

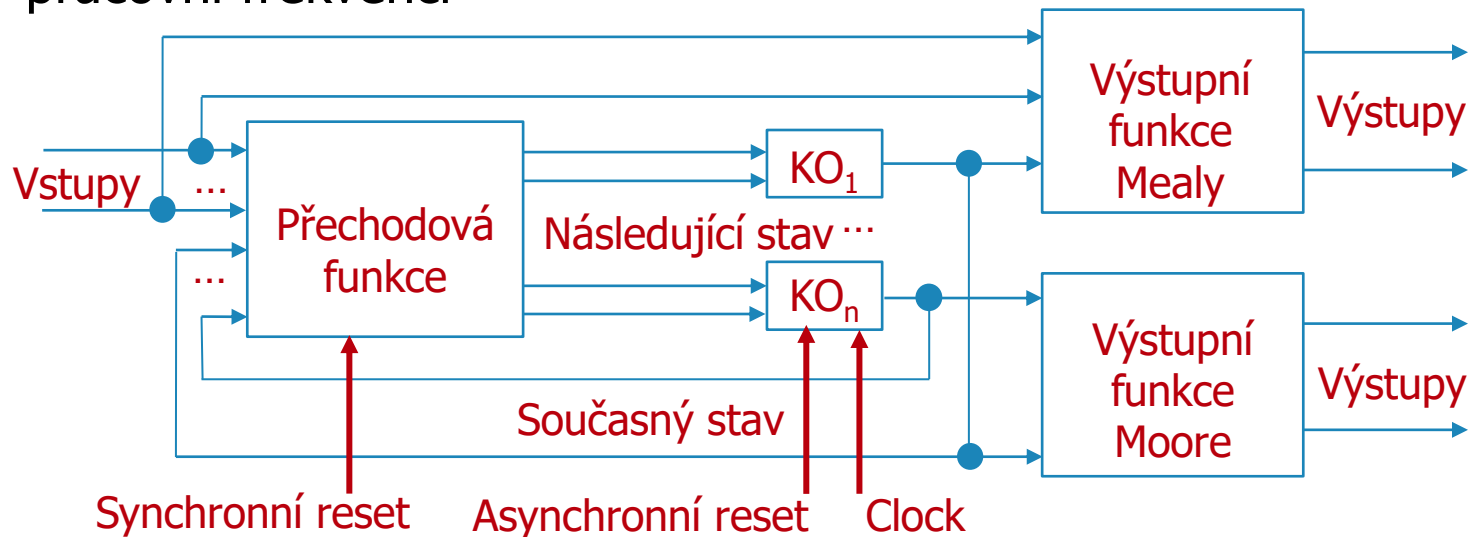
Návrh číslicových systémů (INC)

Otto Fučík

Vysoké učení technické v Brně
Fakulta informačních technologií
Božetěchova 2, 612 66 Brno



- Paměť stavu
 - Flip-flop klopné obvody s globální synchronizací (hodinový signál)
 - KO vzorkuje hodnoty na excitačních vstupech pouze v době platné hrany hodinového signálu, ve které musí být vstupy stabilní => jednoduchý návrh sekvenčních sítí
- Nevýhody
 - Pomalé, větší příkon, složitý a drahý rozvod hodinového signálu, problém elektromagnetické kompatibility, nejhorší zpoždění určuje pracovní frekvenci



- Nevýhody

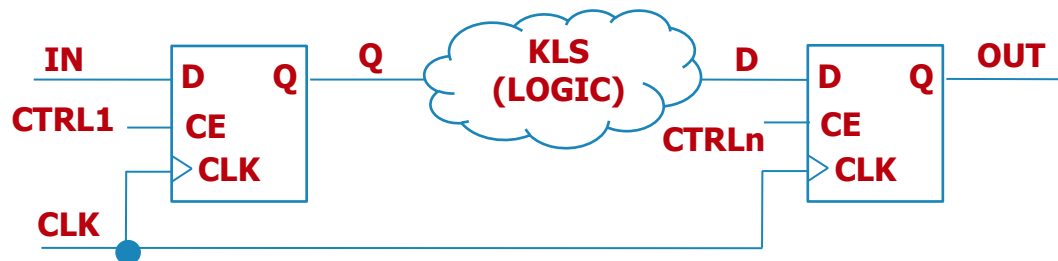
- Pokud je v obvodu více kombinačních sítí mezi synchronizačními KO, musí být perioda hodin přizpůsobena té nejpomalejší z nich (do jisté míry lze omezit vhodným návrhem – zřetězení zpracování atd.)
- Typicky je celý obvod řízen jedním hodinovým signálem – rozvod hodinových signálů musí být stabilní (jitter) a s malým zpožděním (skew) mezi jednotlivými KO – viz dále

- Výhody

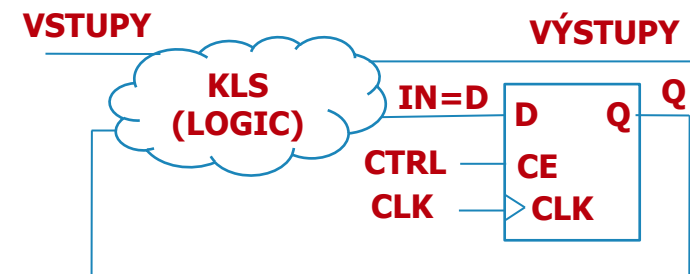
- Jednoduchost - stačí dodržet, aby kombinační logická síť mezi jednotlivými KO synchronního obvodu měla takové zpoždění, aby pro danou periodu hodinového signálu byly dodrženy časy „ t_{setup} “ a „ t_{hold} “ definované pro jednotlivé KO
- KO izolují části kombinačních logických sítí (KLS) od sebe - výstup KLS se vzorkuje po odeznění přechodových dějů
- Tok informace se ovládá z řadiče pomocí signálů CE jednotlivých KO
- Lze automatizovat syntézu logických systémů z popisu jejich chování na vysoké úrovni abstrakce (VHDL, Verilog atd.)

- Příklad: Kombinační logická síť (KLS, LOGIC) a D klopný obvod (registr)
 - T_{CQ} : (CLK to Q propagation delay) nejhorší zpoždění od okamžiku kladné hrany hodin CLK do platné hodnoty výstupu Q
 - $T_{CQ(CO)}$: (CLK to Q contamination delay) minimální doba od platné hrany hodin do okamžiku, kdy se výstup Q začne měnit (pozor na rozdíl od T_{CQ} , kde se uvažuje okamžik, kdy Q nabude nové hodnoty)
 - T_{SU} : setup time
 - T_H : hold time
 - T_{LOGIC} : (LOGIC propagation delay) nejhorší zpoždění kombinační logické sítě
 - $T_{LOGIC(CO)}$: (LOGIC contamination delay) minimální doba od okamžiku platných vstupů KLS do okamžiku, kdy se začnou měnit výstupy
- Stejně pro sekvenční sítě bez i se zpětnou vazbou

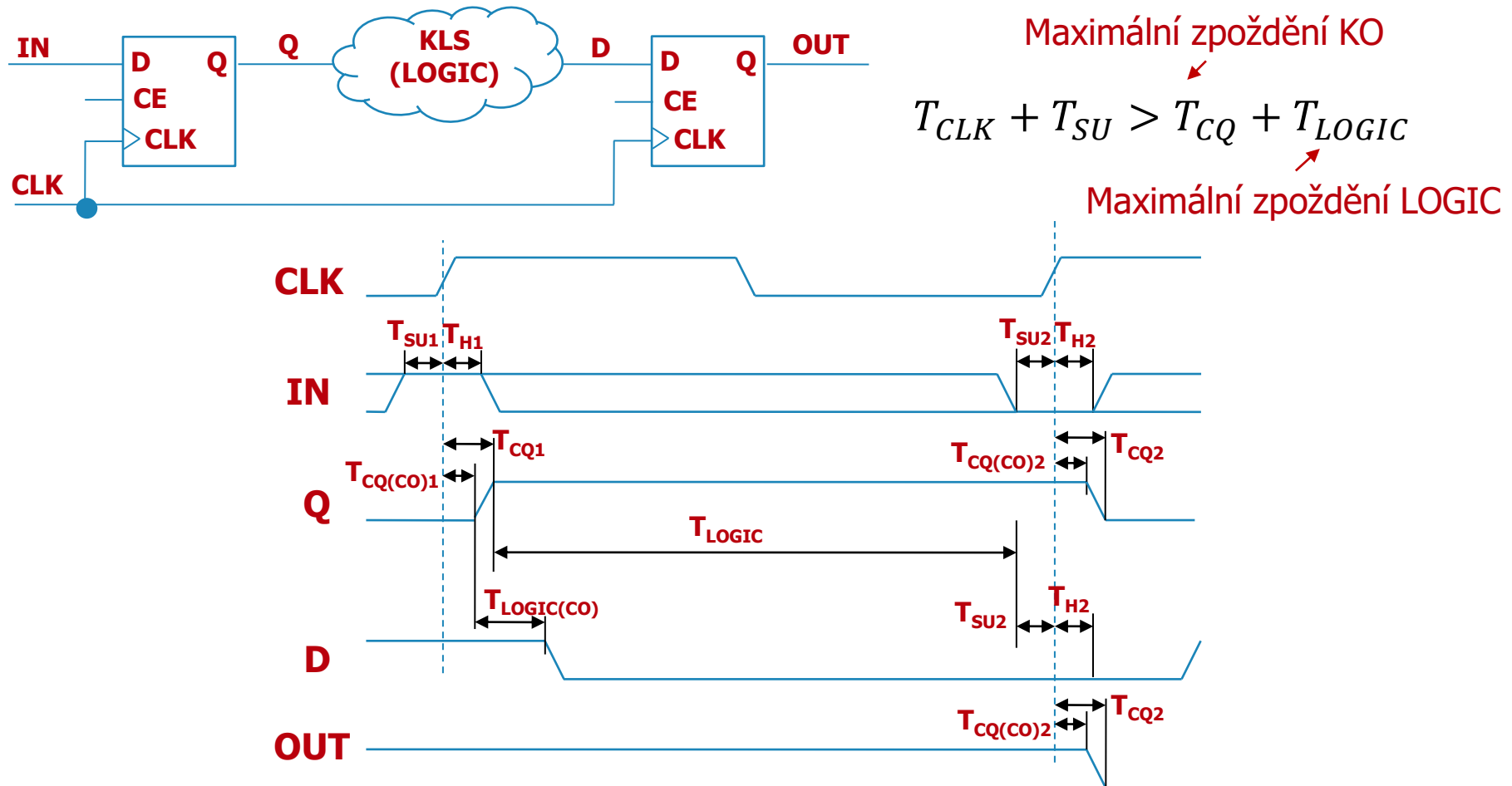
Sekvenční síť bez zpětné vazby



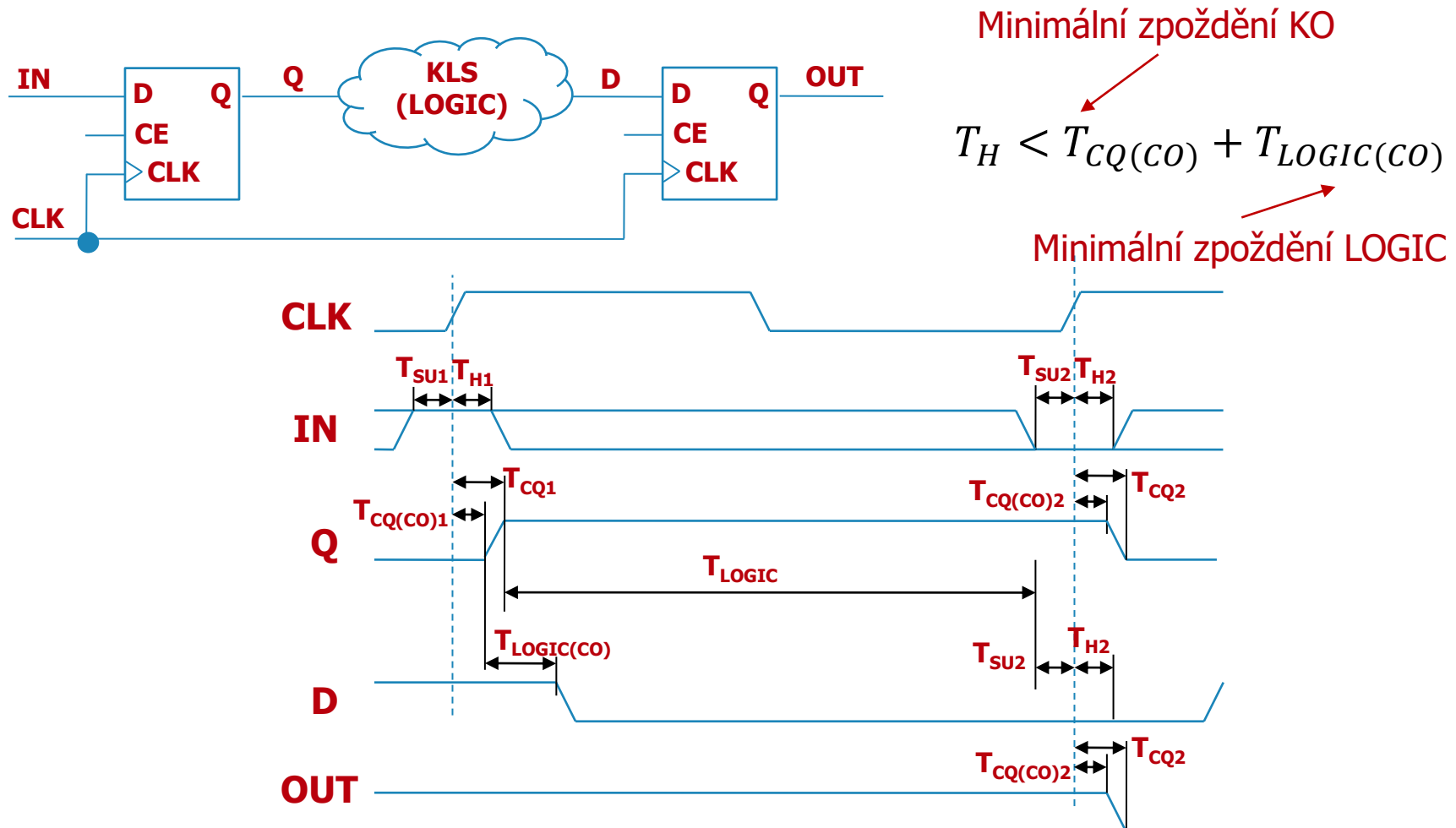
Sekvenční síť se zpětnou vazbou - konečný automat



