Návrh číslicových systémů (INC)

Jiří Matoušek, Otto Fučík, Jan Kořenek

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

Použitá literatura

N. Frištacký, M. Kolesár, J. Kolenička a J. Hlavatý: „Logické systémy", SNTL Praha, 1986
M. Eysselt: „Logické systémy", SNTL Praha, skriptum VUT v Brně, 1985
J. F. Wakerly: „Digital Design. Principles and Practices", Prentice Hall, ISBN 0-13-769191-2, 2000
V. P. Nelson, H.T.Nagle, B.D.Carroll, J.D.Irwin: „Digital Logic Circuit Analysis & Design", ISBN 0-13-463894-8, 1995
T.L.Floyd: „Digital Fundamentals", Prentice Hall, ISBN 0-13-080850-4, 2000
D.J Smith: „HDL Chip Design", ISBN 0-9651934-3-8, 1996

Optimalizace parametrů obvodu

Návrh = algoritmus + syntéza

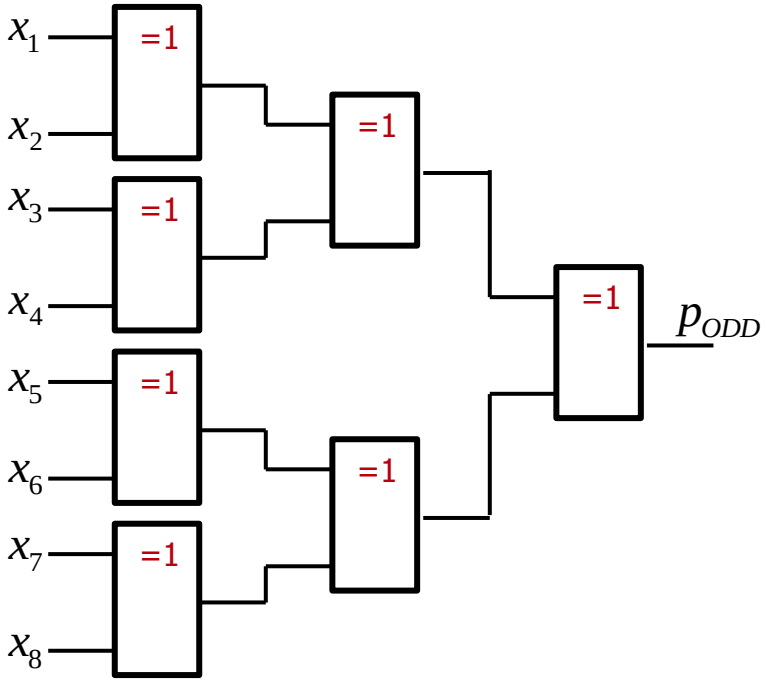
- Návrh = tvorba algoritmu a jeho implementace (syntéza)
- Algoritmus
  - Intuitivně – postup, který nás dovede k řešení úlohy
  - Formálně – „Přesně definovaná konečná posloupnost příkazů (kroků), jejichž prováděním pro každé přípustné vstupní hodnoty získáme po konečném počtu kroků odpovídající výstupní hodnoty“ [z kurzu Základy programování]
- Syntéza
  - Z formálního popisu algoritmu je třeba vytvořit výpočetní strukturu (logický obvod), která jej implementuje
  - Tvorba vhodné struktury se dnes provádí do značné míry automatizovaně
- Existuje mnoho výpočetních struktur, které implementují daný algoritmus (vícenásobná realizace)
  - Jejich vlastnosti určují rychlost výpočtu, příkon, rozměry, atd.

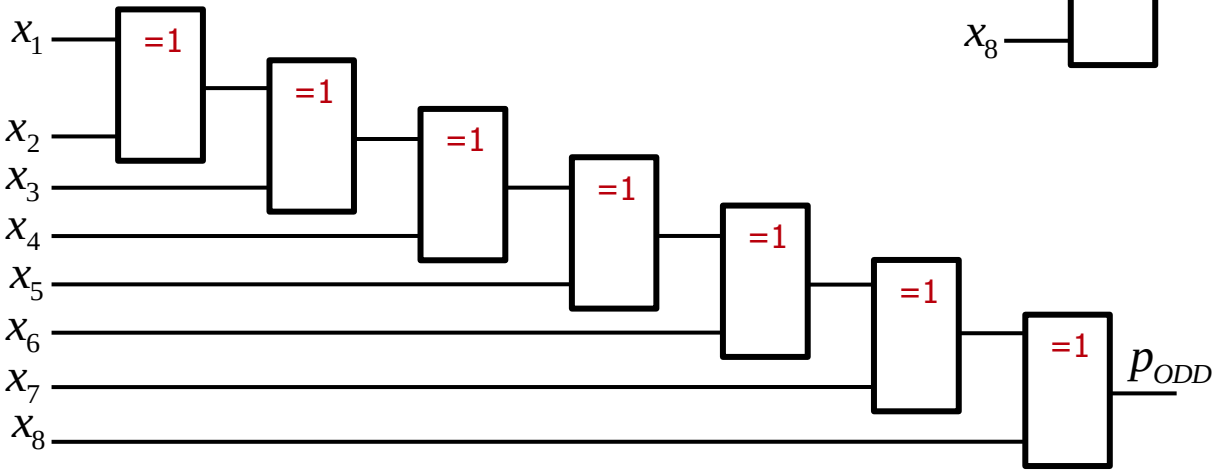
Obsah

- Doposud představené optimalizační metody
- Nejčastější kritéria optimalizací
- Optimalizace časování
- Úvod do optimalizace příkonu

