V každém řádku je jeden stabilní stav.

- **Jazykem HDL** – vhodný pro návrh složitých LKO i LSO. Firemní jazyky - AHDL, LHDL, ABEL, atd. počátky vývoje PLD – **nepřenosné**. Univerzální HDL - **Verilog** a **VHDL.**
- **Vývojovým (programovým) diagramem**

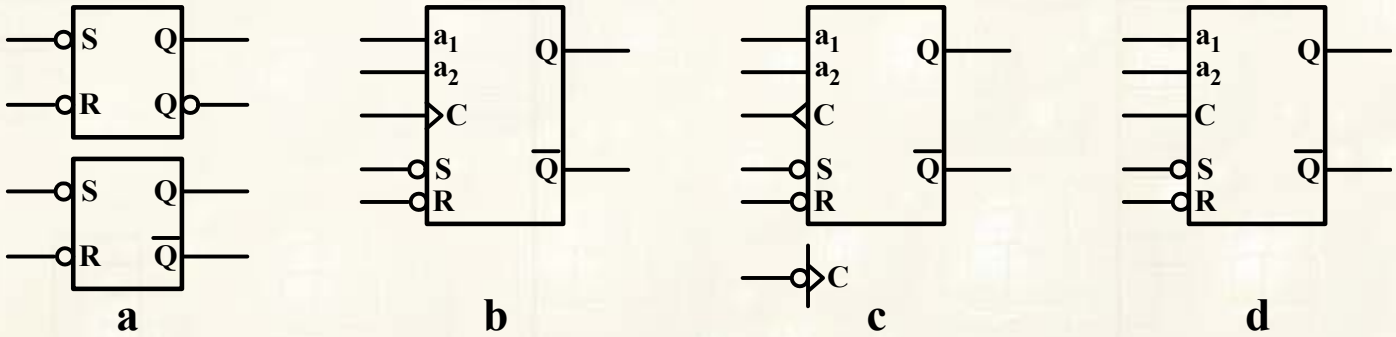


PAMĚŤOVÉ ČLENY

Paměťové členy (klopné obvody) - asynchronní LSO, s dvoustavovým výstupem. Slouží 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



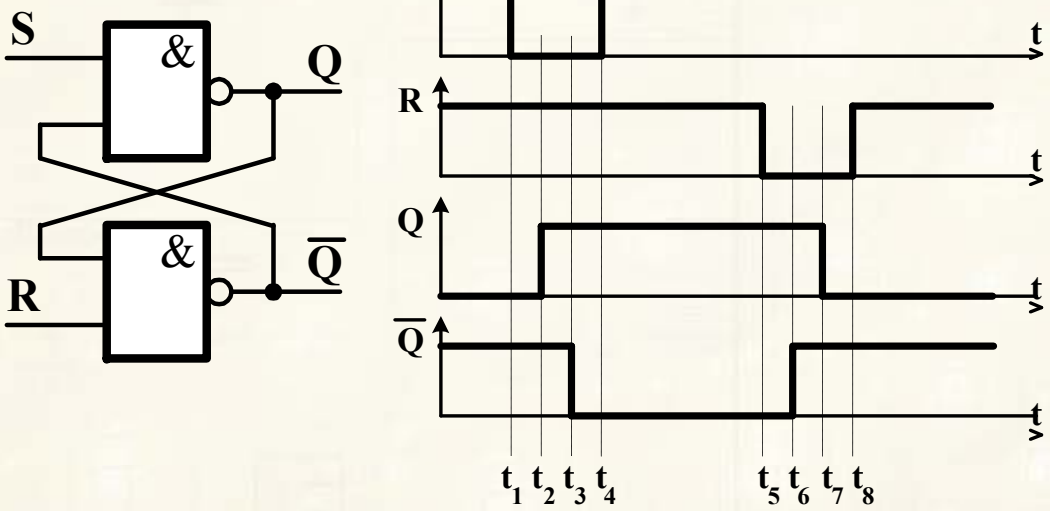
Schematické značení jednotlivých typů paměťových členů

a) asynchronní obvod s aktivní úrovní v log.0
b) obvod synchronizovaný náběžnou hranou se vstupy (R,S)
c) obvod synchronizovaný sestupnou hranou
d) obvod synchronizovaný úrovní hodinového signálu.

Paměťový člen RS

Asynchronně řízený obvod se vstupy R (reset) a S (set).

- RS-NAND jsou vstupy S a R aktivní v log.0.
- RS-NOR jsou vstupy S a R aktivní v log.1.

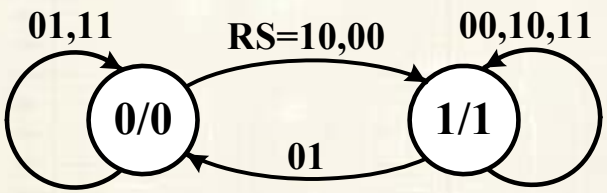


Z průběhů je zřejmé, že minimální šířka signálu musí být větší, než doba

$$\Delta t = t_3 - t_1 \approx 2.t_{pd},$$

kde t_{pd} je střední doba zpoždění signálu logického členu.

Zobrazení stavového diagramu a stavové tabulky s běžnými přechody obvodu.



Q^i \ R^iS^i	00	10	11	01
0	1	1	0	0
1	1	1	1	0

Q^{i+1}

PAMĚŤOVÝ ČLEN – RS

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

Pravdivostní tabulka

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

Tabulka přechodů nezbytná pro návrh LSO s RS.

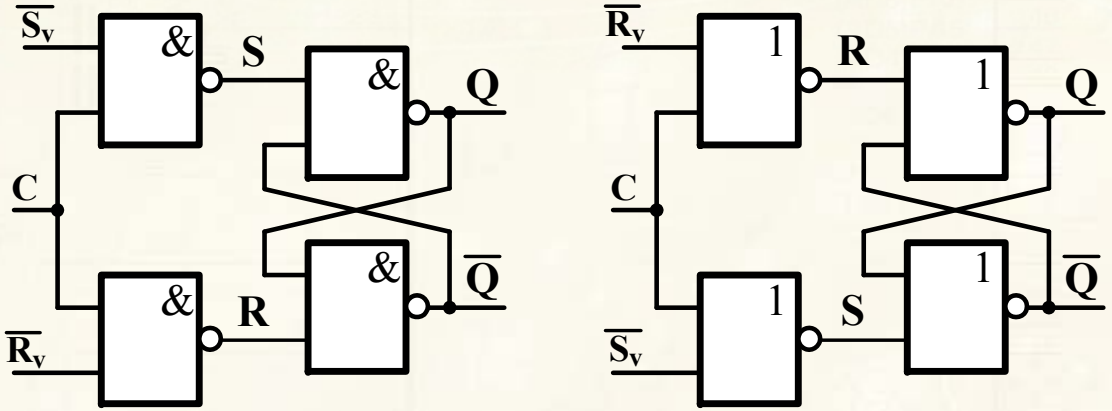
Operátor paměťového členu RS NAND

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

Stav R=S=0 se označuje jako zakázaný (nepřípustný)

- Výstupy obvodu nejsou vzájemně komplementární.
- Při současné změně vstupů R a S z 0 →1 není jisté, zda výstup Q = 0 nebo Q=1.

Paměťový člen RST



Obvod je synchronizován úrovní vstupní proměnné C (Clock) nebo H. U varianty NAND při C=0 je výstup Q nezávislý na vstupních hodnotách R_v a S_v. Pro C=1 se chová jako paměťový člen RS.

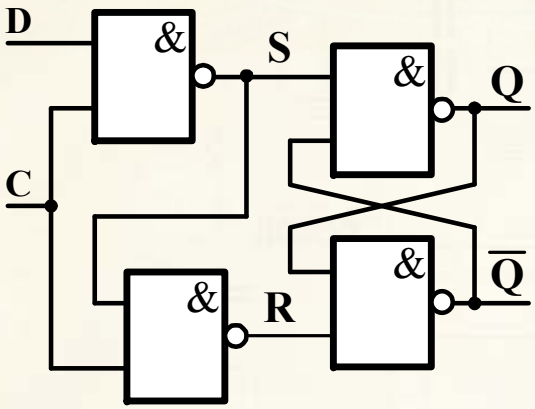
$$Q^{i+1} = C^i.\overline{S}_v^i + \left(R_v^i + \overline{C}^i\right).Q^i \quad \text{pro} \quad \text{RST} - \text{NAND}$$

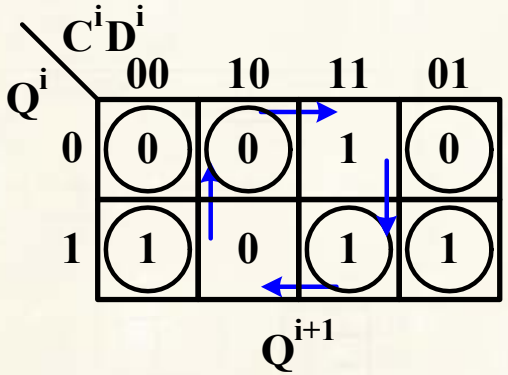
$$Q^{i+1} = \left(\overline{C}^i + \overline{R}_v^i\right).\left(S_v^i.C^i + Q^i\right) \quad \text{pro} \quad \text{RST} - \text{NOR}$$

Proměnné R_v a S_v by měly konstantní, je-li obvod synchronizován (C=1). Při sériovém řazení členů tohoto typu se dá splnit

- Jednofázová impulzní synchronizace - šířka C ovšem kritická
- Častěji se proto používá dvou a vícefázové řízení.

Paměťový člen D - Delay





Základ PČ D tvoří opět paměťový člen RS.

- Pro C=0 jsou na vstupech R a S neaktivní úrovně (log.1). Výstup uchovává poslední stav.
- Pro C=1 je na vstupu R hodnota D, a na S její negace.
- Výstup Q kopíruje vstupní proměnnou D.