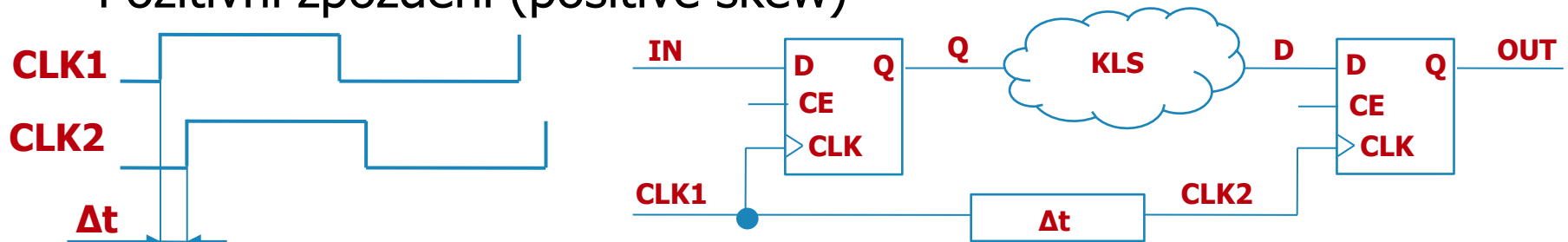
- Minimální perioda hodinového signálu
 - Předpokládejme, že parametry klopných obvodů jsou stejné
 - Perioda hodinového musí být dostatečně dlouhá



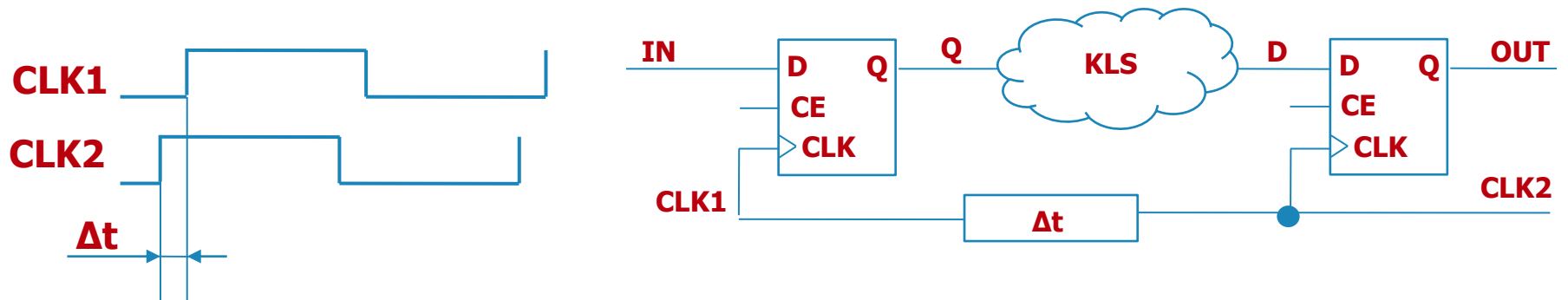
- Minimální zpoždění komponent
 - Musí být dodržen „hold time“ klopných obvodů



- Pozitivní zpoždění (positive skew)



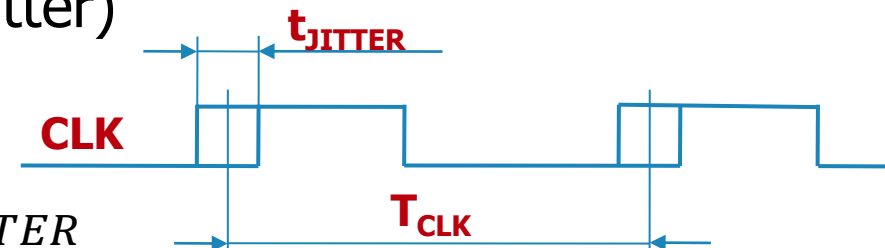
- Musí platit: $T_H + \Delta t < T_{CQ(CO)} + T_{LOGIC(CO)}$
- Negativní zpoždění (negative skew)



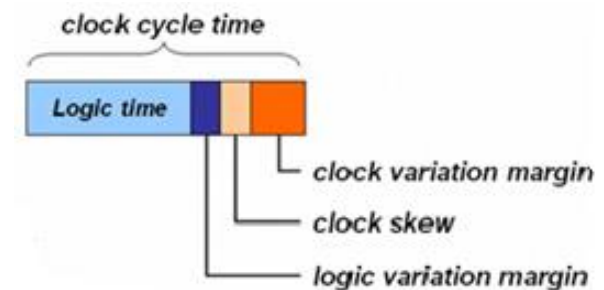
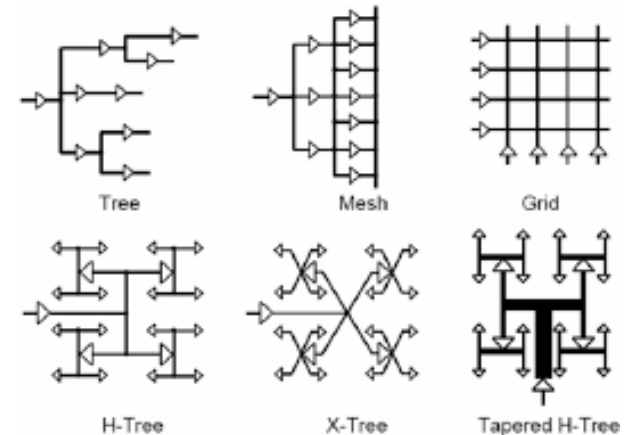
- Nestabilita hodinového signálu (jitter)

- Musí platit:

$$T_{CLK} > T_{CQ} + T_{LOGIC} + T_{SU} + 2 \cdot t_{JITTER}$$

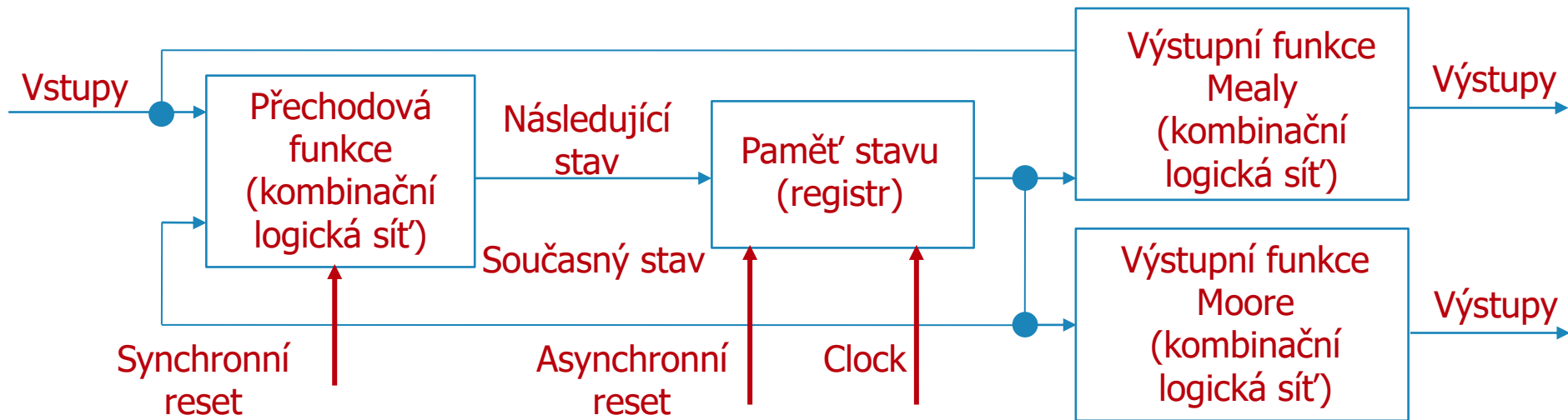


- Globální hodinový signál
 - Určuje počet operací za sekundu
 - Drahý a rozsáhlý systém distribuce
 - Zpoždění nejdelší logické větve určuje frekvenci, na které obvod pracuje
- Fyzikální limity
 - Rozptyl parametrů – problém implementace hodinově synchronních obvodů
 - Důsledkem je omezení pracovní frekvence
 - Až 40% hodinového cyklu je využito pro rozvod hodinového signálu
 - Kombinační logická síť mezi jednotlivými registry musí být tak rychlá, aby její výstupy byly stabilní za dobu kratší, než je cca 60 % periody hodinového signálu



[Zdroj: Synopsys]

- Finite State Machine (FSM) a využitím KO
- Paměť současného stavu - klopné obvody (KO)
 - KO jsou citlivé na hranu hodinového signálu (Clock)
 - KO lze asynchronně (nezávisle na Clock) nastavit (např. signálem Preset) či nulovat (např. Clear) – počáteční stav (q_0)
 - Alternativně může být účelné uvést automat do počátečního stavu q_0 i synchronně – signálem „synchronní reset“ se vnutí přechodové funkci, aby (na základě aktivní hrany hodin) generovala stav q_0



- Stavy automatu jsou reprezentovány unikátními kódy
 - Vhodný kód se volí dle aplikace, s ohledem na technologické aspekty návrhu (rušení apod.), optimalizaci výsledné implementace atd.
- Počet klopných obvodů = $\lceil \log_2(\text{počet stavů}) \rceil$  Zaokrouhleno nahoru
 - Např. na 6 bitech můžeme kódovat až $2^6=64$ různých stavů
 - Např. 9 stavů musíme kódovat na alespoň 4 bitech, neboť $2^4=16>9$ (celkem 7 možných kódových kombinací nebude využito)
- Nepoužité (nevyužité) stavy
 - Sekvenční obvod může přejít vlivem např. rušení do nevyužitého stavu (neočekávané chování)
 - Pro omezení rizika nesprávné činnosti, může být třeba tuto situaci ošetřit – z nevyužitých stavů se přechází do stavů využitých (např. počátečního)
- Počáteční stav
 - Nutno volit s ohledem na jeho snadné vynucení (reset) – nejčastěji 00..0 či 11...1 (asynchronní vstupy Preset a Clear klopných obvodů)

- Binární
 - Číslo stavu je dáno příslušným binárním číslem
- Grayův
 - Sousední hodnoty se mění v jednom bitu
 - Výhodné s ohledem na příkon, rušení, souběhy atd.
- Johnsonův (plazivý)
- 1 z n (one-hot)
 - Každý stav má k dispozici jeden KO – např. 18 stavů vyžaduje 18 KO

Binární	Grayův	Johnsonův	1 z n (one-hot)
0000	0000	00000000	0000000000000000 1
0001	0001	00000001	0000000000000000 10
0010	0011	00000011	0000000000000000 100
0011	0010	00000111	0000000000000000 1000
0100	0110	00001111	000000000000 10000
0101	0111	00011111	0000000000 100000
0110	0101	00111111	00000000 1000000
0111	0100	01111111	0000000 10000000
1000	1100	11111111	0000000 100000000
1001	1101	11111110	000000 1000000000
1010	1111	11111100	00000 10000000000
1011	1110	11111000	0000 100000000000
1100	1010	11110000	000 1000000000000
1101	1011	11100000	00 10000000000000
1110	1001	11000000	0 100000000000000
1111	1000	10000000	1000000000000000

- Kolik je celkem možností přiřazení kódů jednotlivým stavům automatu? (state encoding)
 - k – počet požadovaných stavů automatu
 - n – počet možných stavů (počet kombinací daných počtem bitů)
 - p – počet kombinací výběru k požadovaných stavů z n možných (variace bez opakování)

$$p = V(k, n) = \frac{n!}{(n - k)!}$$

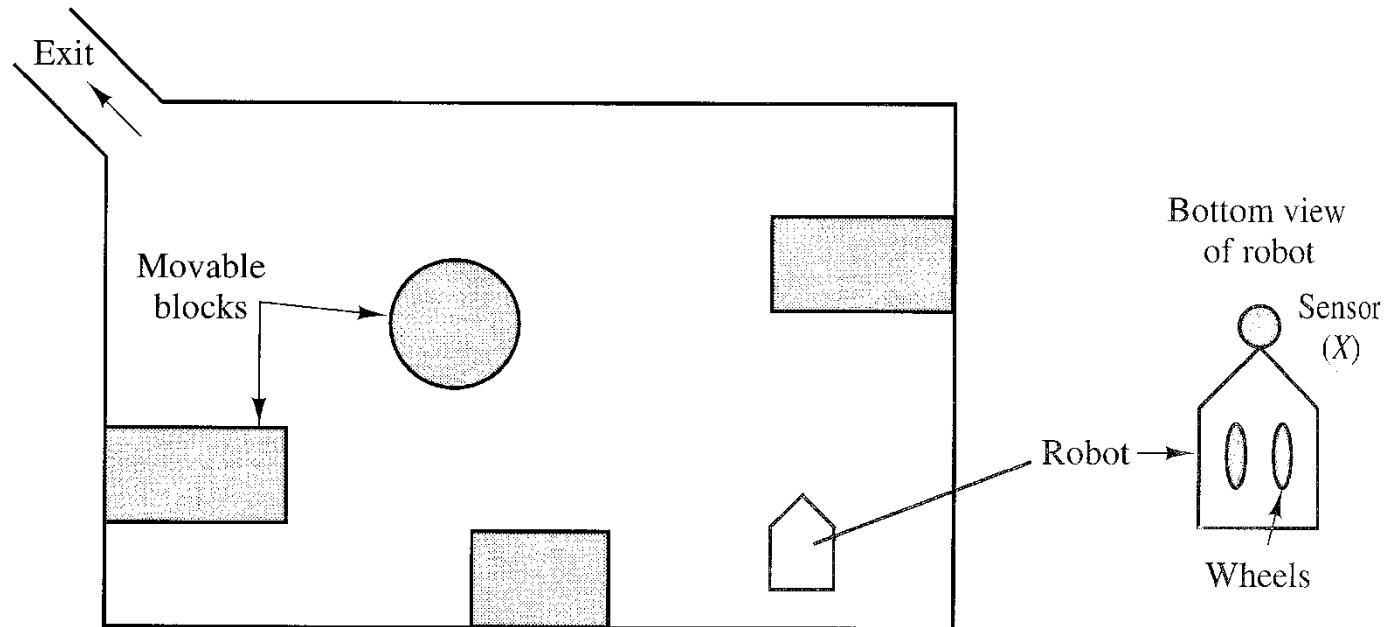
- Příklad: Mějme navrhnout automat s pěti stavy
 - Potřebujeme nejméně 3 bity pro kódování stavů => máme celkem 8 možných stavů
 - Počet možných přiřazení stavů vypočítáme jako:

$$p = \frac{8!}{(8 - 5)!} = \frac{8!}{3!} = 6720$$

- Pro daný automat máme celkem 6720 možností, jak zakódovat stavy

- Navrhněme řídicí obvod (konečný automat) robota, který bude fungovat následovně
 - Robot má dvě kolečka a umí se otáčet vlevo a vpravo
 - Výstup $z1=1$ – zatoč vlevo
 - Výstup $z2=1$ – zatoč vpravo
 - Výstupy $z1=0$ a $z2=0$ – robot jede dopředu
 - Robot má vpředu dotykový senzor, který generuje signál x následovně
 - Vstup $x=1$ – je detekována překážka
 - Vstup $x=0$ – žádná překážka není detekována
 - Robot má na střeše majáček
 - Výstup $m=1$ – svítí, pokud robot jede dopředu
 - Výstup $m=0$ – nesvítí, pokud robot zatáčí vlevo či vpravo
 - Jedno z možných „inteligentních“ chování robota – definice stavů řídicího automatu
 - Stav A – překážka není detekována, poslední zatočení bylo vlevo
 - Stav B – překážka detekována, zatoč vpravo
 - Stav C – překážka není detekována, poslední zatočení bylo vpravo
 - Stav D – překážka detekována, zatoč vlevo

- Díky své „inteligenci“ může robot nalézt cestu ven z uzavřeného prostoru
 - Intelligence robota je dána algoritmem, který implementuje
 - Lepší algoritmus = lepší chování robota
- Pohled na vymezený prostor pro pohyb robota a pohled na robota zespodu



- Definice výstupů
 - Robot začne zatáčet ihned, jakmile detekuje překážku a po celou dobu její přítomnosti => z1 a z2 jsou Mealyho výstupy, neboť jsou funkcí stavů a vstupu x
 - Majáček svítí jen v případě, že není detekována žádná překážka a robot jede => m je Mooreův výstup, neboť závisí pouze na stavu
- Přiřazení kódů jednotlivým stavům (např. binární)
 - A=00, B=01, C=10 a D=11
 - Budeme tedy potřebovat dva KO pro kódování všech 4 stavů
- Podrobný popis chování robota
 - Pokud je v A a není detekována překážka, robot zůstane v A, jede dopředu a maják svítí
 - Pokud je v A a je detekována překážka, robot přechází do B, zatáčí doprava a maják nesvítí
 - Pokud je v B a je detekována překážka, robot zůstane v B, zatáčí doprava a maják nesvítí
 - Pokud je v B a není detekována překážka, robot přechází do C, jede dopředu a maják svítí
 - Pokud je v C a není detekována překážka, robot zůstane v C, jede dopředu a maják svítí
 - Pokud je v C a je detekována překážka, robot přechází do D, zatáčí doleva a maják nesvítí
 - Pokud je v D a je detekována překážka, robot zůstane v D, zatáčí doleva a maják nesvítí
 - Pokud je v D a není detekována překážka, robot přechází do A, jede dopředu a maják svítí
- Je třeba zvolit, který typ KO bude použit pro paměť stavu
 - V daném příkladu uvedeme všechny možnosti – tedy S-R, J-K, D a T
 - Požadované hodnoty excitačních vstupů KO budeme určovat pomocí tabulky přechodů

- Tabulka definuje
 - Přechodovou funkci
 - Přiřazení kódu jednotlivým stavům
 - Výstupní funkce – Mealyho a Mooreovy výstupy

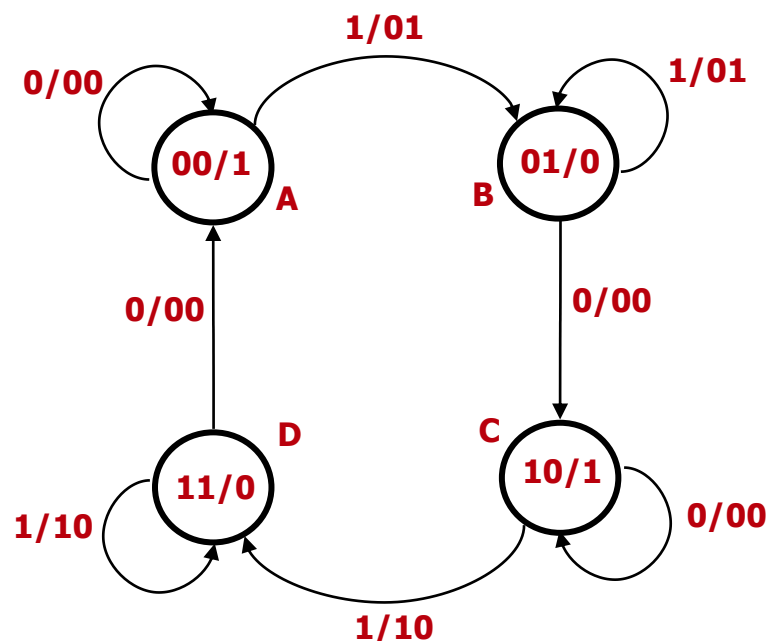
Současný stav		Vstup X	Následující stav		Mealyho výstupy Z1 Z2	Mooreův výstup M
Název	Kód		Název	Kód		
A	00	0	A	00	00	1
A	00	1	B	01	01	1
B	01	0	C	10	00	0
B	01	1	B	01	01	0
C	10	0	C	10	00	1
C	10	1	D	11	10	1
D	11	0	A	00	00	0
D	11	1	D	11	10	0

• Graf přechodů

- Lze nakreslit např. dle slovního popisu či tabulky přechodů
- Ve stavech jsou uvedeny kódy stavu/hodnota Mooreova výstupu M - kód/M
- Na hranách je uvedena hodnota vstupu X/hodnota Mealyho výstupů Z₁ a Z₂ - X/Z₁Z₂

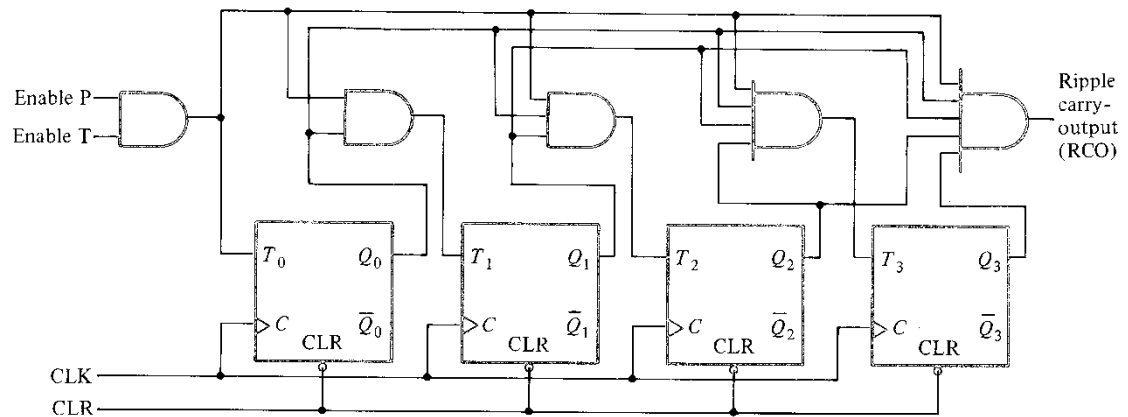
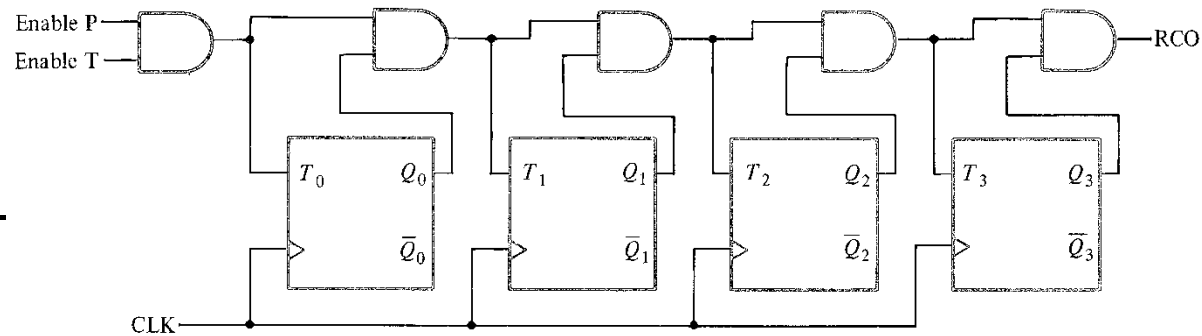
Přechody jsou spouštěny signálem CLK

Současný stav		Vstup X	Následující stav		Mealyho výstupy Z ₁ Z ₂	Mooreův výstup M
Název	Kód		Název	Kód		
A	00	0	A	00	00	1
A	00	1	B	01	01	1
B	01	0	C	10	00	0
B	01	1	B	01	01	0
C	10	0	C	10	00	1
C	10	1	D	11	10	1
D	11	0	A	00	00	0
D	11	1	D	11	10	0



- Speciální případ synchronního automatu (counter)
 - Navrhuje se stejně jako Mooreův automat (výstupní funkce je triviální: stav = výstup) – budeme probírat později
 - Na aktivní hodinový signál automat přechází ze stavu do stavu - dáno čítací posloupností
 - Některé čítače mohou na základě hodnoty na vstupu (např. UP/DOWN) čítat např. nahoru (vzestupná posloupnost stavů) nebo dolů (sestupná posloupnost)
 - Mají též výstup indikující přenosu čítače
 - Anglicky Ripple Carry Output (RCO), Terminal Count (TC) ap.
 - Jedná se o dekódování stavu, kdy čítač přeteče
 - Např. přechod z hodnoty 15 na 0 u čtyřbitového čítače čítajícího nahoru
 - Většinou mají též vstup, pomocí kterého lze řídit činnost čítače
 - Anglicky Clock Enable (CE), Counter Clock Enable (CTEN), T Enable, P Enable ap.
 - $CE=0$...nečítá, pamatuje si poslední hodnotu, $CE=1$...čítá
 - Prodloužení čítací posloupnosti (dělicího poměru)
 - Zapojováním do kaskády tak, že se zapojí výstup RCO (TC) předchozího čítače na CE následujícího