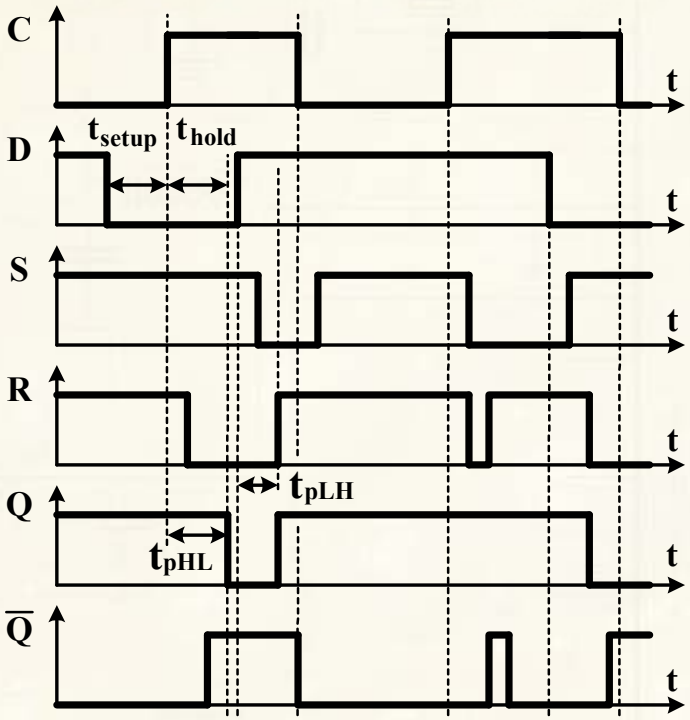
Operátor paměťového členu D je dán rovnicí

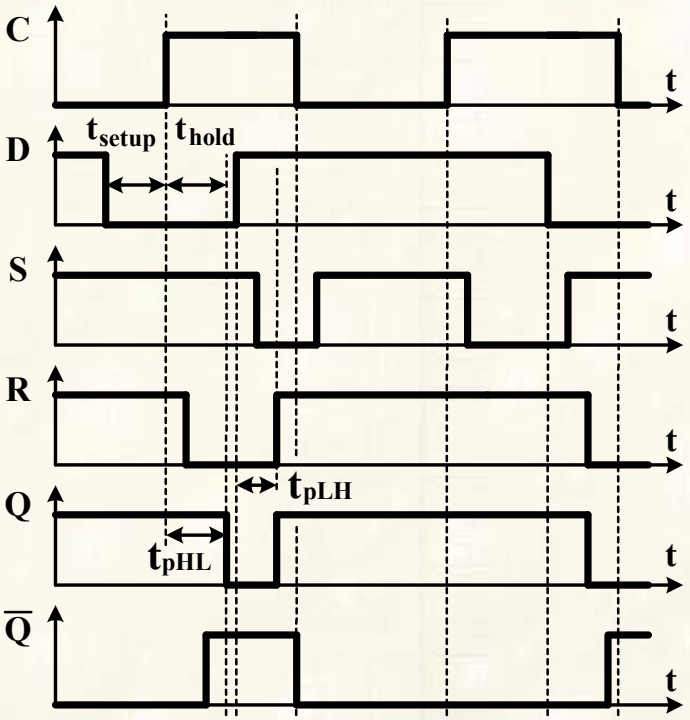
$$Q^{i+1} = \overline{C}^i.Q^i + C^i.D^i$$

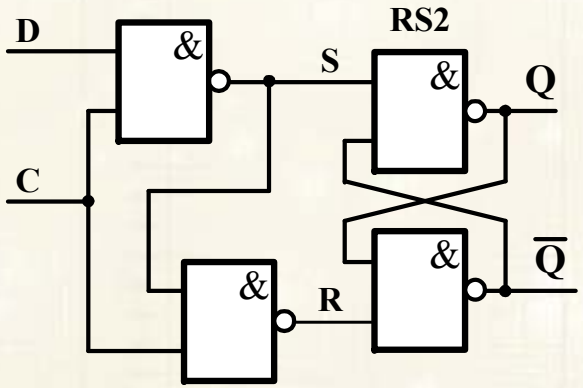
Clock	Data	Vstupy RS	Q^{i+1}	$nonQ^{i+1}$	Poznámka
0	0 or 1	1 1	Q^i	$nonQ^i$	žádná změna
1	0	0 1	0	1	Nulování Q
1	1	1 0	1	0	Nastavení Q

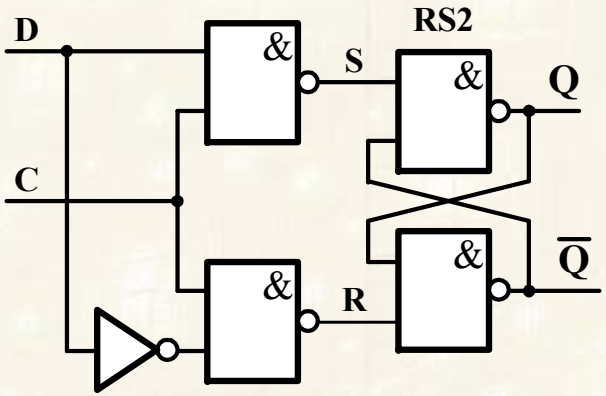
PAMĚŤOVÝ ČLEN – D (LATCH)

Rozdíly mezi realizacemi PČ D při uvažování zpoždění log.členů. Varianta s pěti hradly má lepší chování při změně D při C=1.



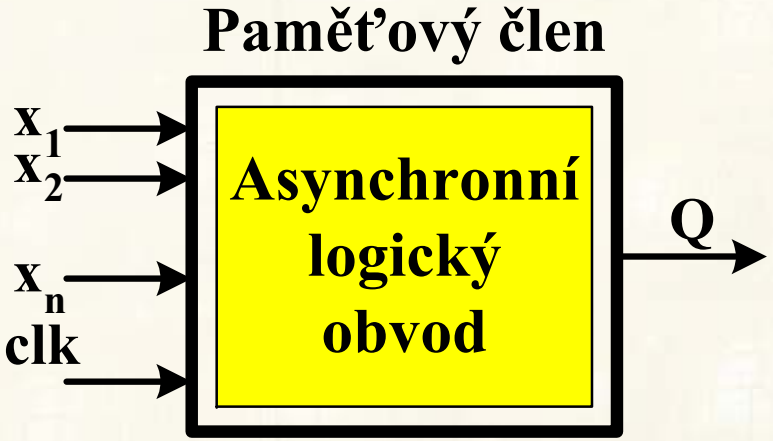






Synchronizované PČ hranami hodinového signálu

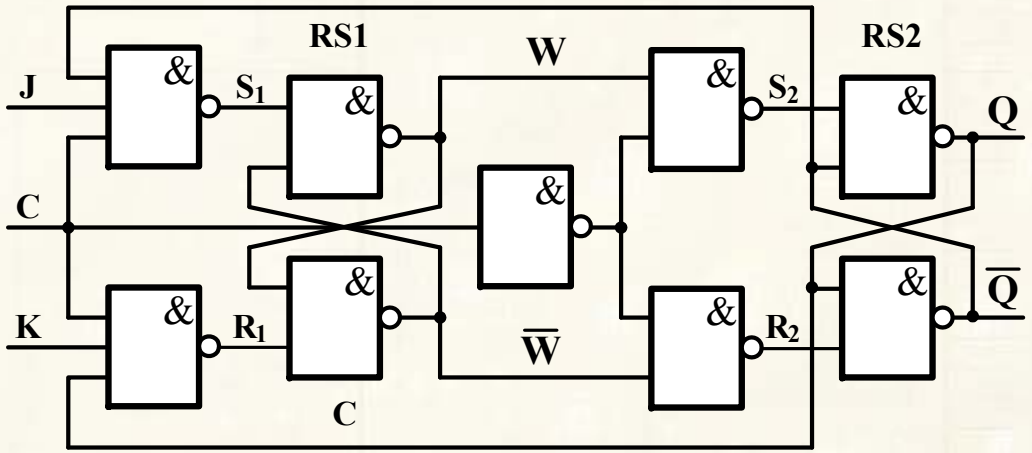
Hranově synchronizované PČ jsou realizovány asynchronními sekvenčními obvody, navrženými pro dané chování. Výstup obvodu ovlivňuje stav vstupních signálů v okamžiku výkonné hrany synchronizačního (hodinového) signálu. Často jsou tvořeny několika paměťovými členy (RS) mezi nimiž je přenos signálu řízen alespoň dvoufázově.



Protože se u asynchronních LSO zakazuje současná změna více vstupních proměnných, nalezeme v katalogu časové parametry pro zajištění bezchybné činnosti PČ. Jejich nedodržení vede k hazardním stavům v LSO nebo k metastabilnímu chování PČ.

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. Realizace starší varianty tohoto obvodu je zobrazena.

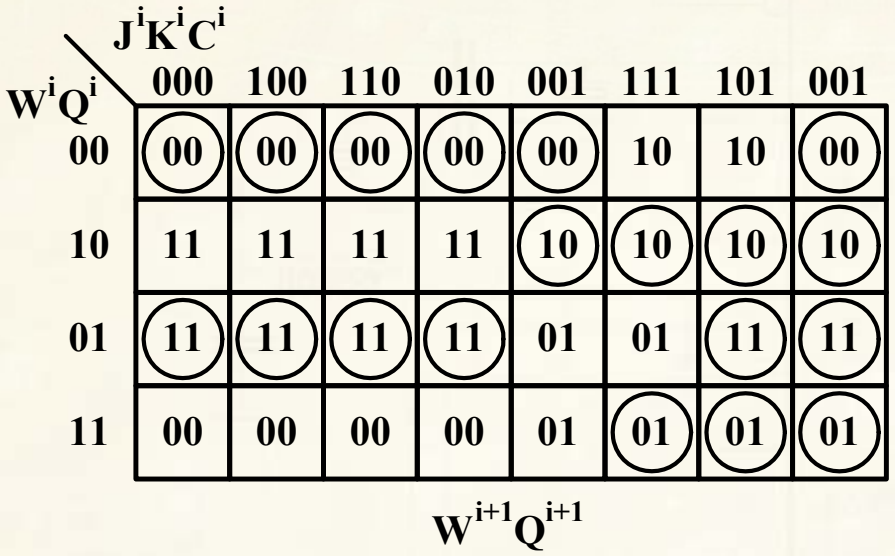


Při analýze JK ztotožníme výstupy RS s vnitřními proměnnými LSO. Obvod má vstupní proměnné J,K,C, výstupní proměnnou Q a vnitřní proměnné W a Q. Pro funkce buzení vstupů paměťových členů můžeme psát

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

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

PAMĚŤOVÝ ČLEN – JK (MASTER-SLAVE)



Dosazením vztahů pro S_j a R_j do operátoru paměťového členu RS získáme tyto funkce přechodů pro vnitřní proměnné W a Q

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

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

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

Z rovnic přechodů snadno vytvoříme stavovou tabulku obvodu, v které zakroužkujeme stabilní stavy. Následně pro změny vstupních proměnných určíme chování obvodu. Pro obvyklou činnost se 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.