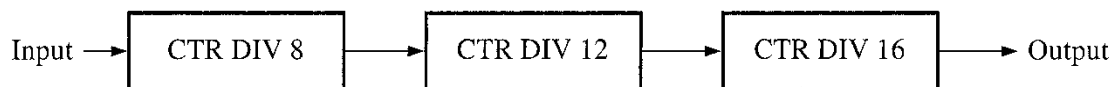
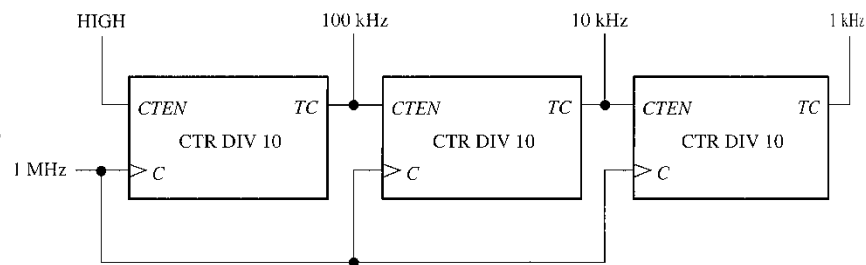
- Implementace dekodéru stavu, ve kterém dochází k přenosu (modulo)
 - Kaskádní dekodér - vyžaduje delší čas k ustálení hodnoty - obdoba šíření přenosu v kaskádní sčítačce
 - Paralelní dekodér
 - Rychlejší
 - Složitější



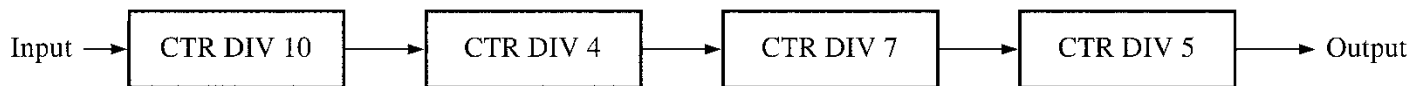
- Kaskádní řazení čítačů
 - Lze použít pro dělení kmitočtů
 - Platí pro asynchronní i synchronní čítače
 - Dělicí poměry (modulo) jednotlivých čítačů zapojených v kaskádě se násobí

- Příklad dělení signálu

- 3 synchronní čítače modulo 10 => poměr je $10 \times 10 \times 10 = 1000$
- (a) příklad děličky $8 \times 12 \times 16 = 1536$
- (b) příklad děličky $10 \times 4 \times 7 \times 5 = 1400$

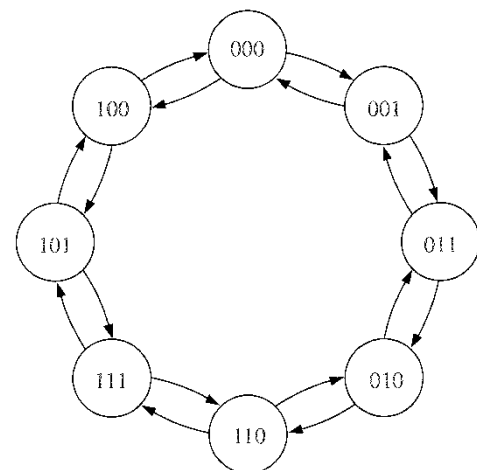


(a)



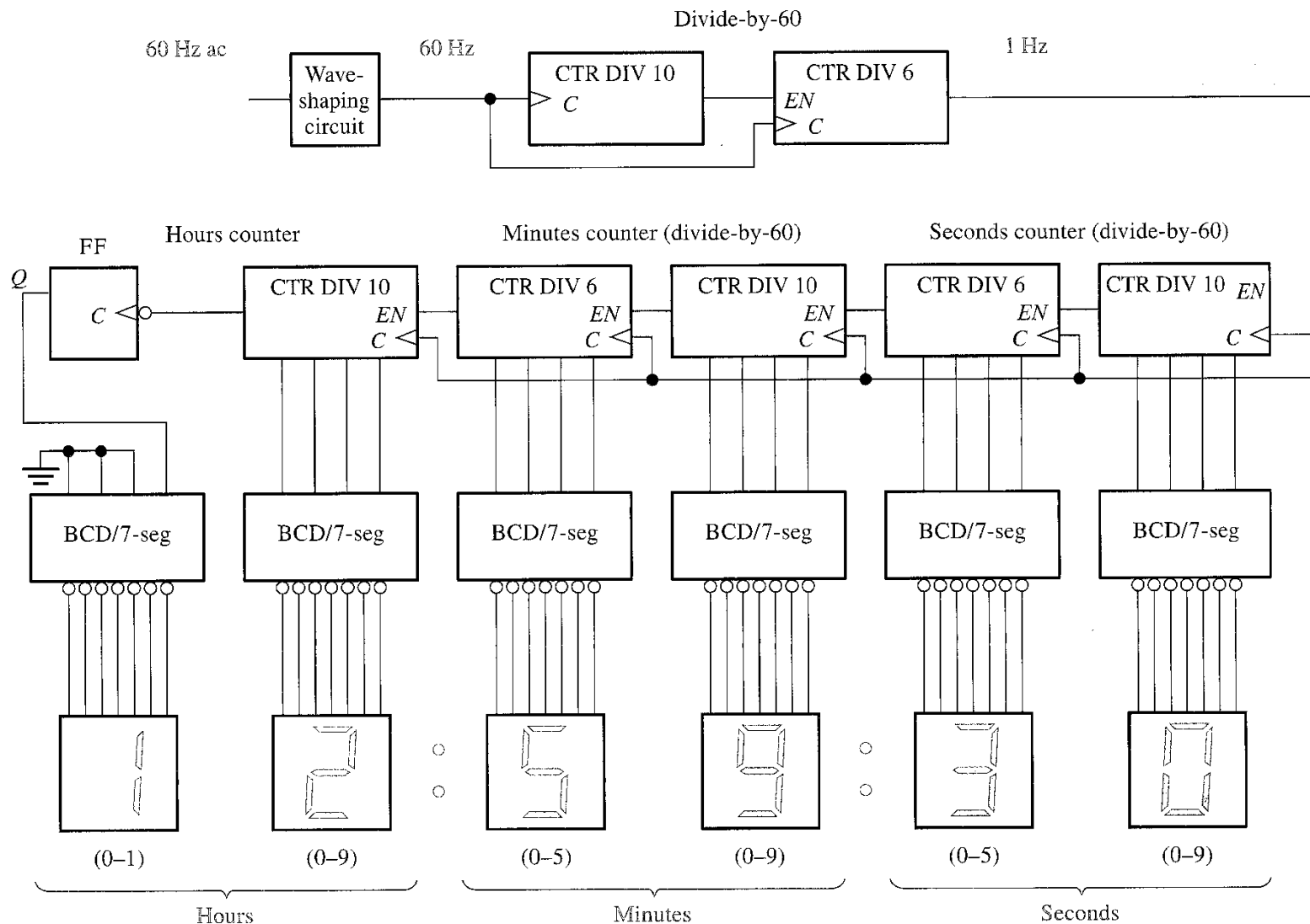
(b)

- Čítač v Grayově kódu
- Graf přechodů
 - Vstup $Y=1$ – přechody po směru hodinových ručiček – čítání nahoru
 - Vstup $Y=0$ – přechody proti směru hodinových ručiček – čítání dolů
- Tabulka přechodů
 - Implementace např. pomocí J-K KO

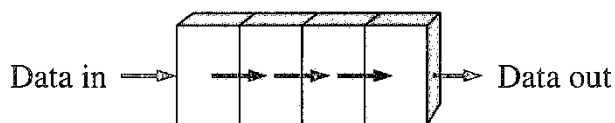


Present State Q_2 Q_1 Q_0			Next State					
			$Y = 0$ (DOWN)			$Y = 1$ (UP)		
			Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	1
0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	1	0
1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	1	0	1
1	0	1	1	1	1	1	0	0
1	0	0	1	0	1	0	0	0

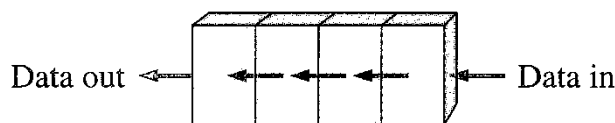
- Generování hodin je odvozeno od kmitočtu sítě (zde 60Hz)



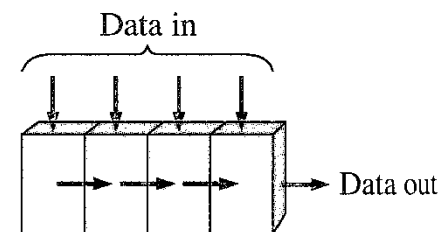
- Synchronní obvod (má společné hodiny), který je sestaven z klopných obvodů zapojených do série (Shift Register)
- Varianty (aritmetický/logický posun/rotace vlevo/vpravo atd.)
- Příklady:



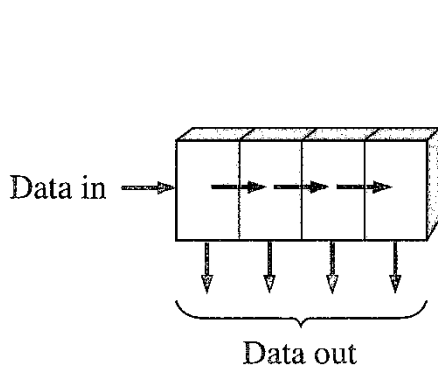
(a) Serial in/shift right/serial out



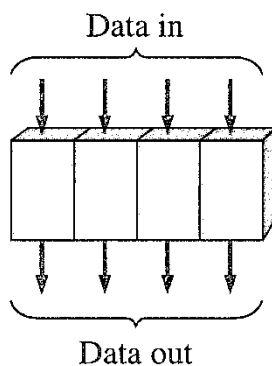
(b) Serial in/shift left/serial out



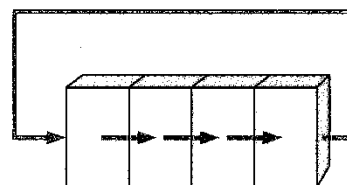
(c) Parallel in/serial out



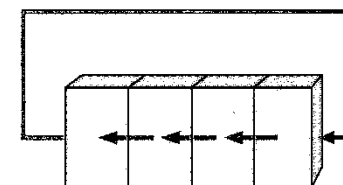
(d) Serial in/parallel out



(e) Parallel in/parallel out



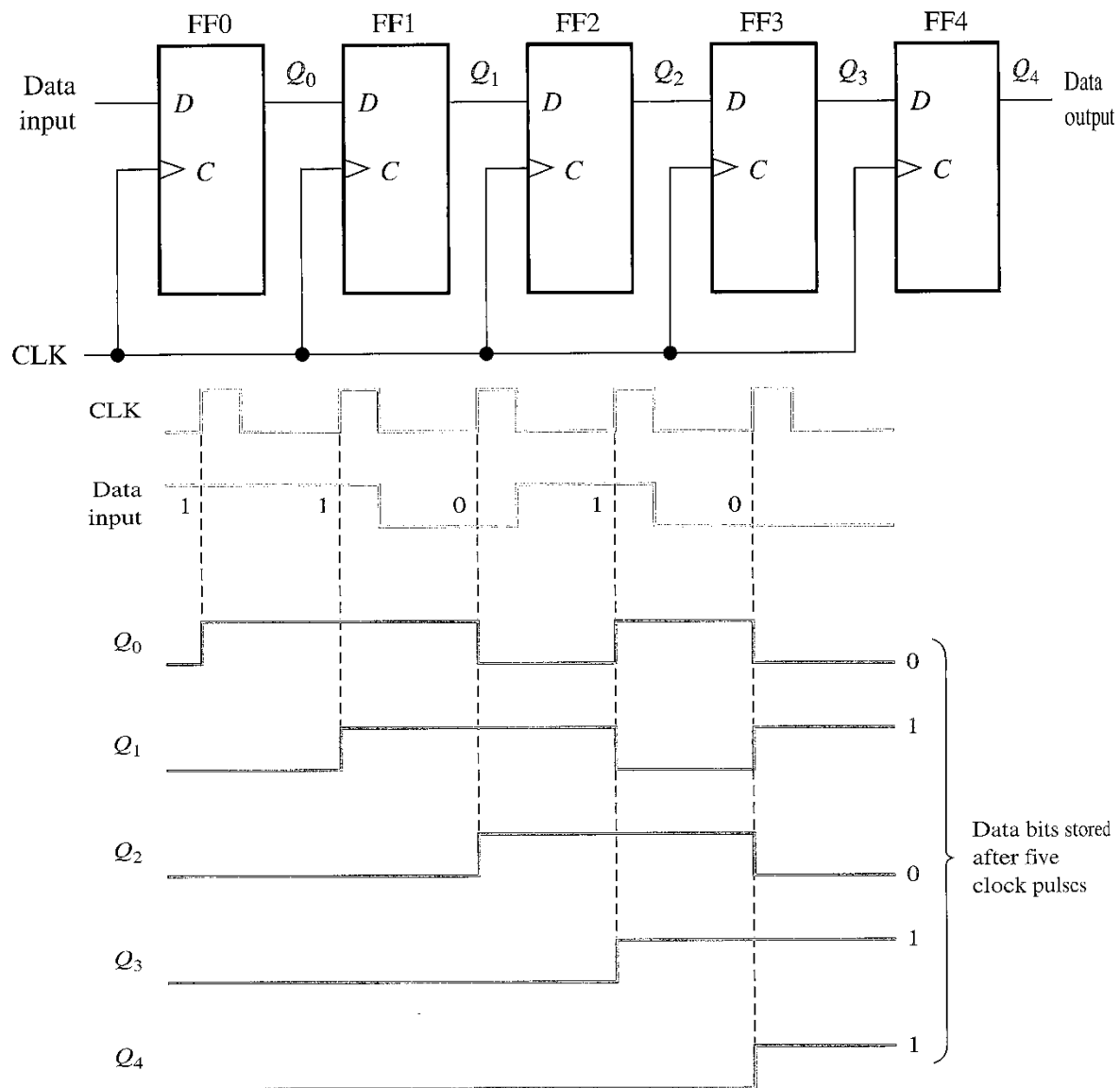
(f) Rotate right



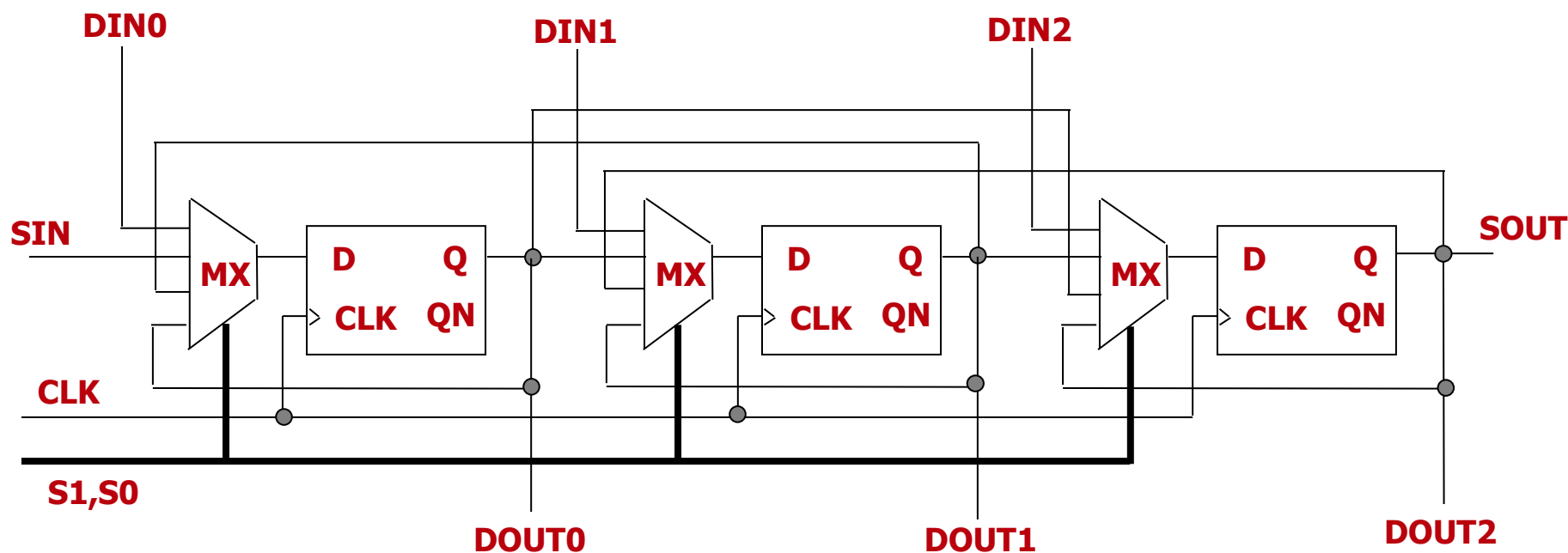
(g) Rotate left

- Příklad posuvného registru

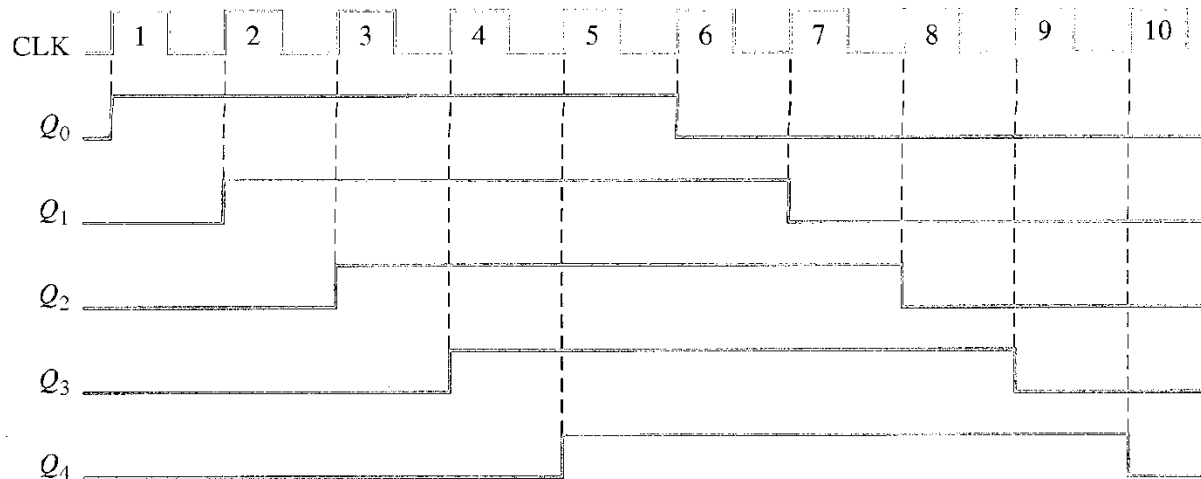
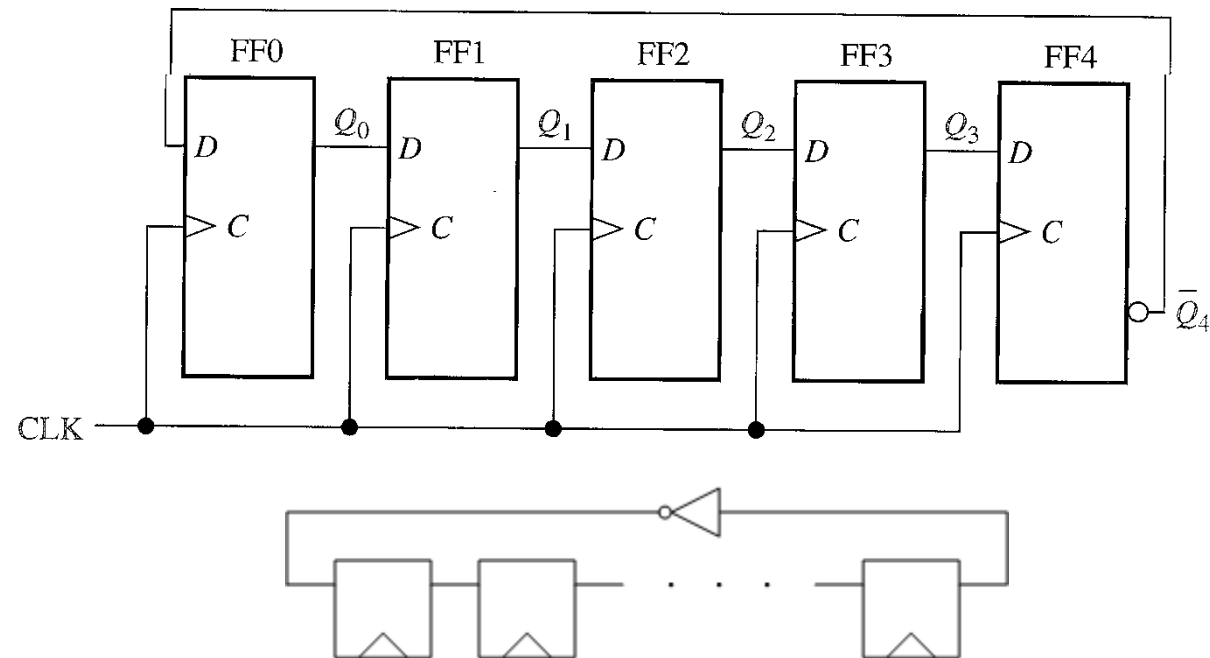
- Serial In
- Shift Right
- Serial Out



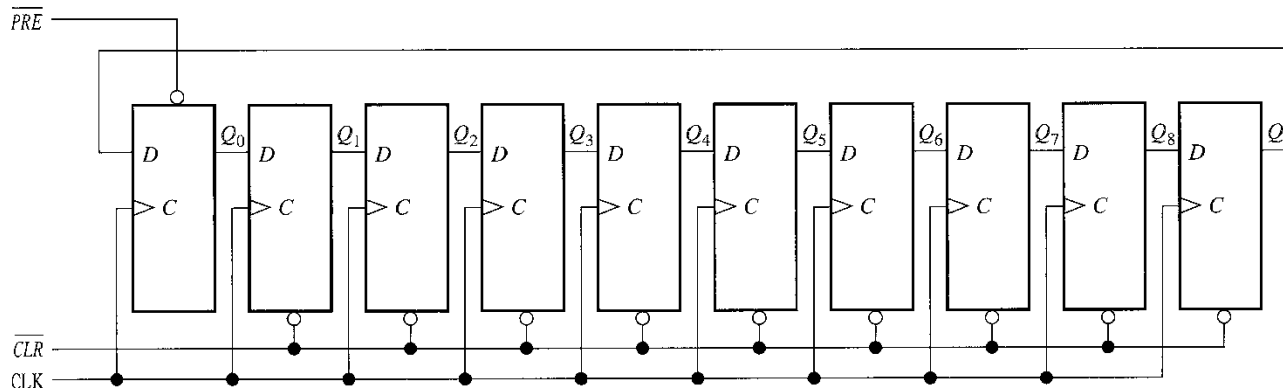
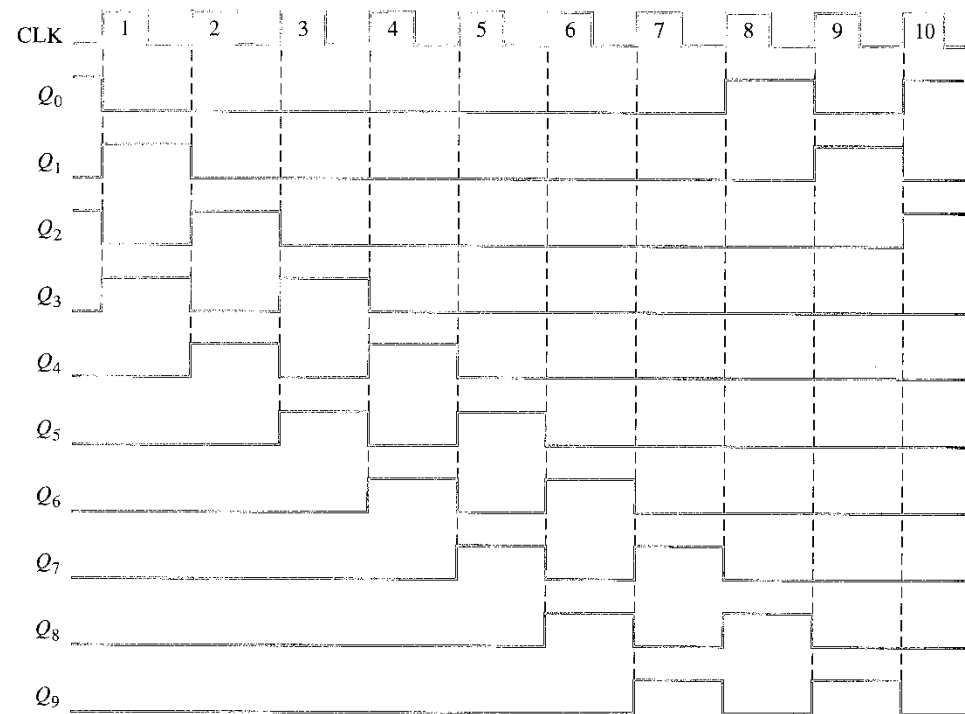
- Signály
 - DIN[2:0] – Paralel Data In
 - DOUT[2:0] – Paralel Data Out
 - SIN – Serial Data In
 - SOUT – Serial Data Out
 - S0,S1 – výběr funkce
- Posuv vlevo, vpravo, paralelní uložení informace a zapamatování stavu
 - S1=0, S0=0 – Paralel Load
 - S1=0, S0=1 – Shift Right
 - S1=1, S0=0 – Rotate Left
 - S1=1, S0=1 – Hold



- Čítá v Johnsonově kódu
 - Negovaný výstup posuvného registru je přiveden zpět na vstup