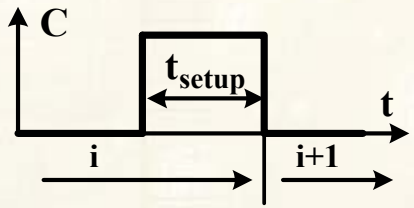
PAMĚŤOVÝ ČLEN – JK (MASTER-SLAVE)

Činnost obvodu se popisuje pravdivostní tabulkou, jeho operátorem nebo 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 je dnes k dispozici v provedení

- Master-Slave
- Ve variantě čistě hranově řízeného obvodu (doba předstihu nesouvisí s šířkou hodinového impulzu)
- Můžeme si jeho chování popsat ve Verilogu nebo VHDL

Paměťový člen D hranově řízený

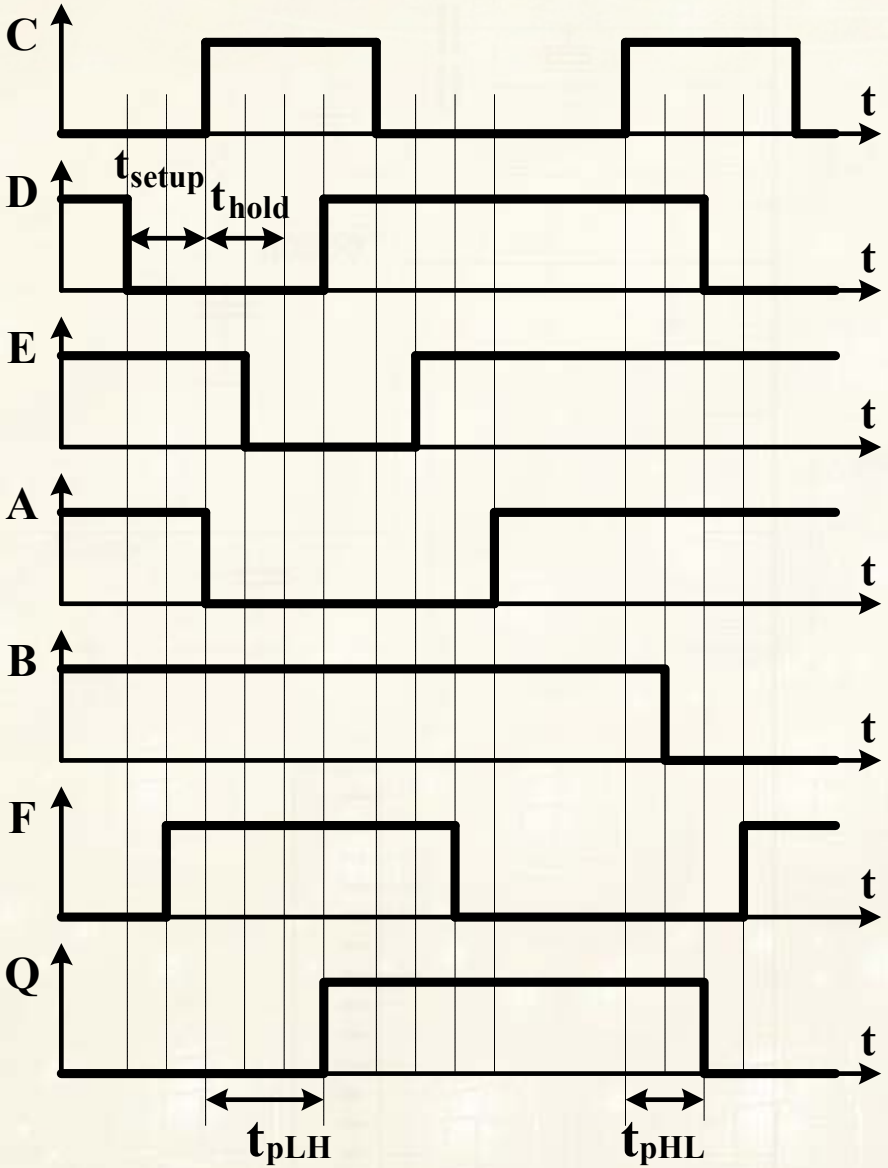
U hranově řízeného PČ D se využívá zpětná vazba, která brání průchodu vstupního signálu D strukturou obvodu bezprostředně po náběžné hraně synchronizačního signálu C. Na následujícím obrázku je PČ D (bez asynchronních vstupů R a S) spolu s časovými průběhy vstupních, vnitřních a výstupních signálů.

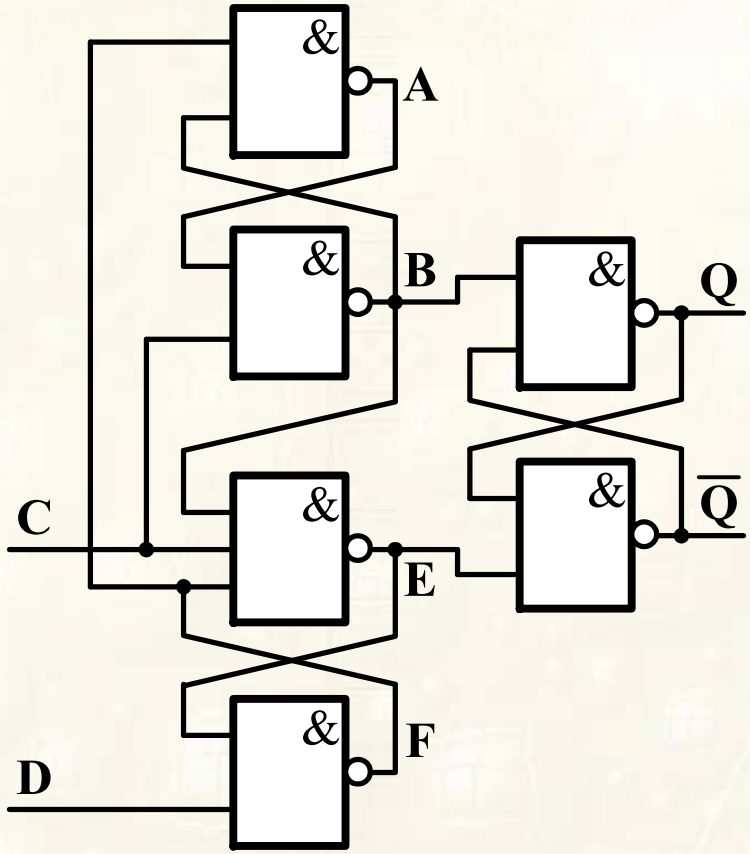
- Pro D=0 je přenos případných změn signálu D blokován v logickém členu signálem E=0.
- Při D=1 se blokování realizuje v logickém členu signálem B=0.

Z časových zpoždění v jednotlivých částech obvodu vyplývá, jak se změna na D šíří obvodem až k výstupu Q. Při malé strmosti náběžné hrany C se zpoždění prodlužují.

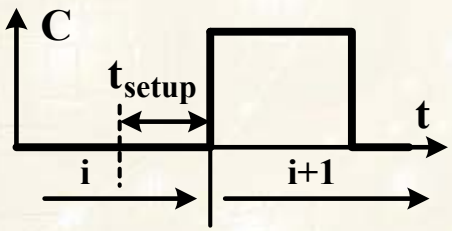
Z činnosti PČ D vyplývá, že v okolí aktivní (náběžné) hrany synchronizačního signálu musí být vstupní signál D konstantní. Doby **předstihu** a **přesahu** signálu D vůči náběžné hraně C jsou znázorněny. Jejich nedodržení způsobuje občas u PČ **metastabilní stav**.

PAMĚŤOVÝ ČLEN – D HRANOVĚ ŘÍZENÝ





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



PAMĚŤOVÝ ČLEN – D HRANOVĚ ŘÍZENÝ

Některé paměťové členy jsou vybaveny 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 nebo na vstupu D nebo JK viz. 4 a 5 řádek v tabulce.

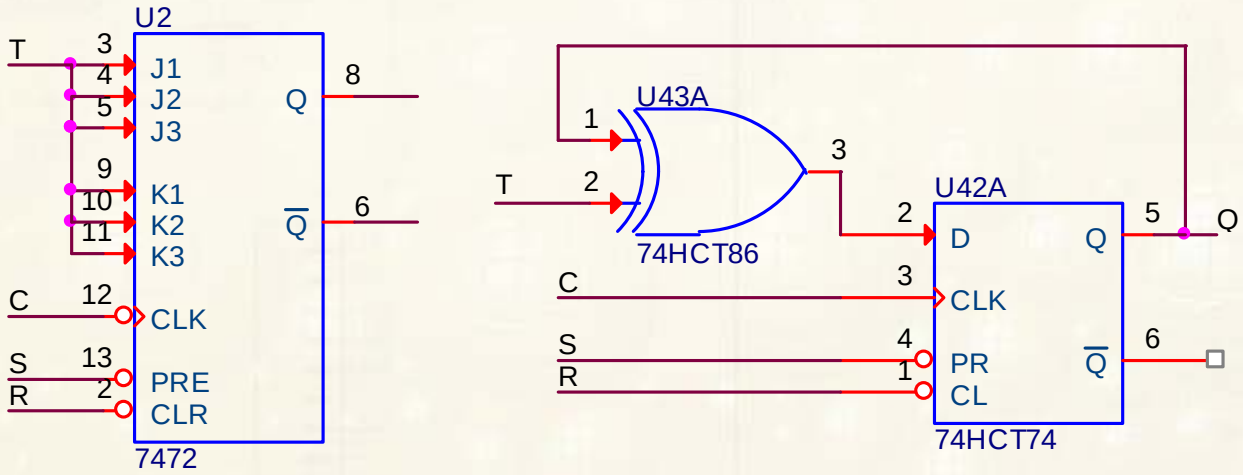
C	D	Vstupy RS	Q^{i+1}	$nonQ^{i+1}$	Poznámka
↓	0 nebo 1	1 1	Q^i	$nonQ^i$	Žádná změna výstupu Q
↑	0	1 1	0	1	Synchronní nulování výstupu Q
↑	1	1 1	1	0	Synchronní nastavení výstupu Q
↑	0 nebo 1	0 1	0	1	Asynchronní nulování výstupu Q
↑	0 nebo 1	1 0	1	0	Asynchronní nastavení výstupu Q

Paměťový člen T

Paměťový člen T (synchronní invertor) je obvod, který pro T=1 změní po výkonné hraně hodin hodnotu svého výstupu Q. Pro vstupní signál T=0 se hodnota výstupu nemění.

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

PČ T není běžně vyráběn, lze jej realizovat pomocí PČ JK (J=K=T) nebo PČ D (D=Q.T).

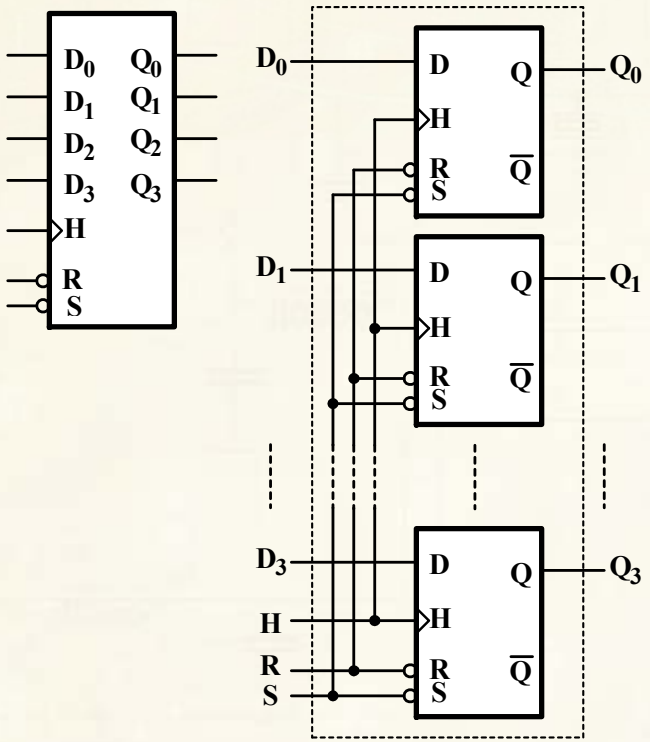


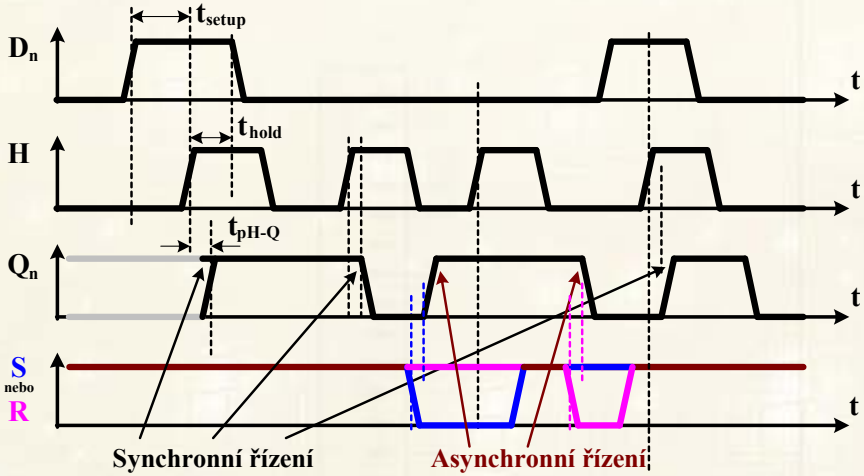
Paměťový člen E

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.