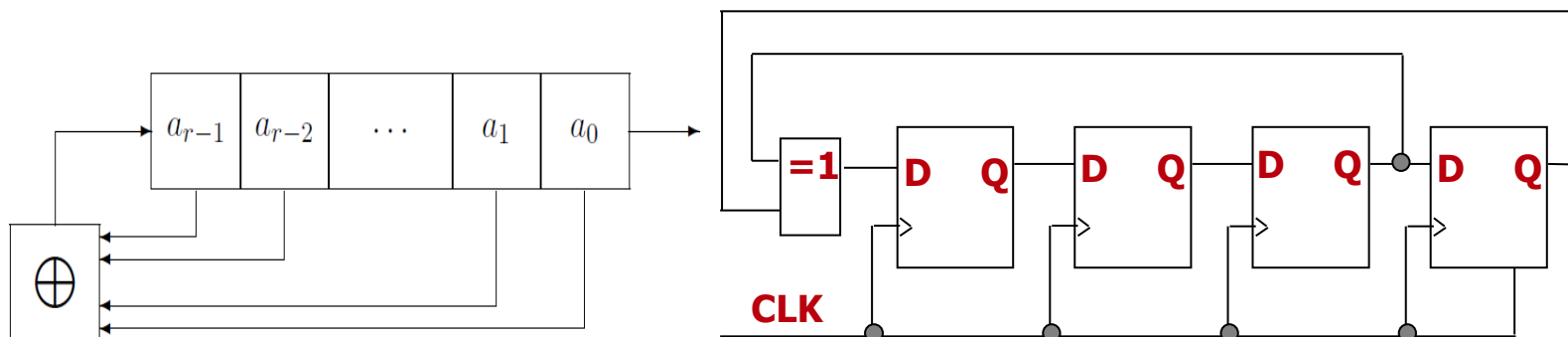


- Použití jako kruhový čítač
 - Čítá v kódu 1 z n (one-hot)
 - Musí být inicializována počáteční hodnota prvního KO
 - Příklad desetibitového kruhového čítače



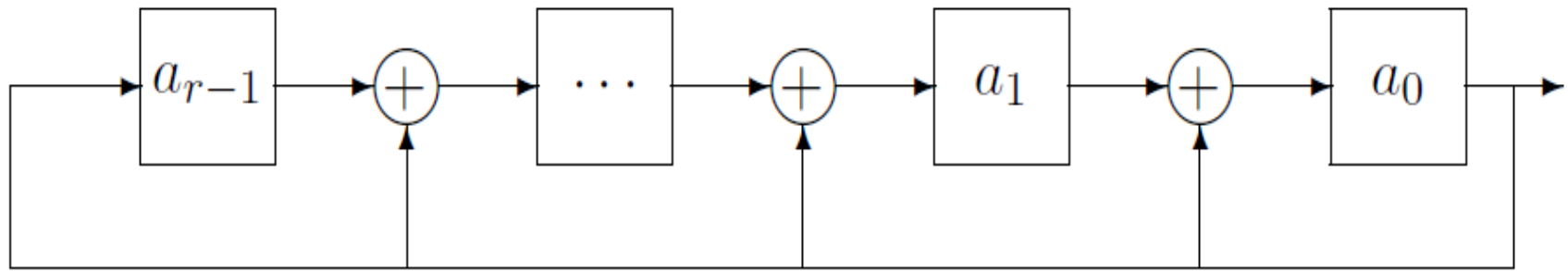
- Linear Feedback Shift Register (LFSR)
 - Posuvný registr, jehož vstup je lineární funkcí jeho stavu
 - Výhoda – jednoduchá kombinační logická síť ve zpětné vazbě = rychlost
 - Použití např. pro generování pseudonáhodných posloupností
- Lineární funkce
 - Nejčastěji generována pomocí XOR hradel
- Příklad generátoru „pseudonáhodné“ posloupnosti
 - Obecné schéma

4bitový LFSR

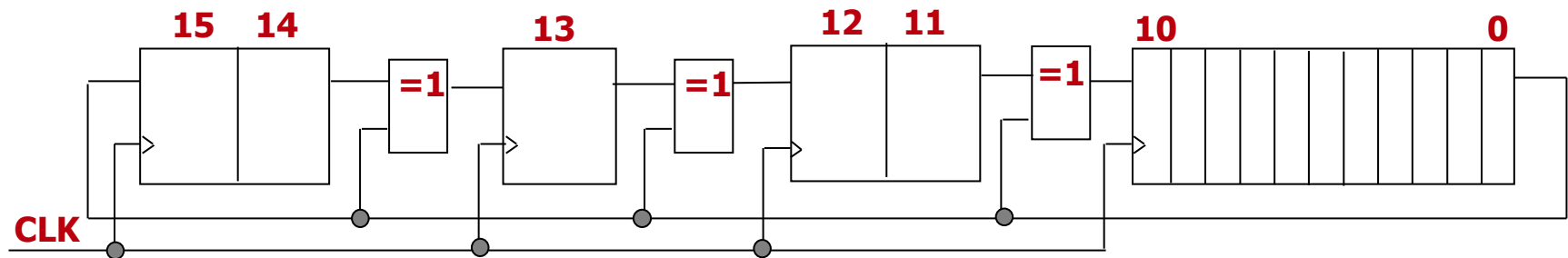


Posloupnost
0111
0011
0001
1000
0100
0010
1001
1100
0110
1011
0101
1010
1101
1110
1111

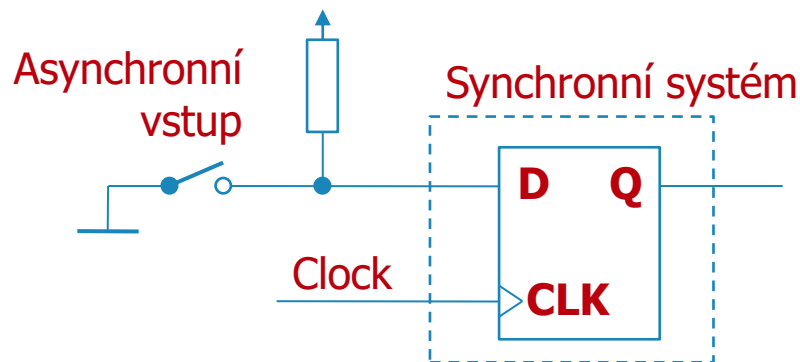
- Příklad LFSR
 - Obecné schéma



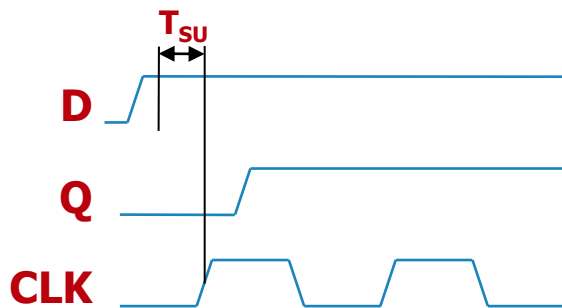
- 16bitový LFSR



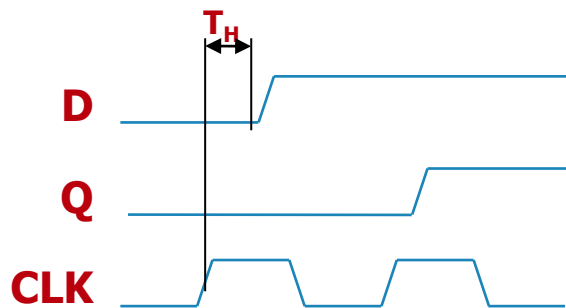
- V případě asynchronních vstupních signálů nelze zaručit, že budou dodrženy T_{SU} (setup) a T_H (hold) časy klopných obvodů, které jsou synchronizovány interními hodinami
 - Může nastat tzv. metastabilní stav - výstup Q KO bude v nedefinované logické úrovni (ani L, ani H) po nedefinovanou dobu



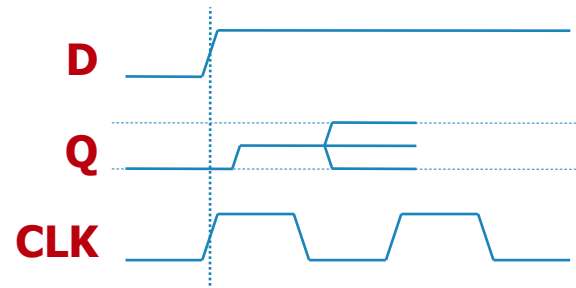
- T_{SU} je dodržen



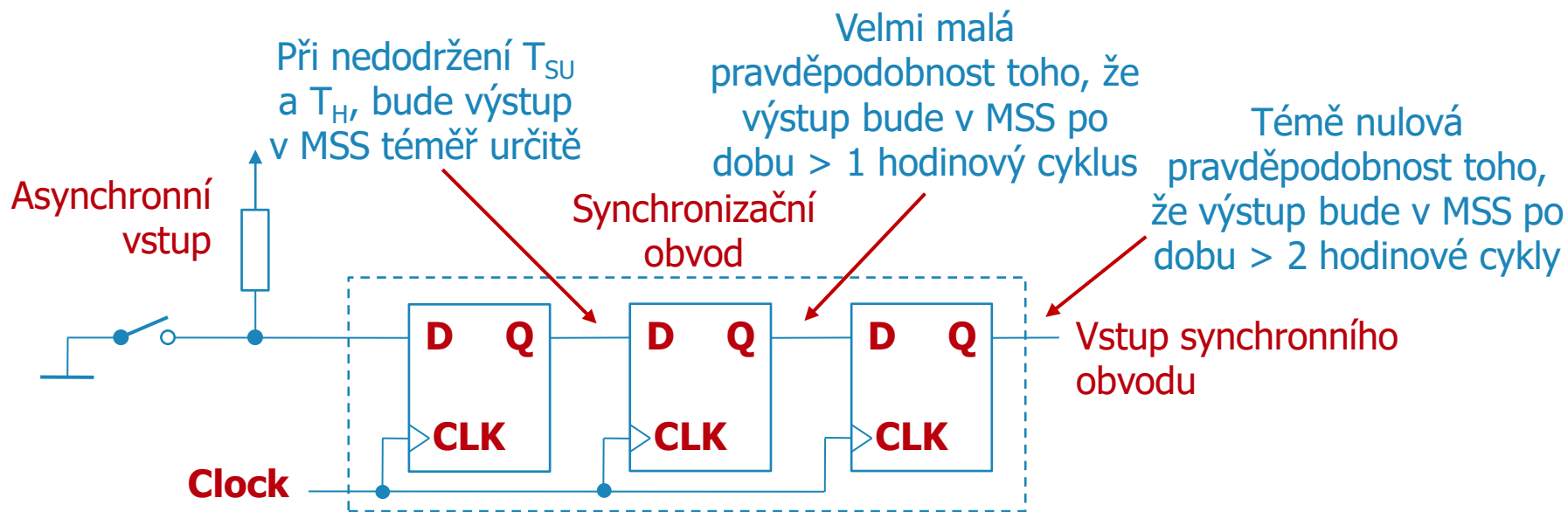
- T_H je dodržen



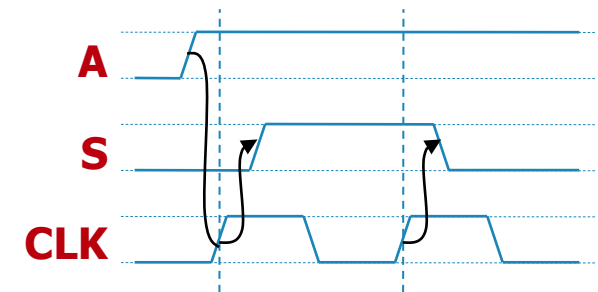
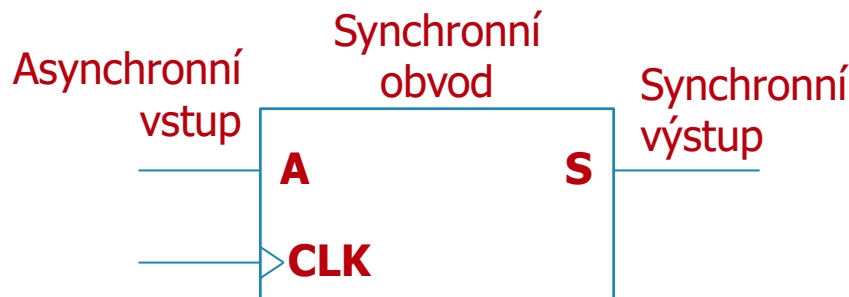
- Metastabilní stav



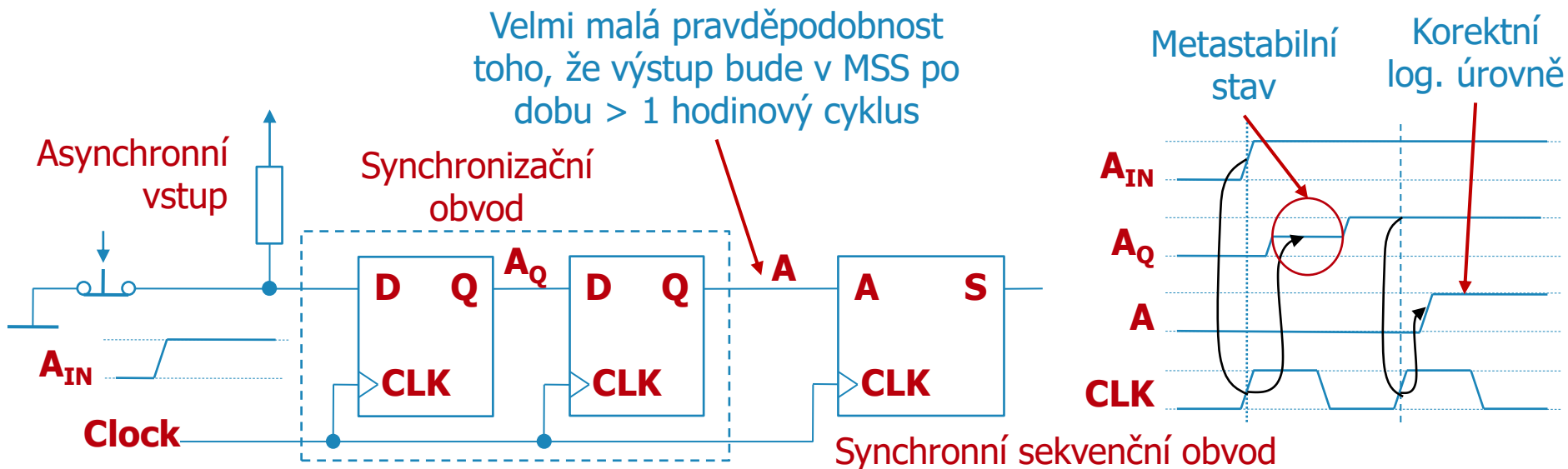
- Metastabilnímu stavu (MSS) nelze zabránit!
 - Lze jen omezit dobu jeho trvání
 - Pokud mají log. členy velké zesílení, zkrátí se doba, po kterou je obvod v metastabilním stavu
- Je třeba s ním počítat při návrhu
 - Jeho vliv lze omezit vhodným vstupním synchronizačním obvodem
 - S počtem synchronizačních registrů se zmenšuje pravděpodobnost, že se MSS projeví, roste však zpoždění vstupní události



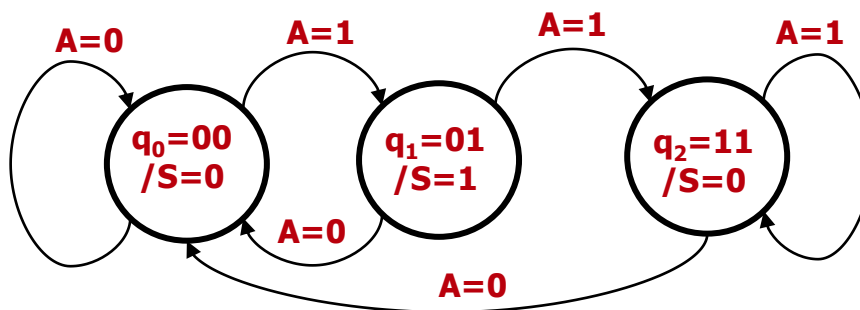
- Příklad: „detektor kladné hrany“ signálu
 - Navrhněte synchronní automat, který čeká na vstupní událost - přechod z log. 0 do log. 1 (kladná hrana)
 - Na základě detekce kladné hrany obvod vygeneruje kladný puls dlouhý jednu periodu hodinového signálu
 - Předpokládejme, že vstupní událost je asynchronní vůči navrhovanému synchronnímu automatu (není v žádné relaci s jeho hodinovým signálem) a že trvá nejméně jednu periodu hodinového signálu
- Detektor hrany
- Časový diagram



- Synchronizační obvod
 - Omezuje pravděpodobnost toho, že se metastabilní stav projeví na vstupu synchronního automatu
 - Výstupem je zasynchronizovaná vstupní asynchronní událost
 - Vstupní událost je zpožděna o 2 takty hodinového signálu
- Synchronní obvod
 - Konečný automat detekující kladnou hranu
 - Zkusíme implementovat jako Mooreův i Mealyho konečný automat

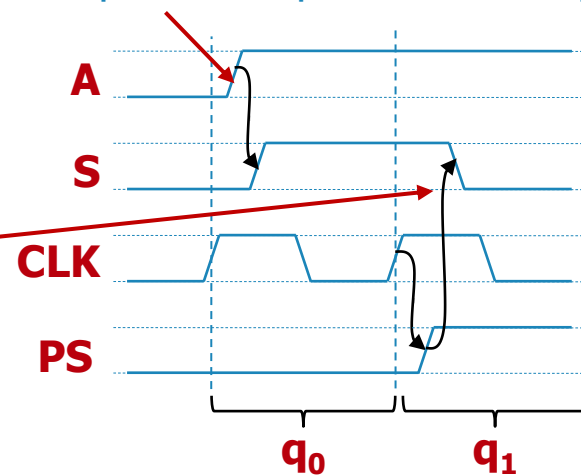
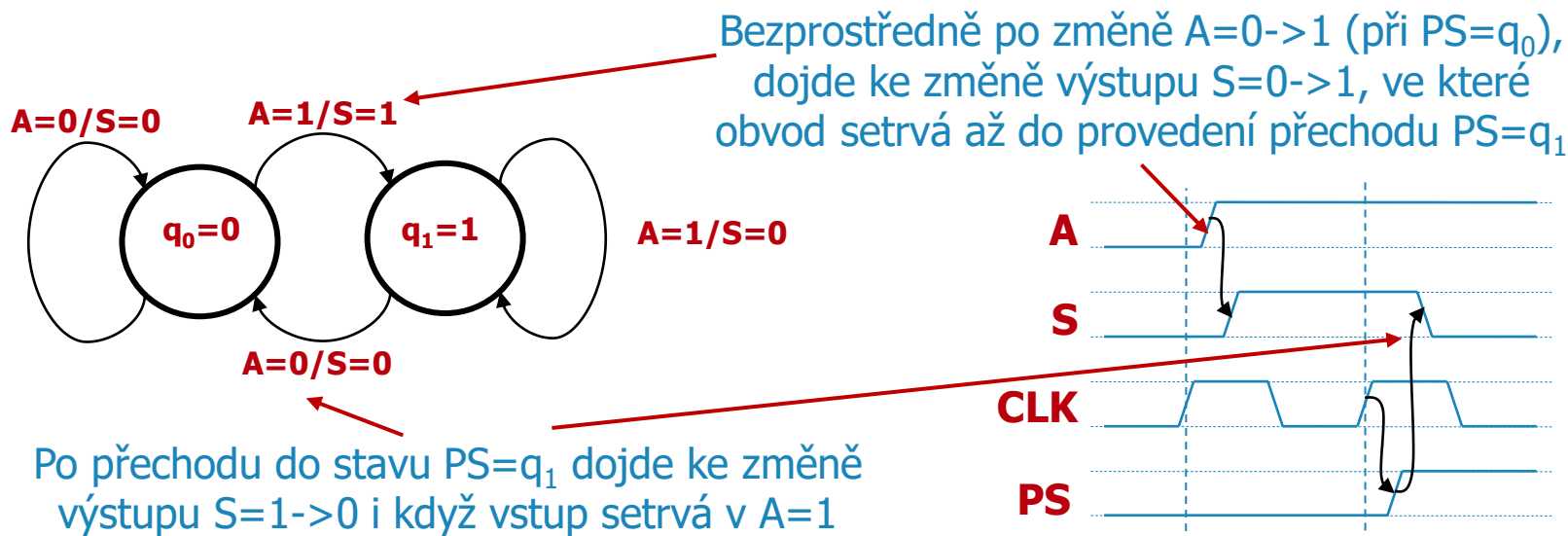


- Mooreův konečný automat (syntéza se bude se probírat později)
 - Kódování stavů je výhodné volit tak, aby byla implementace co nejjednodušší



Současný stav PS		Vstup A	Následující stav NS		Výstup Mealy	Výstup Moore S
Název	Kód Q1,Q0		Název	Kód D1,D0		
q ₀	00	0	q ₀	00	-	0
q ₀	00	1	q ₁	01	-	0
q ₁	01	0	q ₀	00	-	1
q ₁	01	1	q ₂	11	-	1
q ₂	11	0	q ₀	00	-	0
q ₂	11	1	q ₂	11	-	0

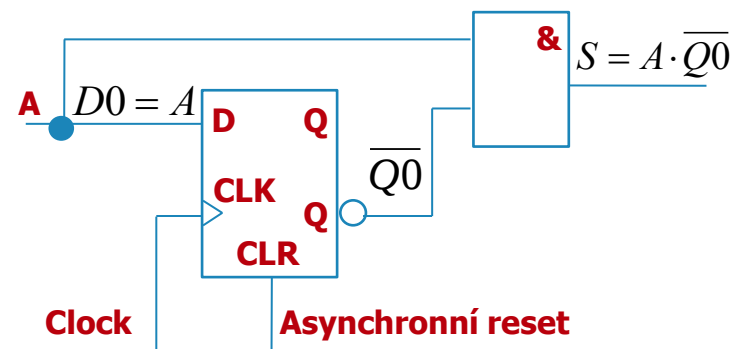
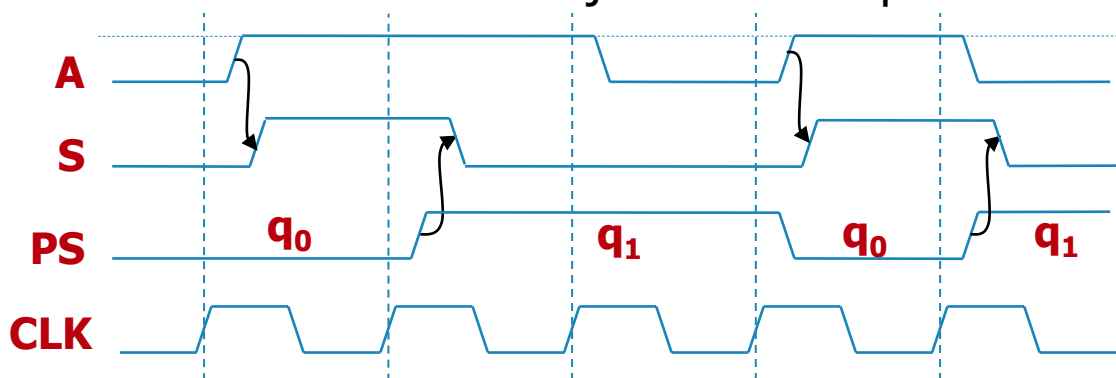
- Mealyho konečný automat (syntéza se bude se probírat později)
 - Mealyho automat může mít méně stavů než Mooreův



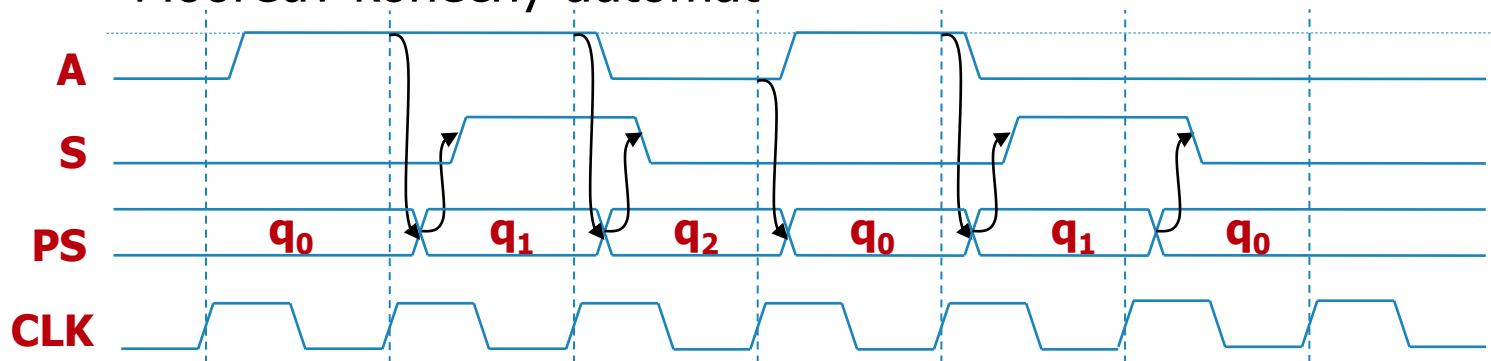
Současný stav PS		Vstup A	Následující stav NS		Výstup Mealy S	Výstup Moore
Název	Kód Q0		Název	Kód D0		
q_0	0	0	q_0	0	0	-
q_0	0	1	q_1	1	1	-
q_1	1	0	q_0	0	0	-
q_1	1	1	q_1	1	0	-

- Mealyho konečný automat

- Výstup je funkcí vstupu a stavu => výstup bezprostředně reaguje na vstup – bude platný o jednu periodu hodinového signálu dříve než v případě Mooreova výstupu (nežádoucí změna hodnoty na vstupu se může projevit na výstupu - vstup musí být stabilní - synchronizován)
- Má méně stavů - jednodušší implementace



- Mooreův konečný automat



- V software (SW)
 - Výpočet běží na univerzálním procesoru
 - Datové struktury
 - Řídicí struktury – sekvence, rozhodování, iterace
- V hardware (HW)
 - Specializované obvody pro konkrétní úlohu
 - Datové struktury – registry, paměti apod.
 - Řídicí struktury – řadič řídí funkční jednotky a datovou cestu
 - Urychlení, nižší cena a spotřeba pro vhodnou třídu aplikací
 - Ne univerzální stroj, ale aplikačně specifický HW – implementuje se jen to, co je nezbytně třeba pro danou funkci
- Limitující faktory
 - Výkonnost, latence, příkon, kapacita paměti, cena a rychlost návrhu atd.