PARALELNÍ REGISTR – PAMĚŤ VĚTŠÍHO POČTU BITŮ

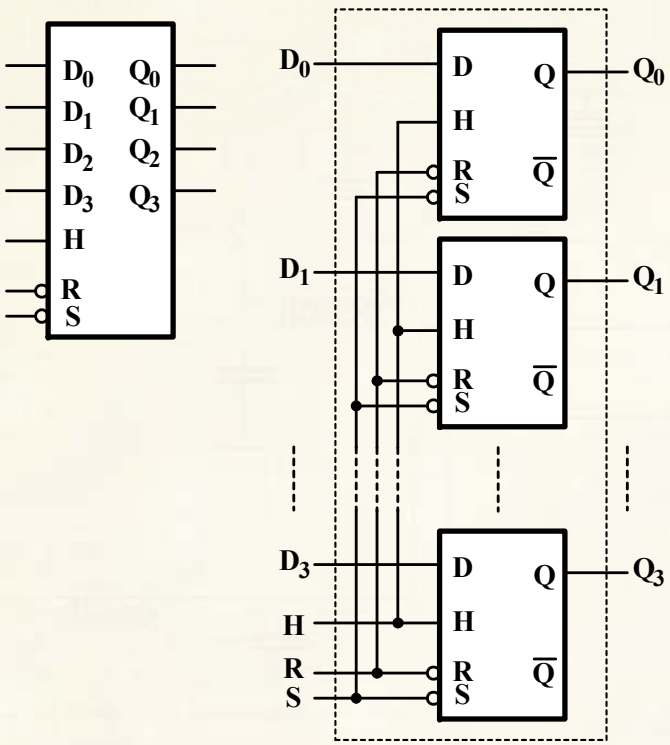


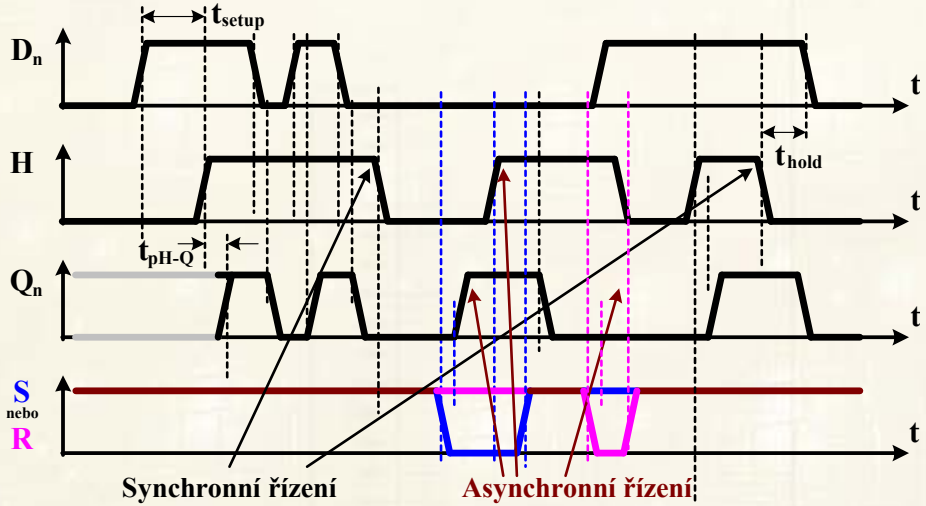


Registr – umožňuje uložení většího počtu bitů náběžnou nebo sestupnou hranou hodinového signálu. Nejdéle za čas $t_{pH \rightarrow Q}$ se přenese vstup D_n na výstup Q_n. Důležité je dodržet dobu předstihu t_{setup} a přesahu t_{hold} dat D_n před hranou CLK. Nesplnění předstihu může způsobit **metastabilní stav.** **Použití:**

- **Vyrovnávací paměť** - uchování krátce platné hodnoty
- **Pipeline** - zvýšení výkonu daného zařízení.
- **Výstupní brána** µP
- **Paměť** vnitřních proměnných LSO
- **Vstupní brána** µP má-li registr třístavový výstup.

LATCH– PAMĚŤ VĚTŠÍHO POČTU BITŮ



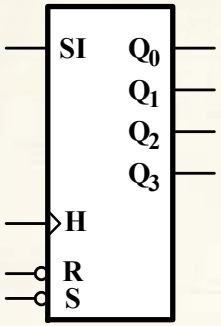


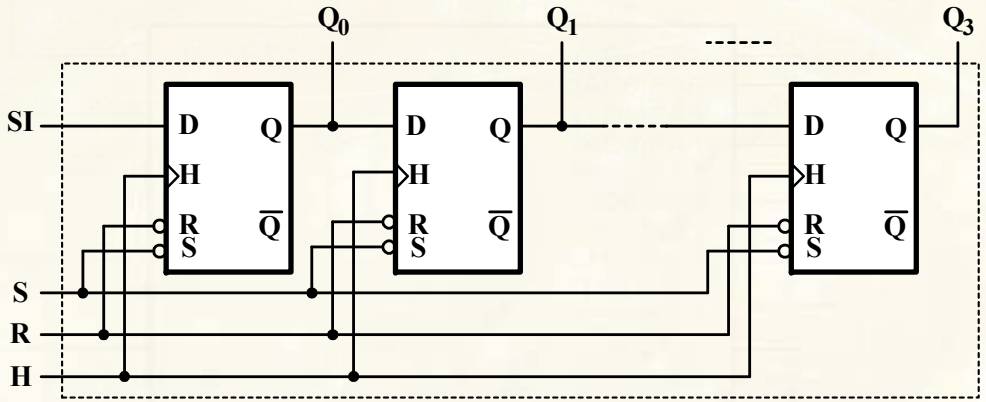
Latch – umožňuje uložení většího počtu bitů jedním hodinovým signálem. Jeho PČ D jsou řízeny úrovní H=1, při kterém výstupy Q_n kopírují hodnotu na vstupech D_n se zpožděním $\mathbf{t_{pH\rightarrow Q}}$. Důležité je dodržet dobu předstihu $\mathbf{t_{setup}}$ a přesahu $\mathbf{t_{hold}}$ dat D_n před hranou CLK. Nesplnění předstihu může způsobit **metastabilní stav.**

Použití:

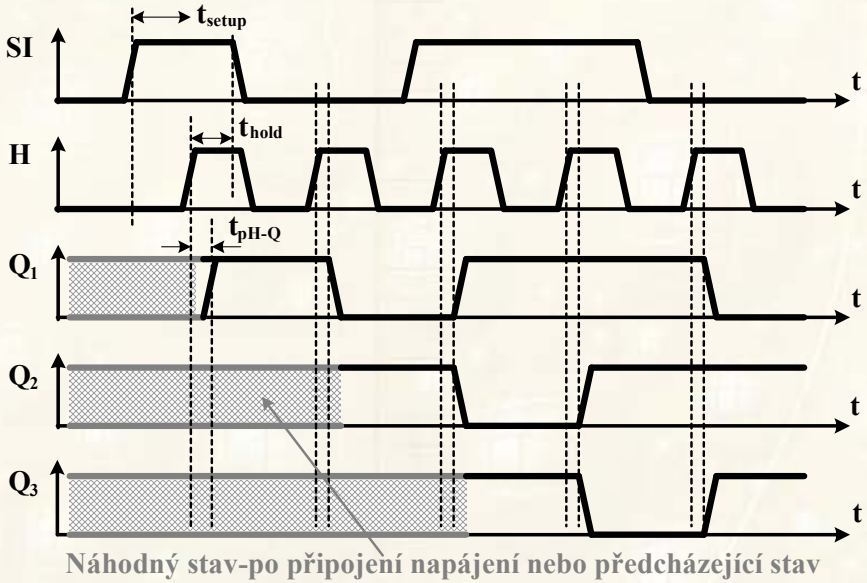
- **Vyrovnávací paměť** - uchování krátce platné hodnoty
- **Pipeline** – šířka impulzu H bude kritická.
- **Výstupní brána** µP
- **Vstupní brána** µP má-li registr třístavový výstup.

POSUVNÝ REGISTR





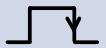
Posuvný registr slouží k vysílání nebo zachycení sériové posloupnosti bitů. Existuje ve variantě sériově-paralelní (obr.) nebo paralelně-sériový nebo ve spojení obou variant. Pro správnou činnost musí být splněny časy $t_{pH \rightarrow Q}$ a t_{setup}.



Použití:

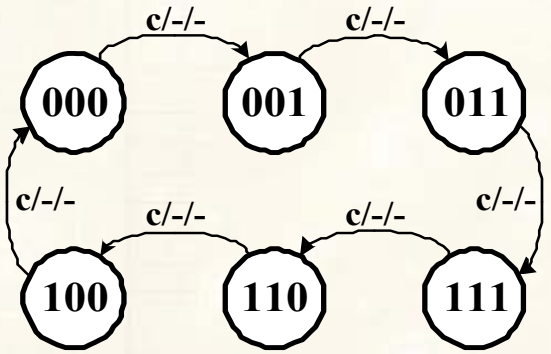
Generování a **příjem** sériových posloupností. Sériové **vstupní a výstupní brány** procesorového systému. **Paměť** vnitřních proměnných binárních generátorů.

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

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 s PČ D.



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

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

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

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

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	101
1	000	011	110	100

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

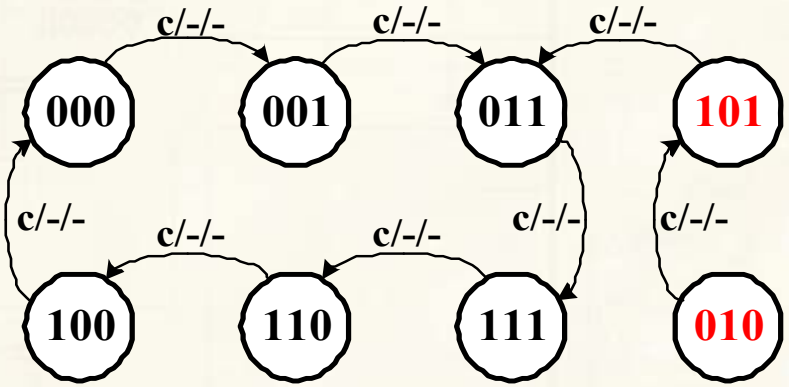
NÁVRH SYNCHRONNÍHO LSO

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

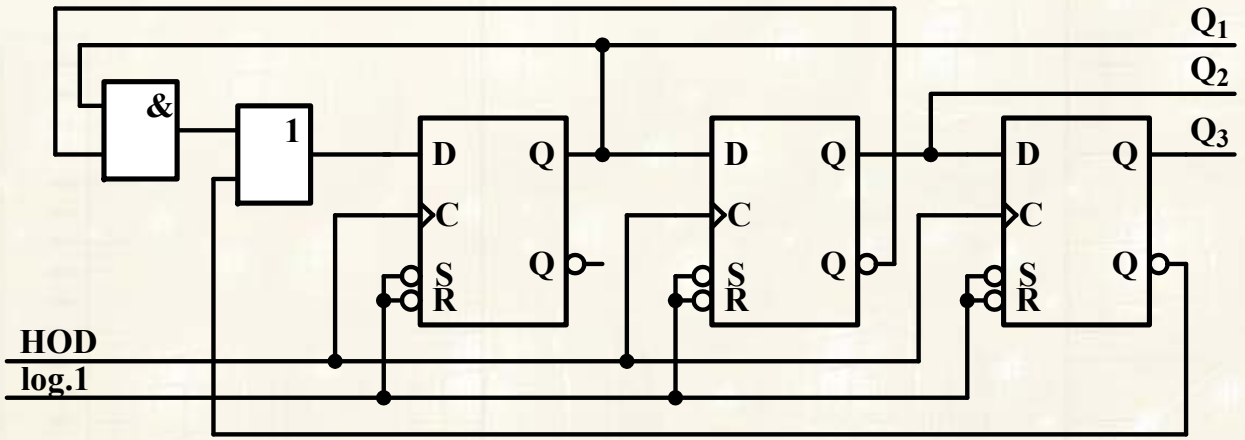
Q_3^i \ $Q_1^i Q_2^i$	00	10	11	01
0	001	011	111	101
1	000	011	110	100

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

5) Před vytvoření schématu ověříme přechody z nevyužitých stavů.



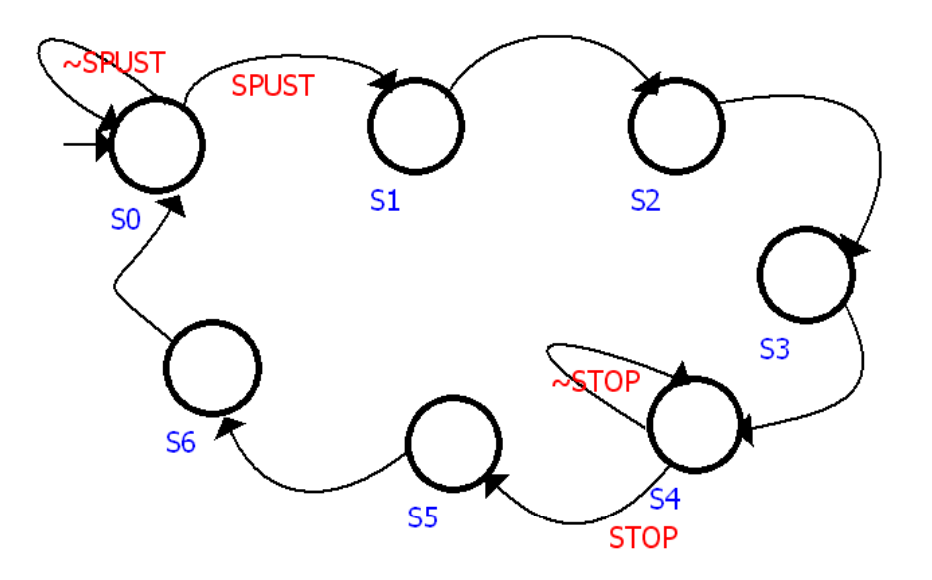
6) Nakreslíme navržené schéma obvodu



NÁVRH SYNCHRONNÍHO LSO

Návrh synchronního LSO velmi často začíná:

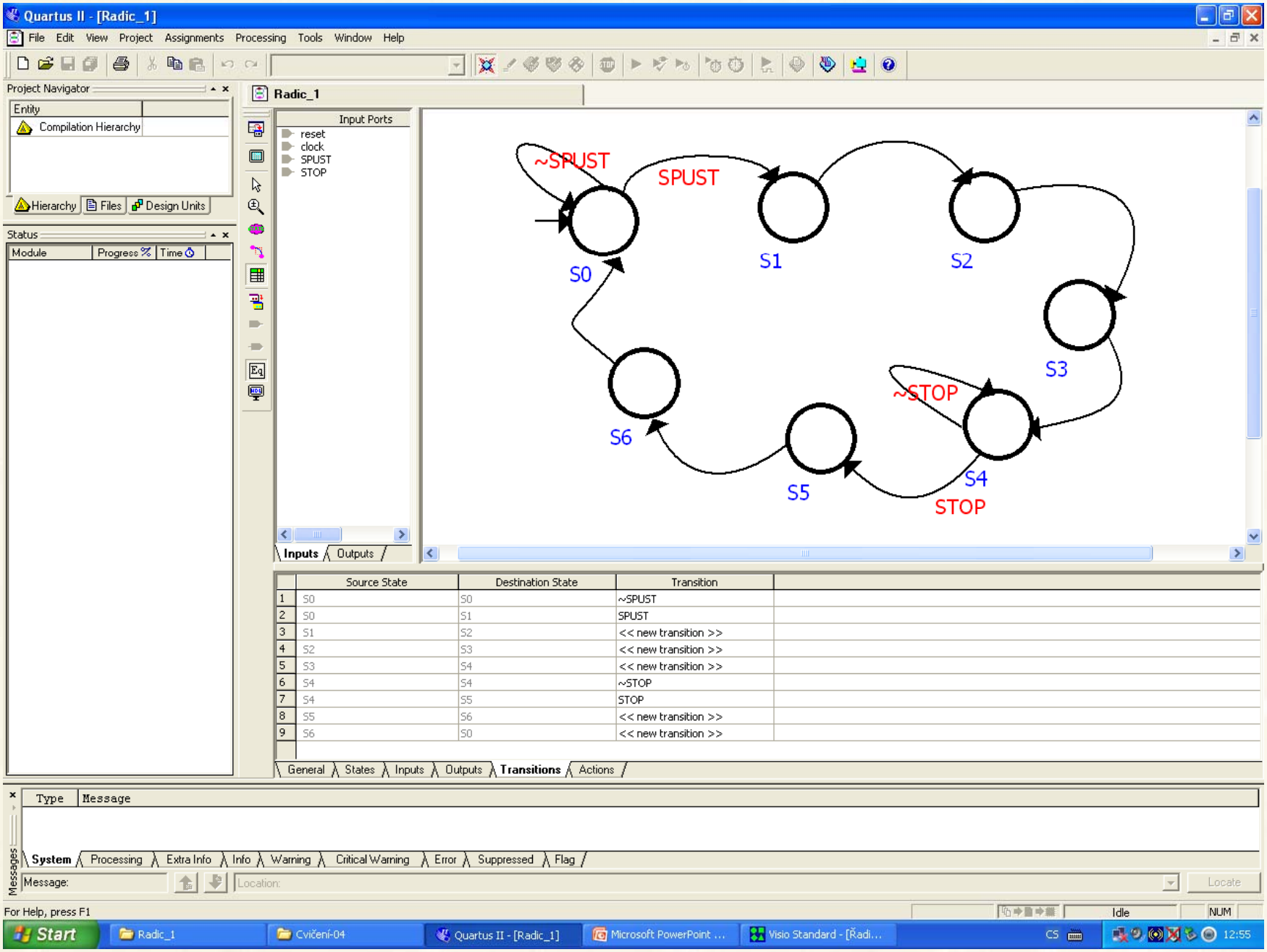
- Vytvořením vývojového diagramu
- Následně z něj vývojové tabulky
- Redukce vývojové tabulky (hledání ekvivalentních nebo slučitelných stavů). Výsledkem minimalizovaná vývojová tabulka.
- Kódování vnitřních stavů ⇒ Stavová tabulka
- Odvození funkcí přechodů a buzení vstupů PČ
- Nakreslení navrženého obvodu



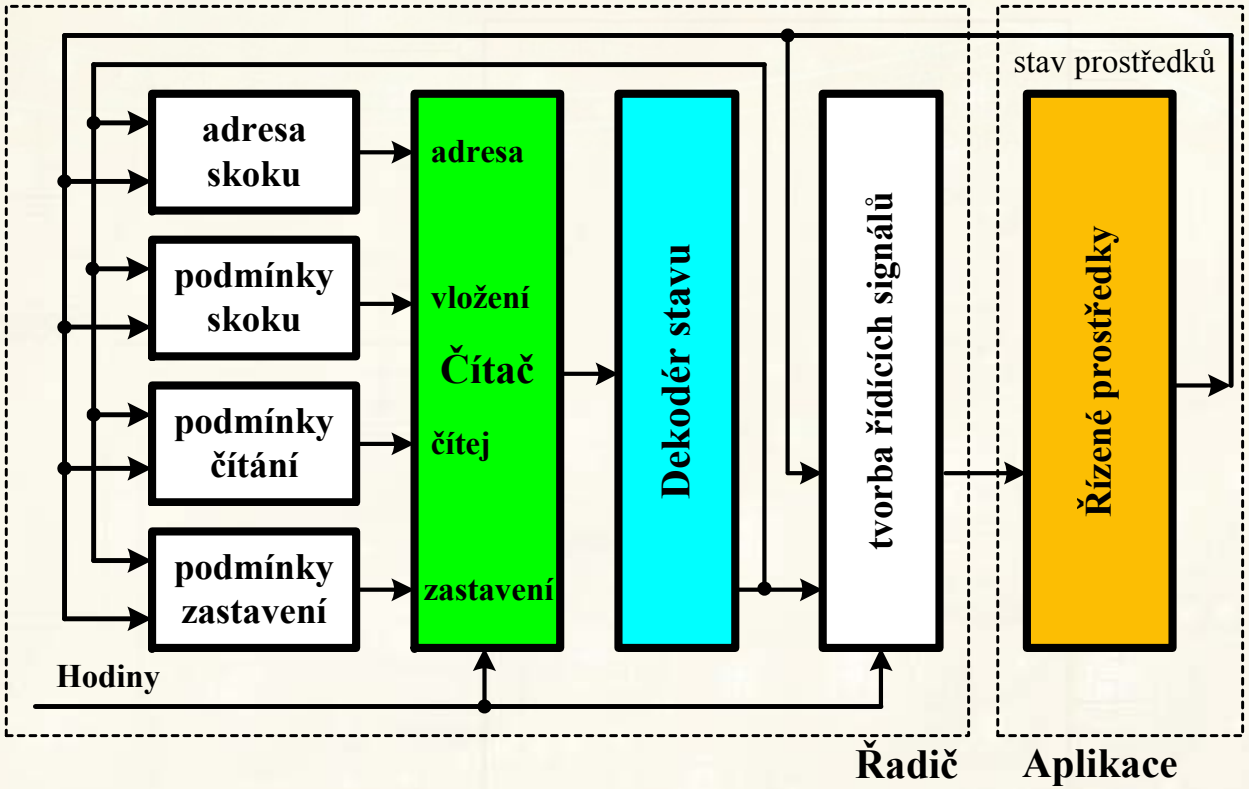
S^i \ SPUST, STOP	00	10	11	01
S_0	S_0	S_1	S_1	S_0
S_1	S_2	S_2	S_2	S_2
S_2	S_3	S_3	S_3	S_3
S_3	S_4	S_4	S_4	S_4
S_4	S_4	S_4	S_5	S_5
S_5	S_6	S_6	S_6	S_6
S_6	S_0	S_0	S_0	S_0