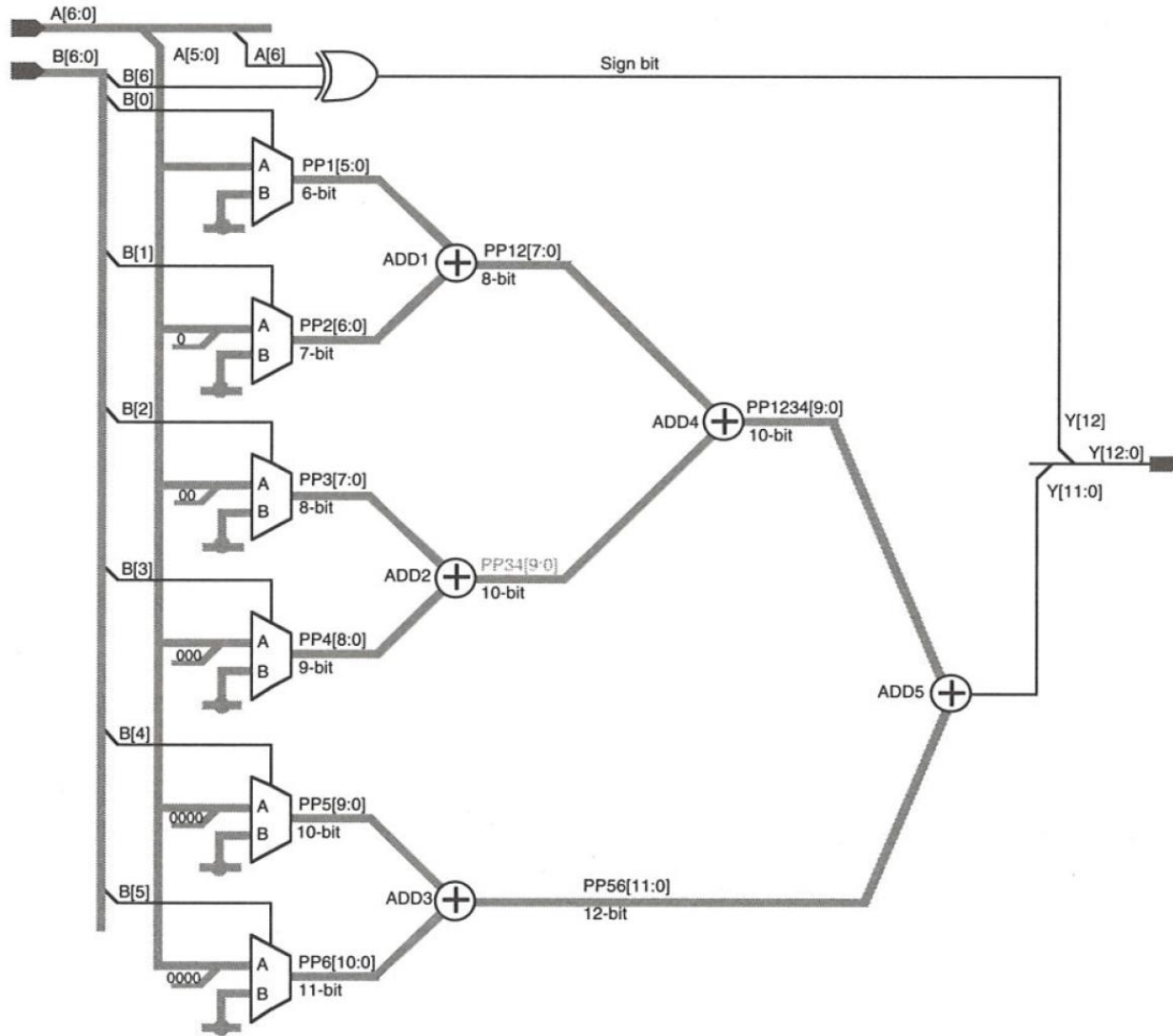
- Návrh SW je efektivnější (cena, doba návrhu) než HW
 - Návrh a výroba HW jsou "drahé"
 - Složité - časově a finančně náročný návrh, nutno amortizovat výrobu atd.
 - Programování SW je "levné"
 - Uživatel pouze formuluje algoritmus, nezabývá se návrhem a výrobou HW systému
- Výpočet je v HW efektivnější (rychlost, příkon) než v SW
 - Lze realizovat speciální kódování dle potřeby dané úlohy
 - Příklad: aritmetické operace v kódu zbytkových tříd jsou extrémně rychlé
 - Lze realizovat speciální výpočetní jednotky dle potřeby dané úlohy – např. rychlá Fourierova transformace (FFT)
 - Paralelní zpracování (násobné výpočetní jednotky)
 - Zřetězené (pipeline) zpracování, atd.

- Kombinační obvod
 - Logická síť bez paměťových prvků (registrů, zpětných vazeb...)
 - Např. ALU, paměťový dekodér...
- Sekvenční obvod
 - Konečný automat s relativně jednoduchou či žádnou kombinační sítí pro realizaci přechodové či výstupní funkce
 - Např. čítače, posuvné registry, detektor nástupné hrany...
- Kombinační + sekvenční obvod = řadič + datová cesta
 - Typická realizace složitějších obvodů
 - Složitější kombinační síť realizující nějakou funkci (ALU...)
 - Např. mikroprocesor...
- Zřetěžené zpracování (pipeline)
 - Kombinační obvod je rozdělen na stupně, které jsou propojeny přes registry
 - Pro urychlení výpočtu funkce realizované kombinačním obvodem

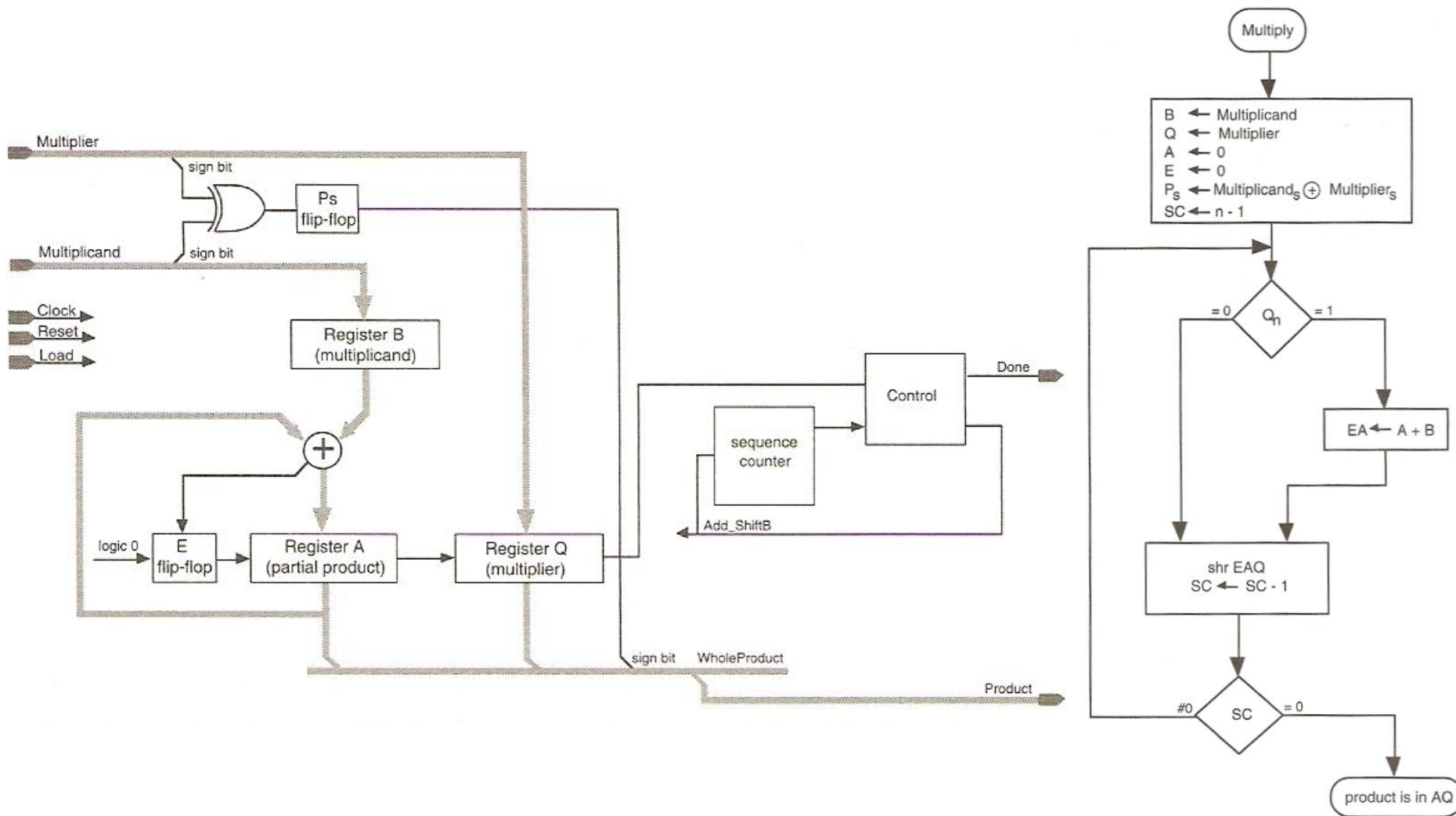
- Sekvenční sčítačka

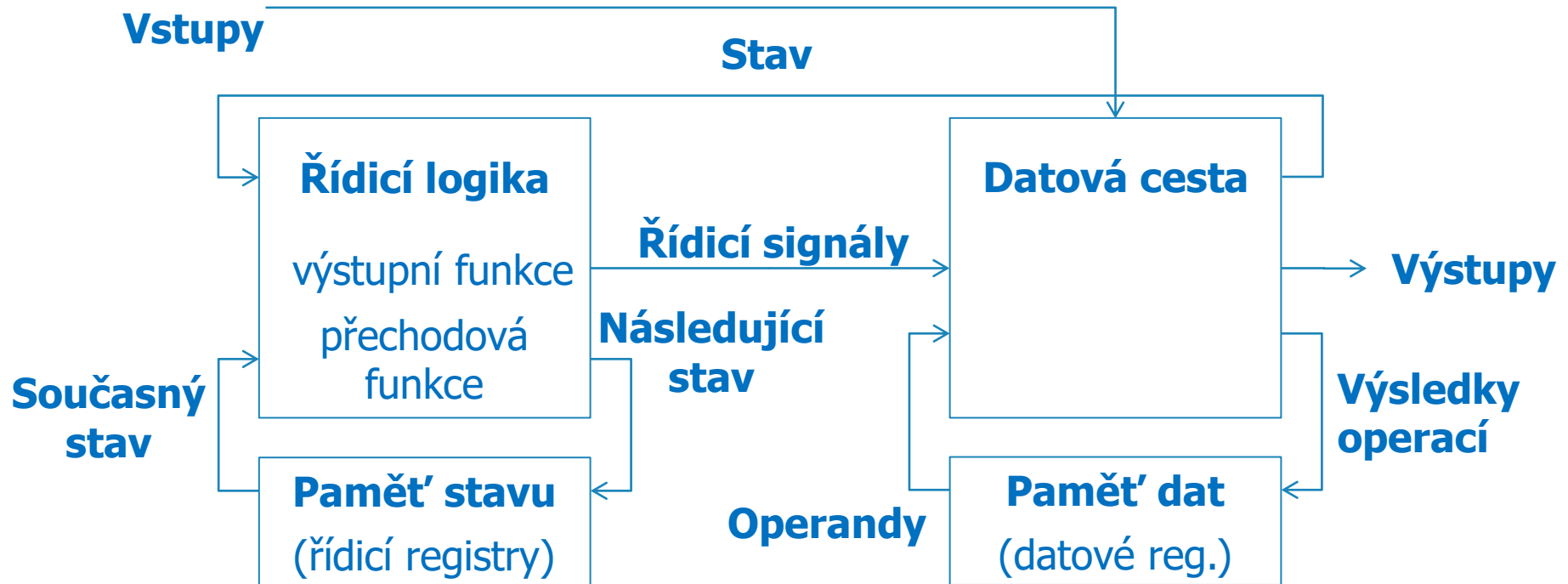


- Algoritmus Shift & Add - čísla ve dvojkovém doplňku



- Algoritmus – Shift & Add





- Řídicí logika (kombinační logická síť)
 - Může být implementována pomocí paměti (např. ROM, viz dále)
 - Chování automatu může tedy být naprogramováno
- Datová cesta
 - Typicky z hradel - implementace pomocí paměti nemusí být efektivní (viz např. násobička)

- Aplikačně specifické výpočetní systémy
 - Funkce je optimalizována pro danou (specifickou) funkci
 - Vysoká efektivita výpočtů (velká výkonnost, nízké energetické nároky)
 - Složitý návrh a výroba = vysoká cena (vyplatí se při hromadné výrobě)
 - Lze modifikovat (programovat, konfigurovat) jen v omezené míře
- Univerzální výpočetní systémy (počítače)
 - Určeny pro obecné použití (univerzální)
 - Amortizace návrhu a výroby výpočetního stroje
 - Mohou být programovány = levný návrh aplikací (software)
 - Nižší efektivita výpočtů oproti aplikačně specifickým systémům
 - Dnes jsou často vestavěny do větších systémů (anglicky Embedded Systems), kde vykonávají specifickou funkci

• Plnička lékových

- pásový dopravník
- plnicí jednotka
 - klapka
 - senzor
- řídicí jednotka
 - klávesnice
 - kódér klávesnice
 - registry
 - dekodéry displejů
 - kódéry
 - komparátor
 - sčítačka
 - multiplexor
 - displeje
- kontrolní jednotka
 - na vzdáleném pracovišti
 - kontrola celkového počtu pilulek