S^{i+1}

REALIZACE ŘADIČE PRO DYNAMICKY OVLÁDANÝ DISPLEJ - VAR.1



OBVODOVÝ ŘADIČ – TYPICKÁ KONFIGURACE

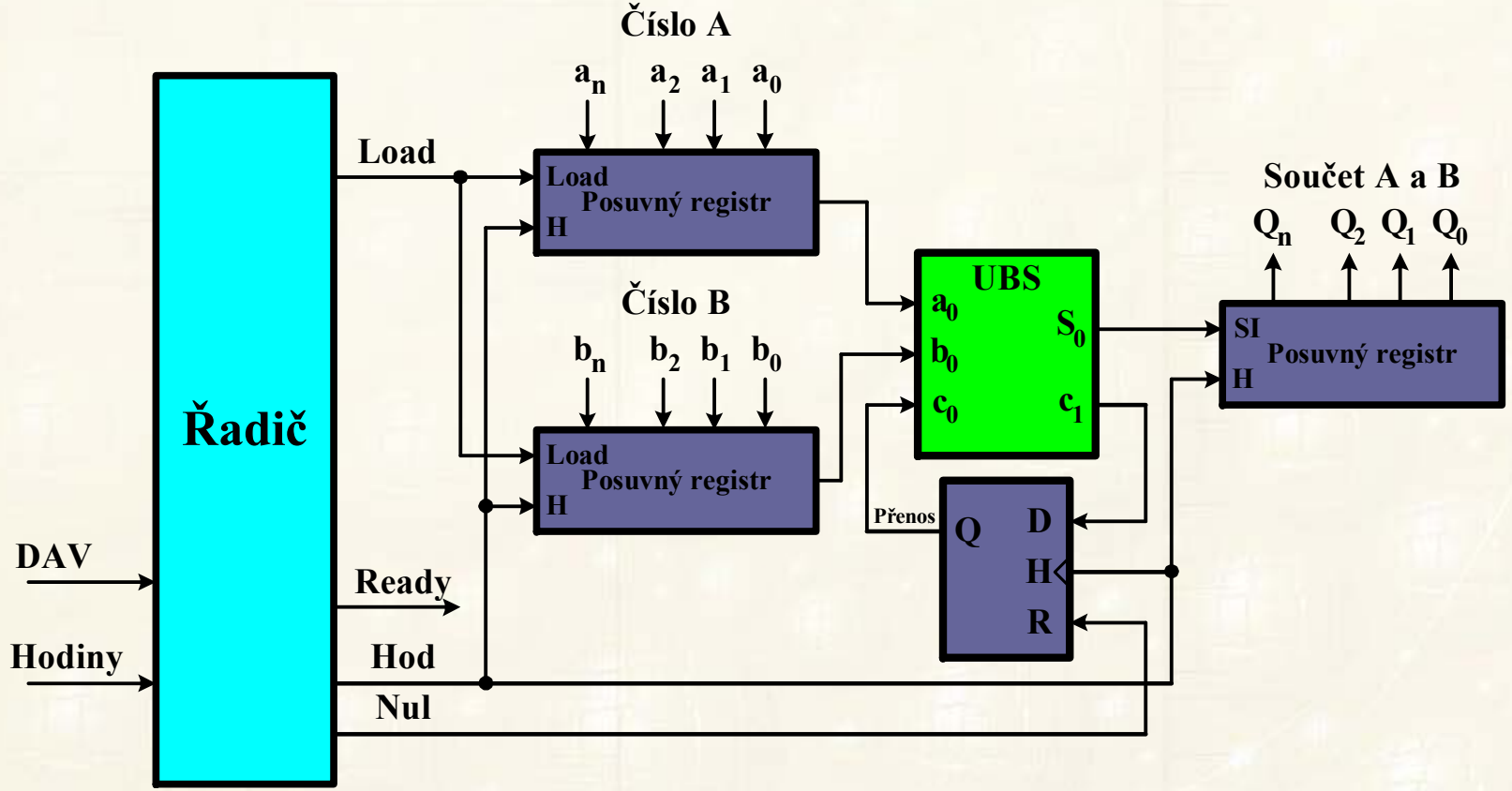


- **Složité obvodové řadiče** – čítač nebo posuvný registr s rotující log.1 dekódovaný LKO na řídící signály.
- **Nevýhoda** – jakákoliv změna v řízení ⇒ Nutná modifikace dekodéru a tvorby řídících signálů (LKO).

PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

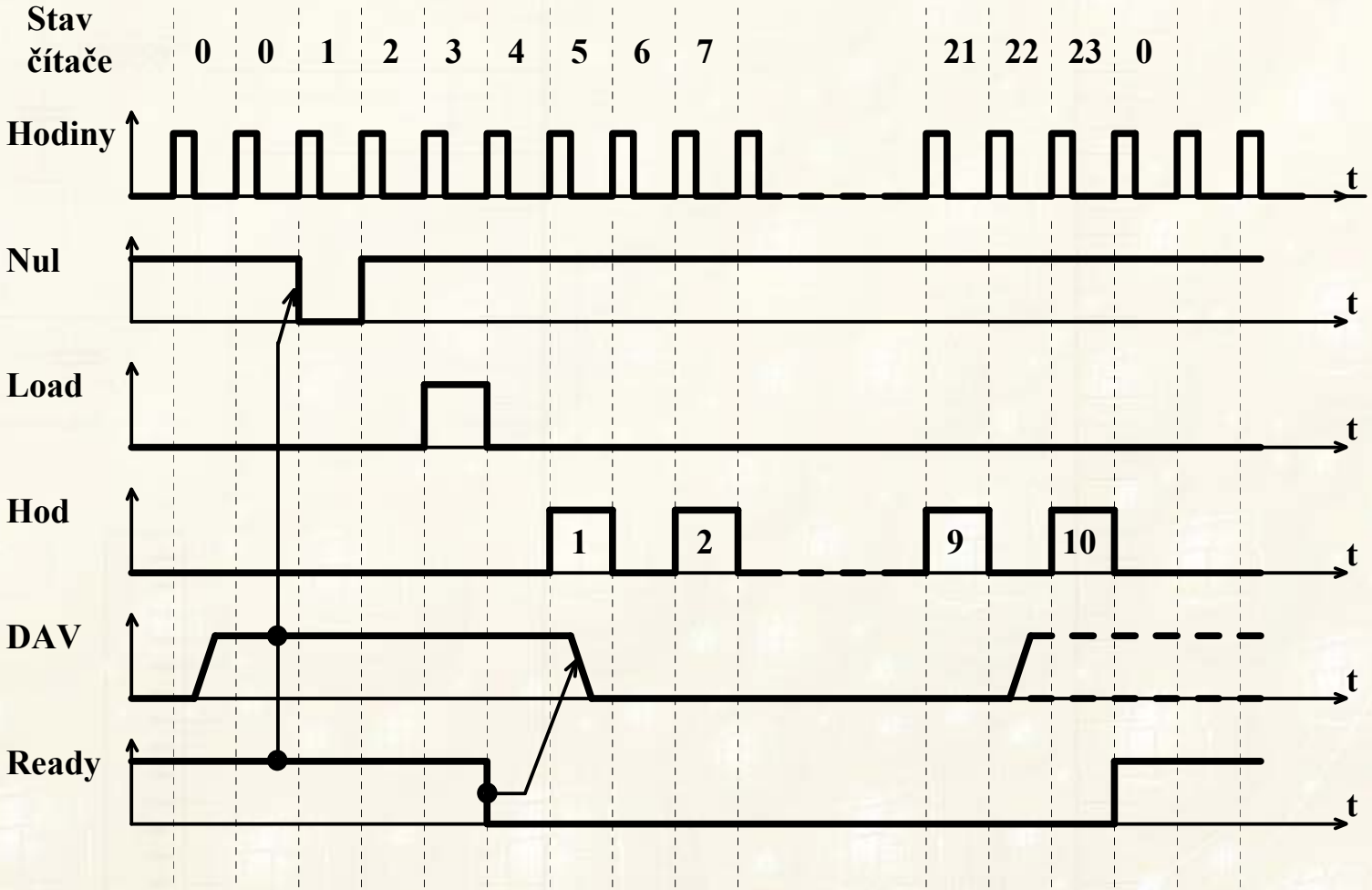
Příklad: Navrhněte obvodový řadič pro sériovou sčítačku dvou desetibitových dvojkových čísel X a Y.

Řízené prostředky = sériová sčítačka z obrázku. Řízení bude direktivní (do řadiče nejde žádný zpětný signál o stavu) ⇒ řídící signály musí vyhovovat časovým parametrům sériové sčítačky.

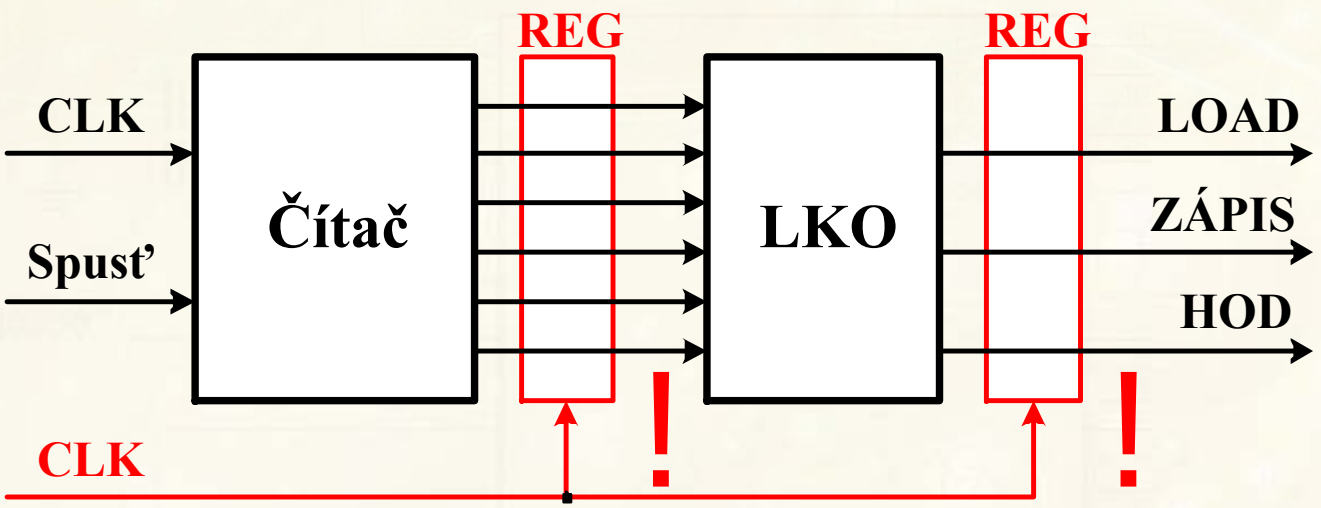


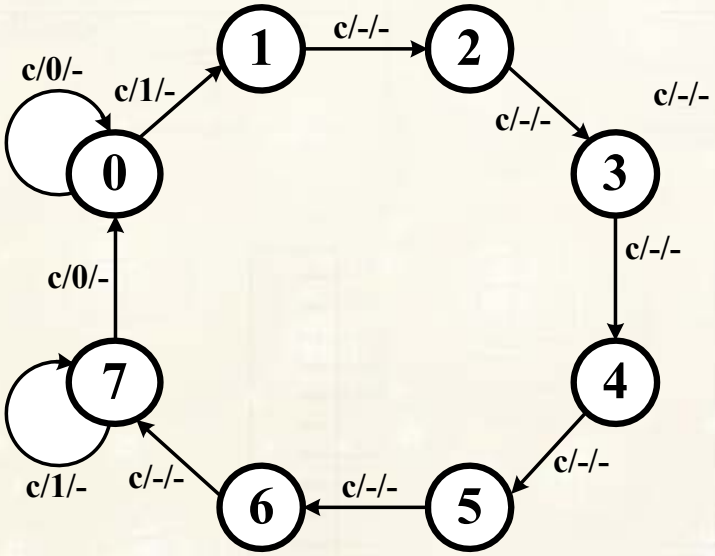
PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

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.



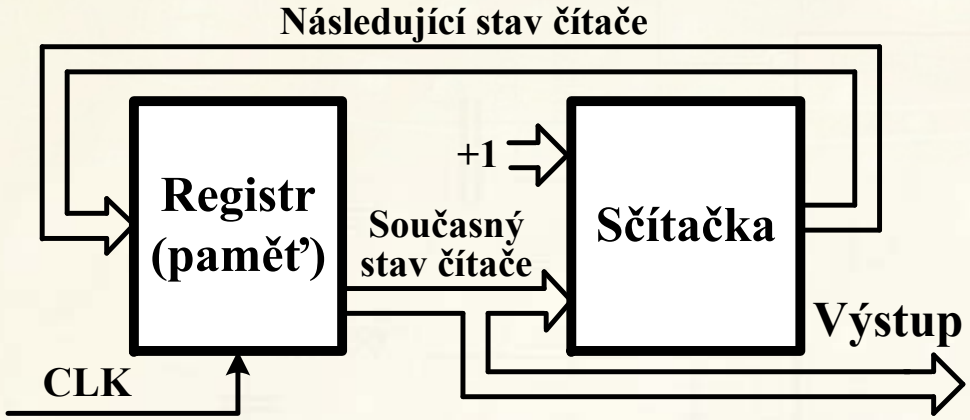
OBVODOVÝ ŘADIČ – JEDNODUCHÁ KONFIGURACE





- **Jednodušší řadič – spouštěný čítač.**
  - **Návrh** ze stavového diagramu
  - **Zkrácení cyklu TTL obvodu**
  - Realizace ve VHDL/Verilog
  - **Registrem a sčítačkou**
  - **Automodifikací registru**

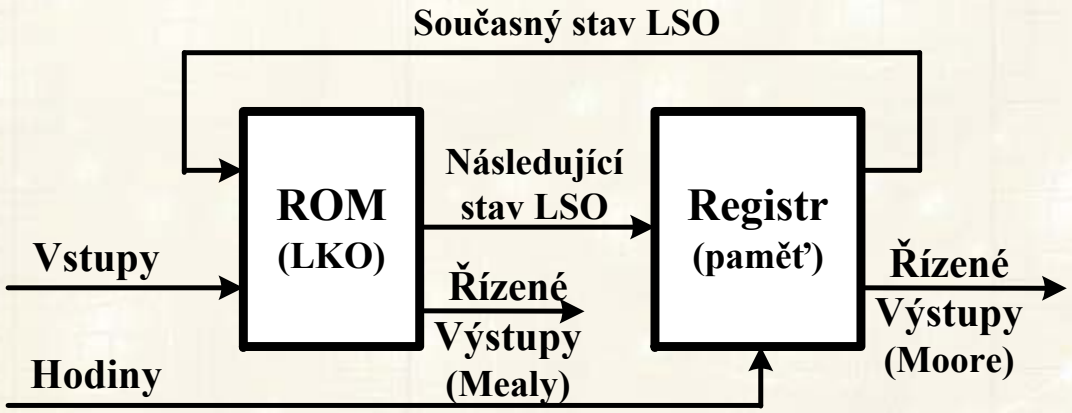
OBVODOVÝ ŘADIČ – MOŽNÉ REALIZACE ČÍTAČE



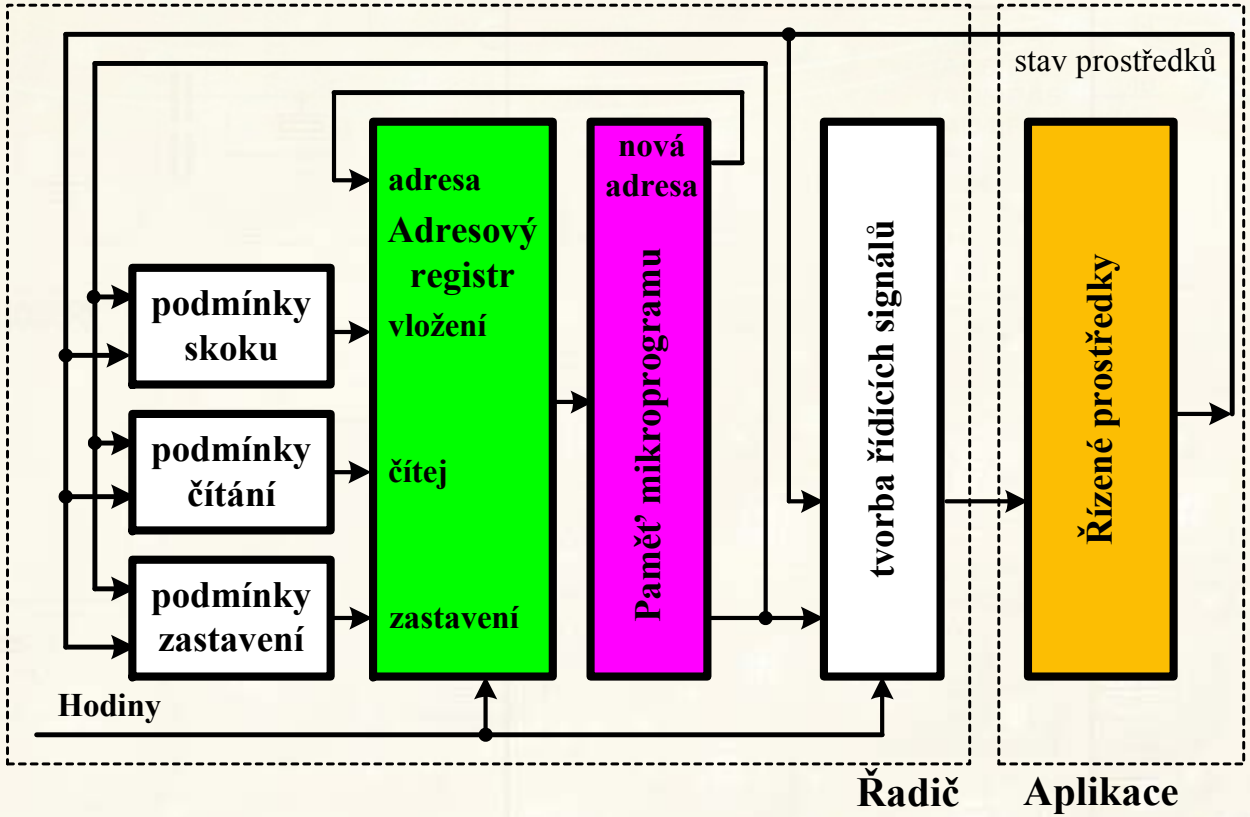
- **Čítač ⇒ +1**
- **Fázový akumulátor ⇒ + >1**
  - **DDS**
  - **Fraktální PLL – dělení neceločíselnou hodnotou**

AUTOMODIFIKACE REGISTRU

- Obecná struktura logického sekvenčního obvodu typu Moore nebo Mealy
- Současný stav LSO = aktuální stav čítače. Následující stav LSO = budoucí stav čítače.

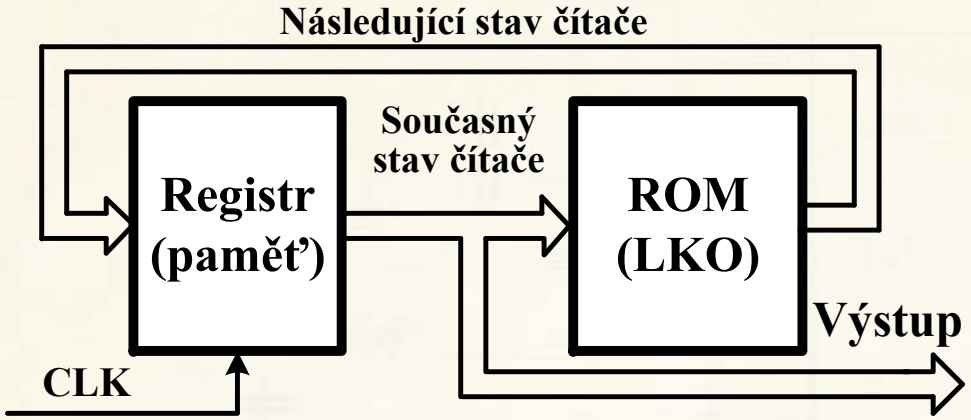


OBVODOVÝ ŘADIČ – PŘECHOD K MIKROPROGRAMOVATELNÉMU

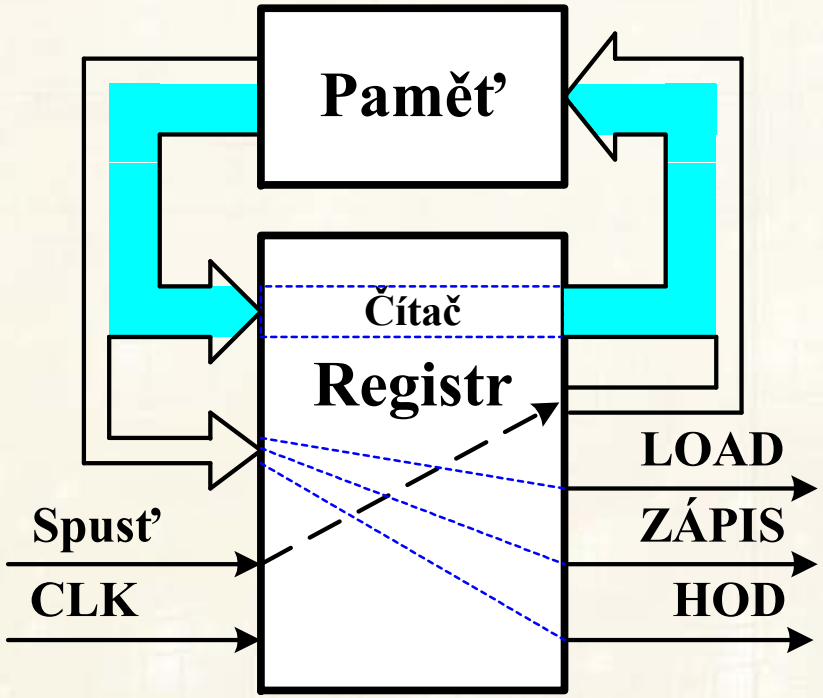


- **Přechod k mikroprogramovatelnému řadiči**
  - Snazší změna signálů ⇒ dekodér nahrazen **pamětí PROM.**
  - Čítač a dekodér nahrazen **mikroprogramovatelným** řadičem.
- **Výhoda –** výměna PROM ⇒ Změna řízení a řídících signálů

MIKROPROGRAMOVATELNÝ ŘADIČ

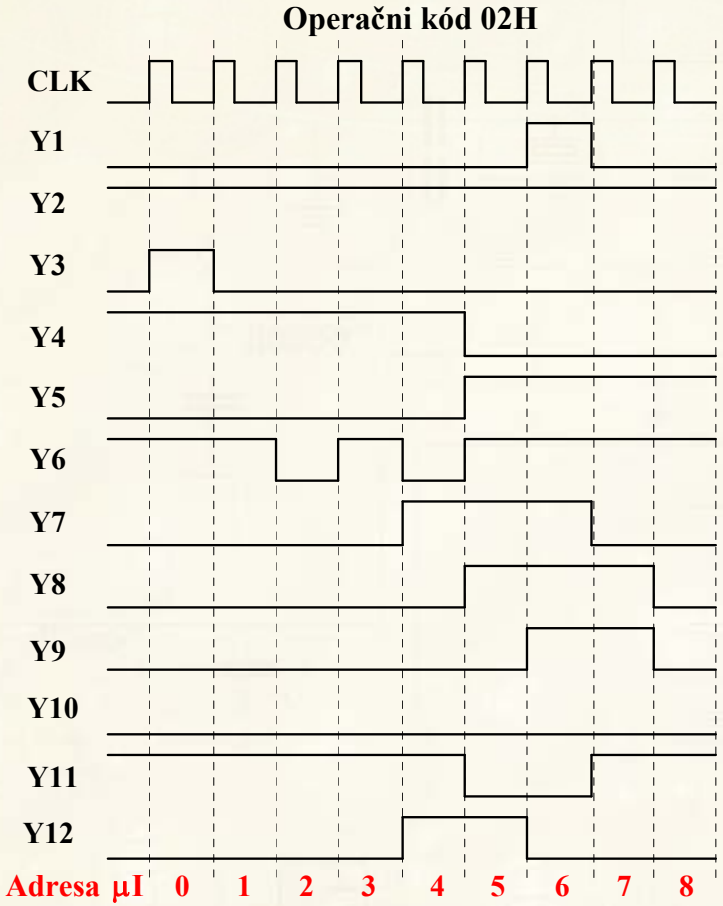


- Část výstupů registru = stav čítače
- Zbývající výstupy = řízené signály
- Výstup ROM = následující stav čítače + řídící signály



- Adresa do paměti
  - Vstupy řadiče + stav čítače
  - Vstup operačního kódu µP + stav čítače
    - Čítač určuje počet fází realizace instrukce

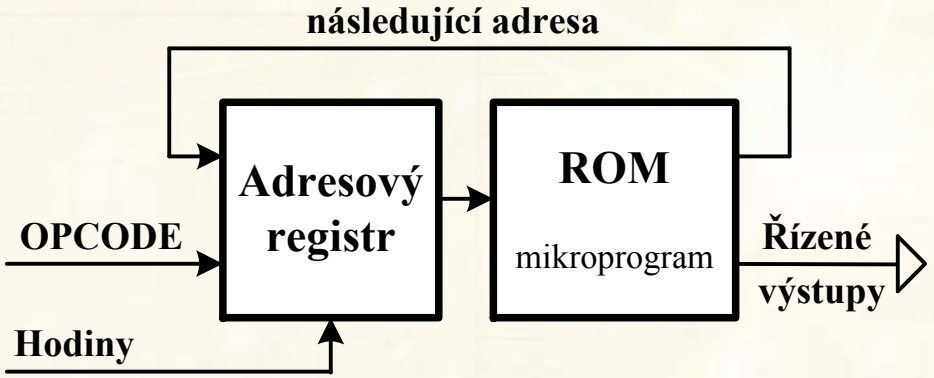
MIKROPROGRAMOVATELNÝ ŘADIČ



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 + stav čítače
  - ❑ Operační kód + adresa mikroinstrukce
- ➢ Výstup ROM
  - ❑ Řídicí signály + následující stav čítače
  - ❑ Řídicí signály + adresa následující mikroinstrukce

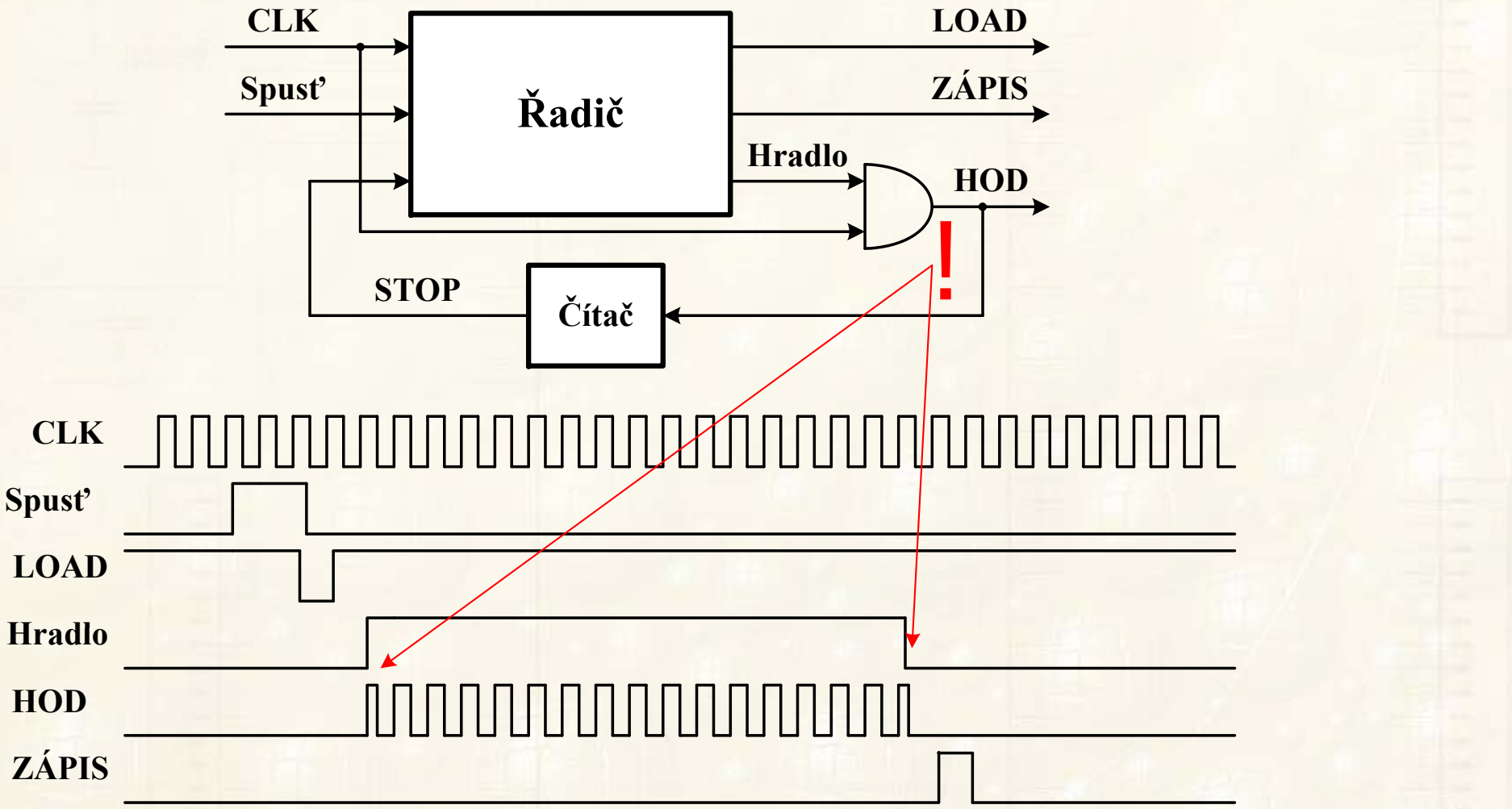
PŘÍKLAD NÁVRHU OBVODOVÉHO ŘADIČE

Možné chyby při návrhu obvodového řadiče:

- Připojení výstupu dekodéru přímo na výstupy synchronního nebo asynchronního čítače. Existují malé nebo nezanedbatelné **rozdíly** ve změnách výstupů a **hazardní stavy** v LKO ⇒ vznik parazitních impulzů. Odstranění zápisem do vyrovnávacího registru při stabilním stavu výstupů.
- Z navrženého časování vyplývají doby trvání nulovacího, nastavovacího impulzu a perioda hodinového signálu. Tyto hodnoty musí 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 signálu ***Hod***. Toto řešení má svá úskalí v zúžení signálu ***Hodiny*** v úrovni log.1 (**riskantní**) a falešném impulzu na konci hradlování.

REALIZACE ŘADIČE PRO DYNAMICKY OVLÁDANÝ DISPLEJ - VAR.1

Řadič s malým počtem stavů využívající **hradlování hodinového** signálu. **Výhody**: signál HOD má 2x vyšší kmitočet proti ostatním řešením. **Nevýhody**: Nebezpečí krátkého impulzu HOD.



Dodatek

VHDL realizace paměťových členů

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;
```

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;
```

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;
```

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;
```

IMPLEMENTACE REGISTRU ŘÍZENÉHO NÁBĚŽNOU HRANOU

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

entity reg_4 is
port (reset, clk: in std_logic;
D: in std_logic_vector (3 downto 0);
Q: out std_logic_vector (3 downto 0));
end entity reg_4;

architecture behavior of reg_4 is
begin
process (reset,clk)
begin
if  reset = '0' then          -- registr s asynchronním nulováním
Q <= "0000";
elsif clk' event and clk = '1' then – náběžná hrana
Q <= D;
end if;
end process;
end behavior;
```

IMPLEMENTACE LATCH ŘÍZENÉHO ÚROVNÍ

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

entity d_latch8 is
port (H: in std_logic;
        D: in std_logic_vector (7 downto 0);
          Q, Qneg: out std_logic_vector (7 downto 0));
end entity d_latch8;

architecture behavior of d_latch8  is
begin
process (H,D)
begin
if  H = '1' then                    -- latch s řízením log.1
Q <= D;
Qneg <= not D;
end if;
end process;
end behavior;
```

IMPLEMENTACE REGISTRU ŘÍZENÉHO NÁBĚŽNOU HRANOU

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

entity shift_reg4  is
port (par: in std_logic_vector(3 downto 0);
        clk, ser, load: in std_logic;
        D: buffer std_logic_vector (3 downto 0));
end entity shift_reg4 ;

architecture behavior of  shift_reg4  is
begin
process (clk)
begin
   if clk' event and clk = '1' then
      if  load = '1' then  D <= par;
      else  D(3) <= D(2); D(2) <= D(1); D(1) <= D(0); D(0) <= ser;
      end if;
   end if;
end process;
end behavior;
```

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;
```