1) Architektura

- Paralelní zapojení
  - (Často) větší počet zdrojů
  - Rychleji dostupný výsledek
- Sériové zapojení
  - (Často) menší počet zdrojů
  - Delší doba zpracování
- Příklad: parita





$$p_{ODD}(x_1,\ldots,x_8) = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8$$

2) Kódování informace

- Kódování = přidělení jisté reprezentace konkrétní informaci
  - Volba kódu výrazně ovlivňuje vlastnosti implementace příslušného systému (netriviální úloha)
- Problematika zahrnuje
  - Počet potřebných bitů (množství komponent – cena a spolehlivost systému)
  - Rychlost manipulace s bity (výkonnost systému)
  - Energetické nároky (změny hodnot odebírají nejvíce energie)
  - Délka (fixní, proměnná)
  - Čísla (bez a se znaménkem, pevná a plovoucí řádová čárka)
  - Komprese (ztrátová, bezztrátová)
  - Šifrování, autorizace
  - Detekce, oprava chyb (redundance, dostupnost)
  - Odolnost proti rušení atd.

2) Kódování informace: stavy konečného automatu

- 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)
- Počáteční stav
  - Nutno volit s ohledem na jeho snadné vynucení (reset)
- Nepoužité (nevyužité) stavy
  - Sekvenční obvod může přejít vlivem např. rušení do nevyužitého stavu (neočekávané chování)
  - Pro omezení případného rizika může být třeba situaci ošetřit

2) Kódování informace: stavy konečného automatu

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

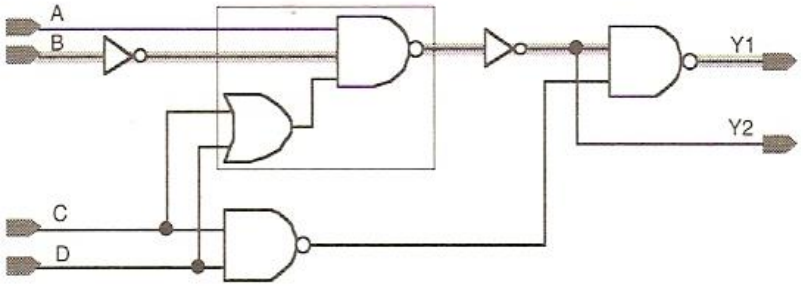
Binární	Grayův	Johnsonův	One-hot (1 z n)
0000	0000	00000000	000000000000000**1**
0001	0001	00000001	000000000000000**1**0
0010	0011	00000011	000000000000000**1**00
0011	0010	00000111	000000000000000**1**000
0100	0110	00001111	000000000000**1**0000
0101	0111	00011111	000000000**1**00000
0110	0101	00111111	000000000**1**000000
0111	0100	01111111	00000000**1**0000000
1000	1100	11111111	0000000**1**00000000
1001	1101	11111110	000000**1**000000000
1010	1111	11111100	00000**1**0000000000
1011	1110	11111000	0000**1**00000000000
1100	1010	11110000	000**1**000000000000
1101	1011	11100000	00**1**0000000000000
1110	1001	11000000	0**1**00000000000000
1111	1000	10000000	**1**000000000000000

3) Minimalizace: metody

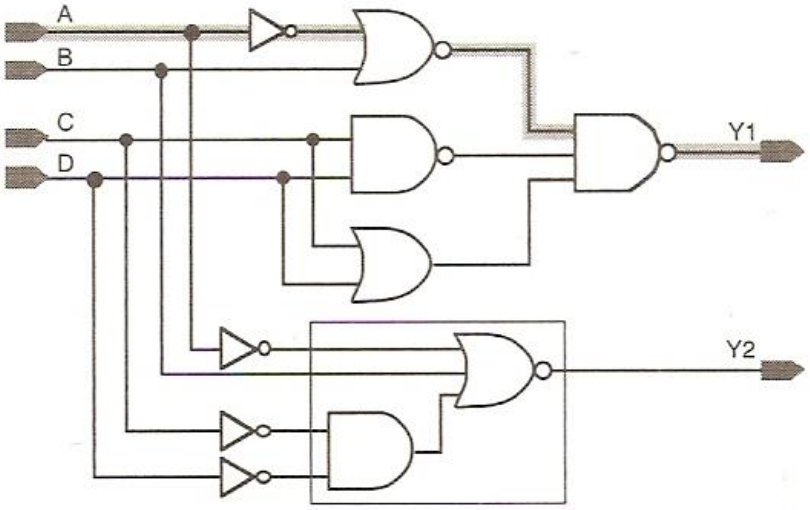
- Metody pro minimalizaci počtu logických členů
  - Algebraické
    - Postupnou aplikací axiomů a teorémů Booleovy algebry
  - Grafické
    - Jednotková krychle
    - Vennův diagram
    - Mapy (Svobodova, Karnaughova)
  - Algoritmické
    - Quine-McCluskey
    - Espresso atd.

3) Minimalizace - příklad

- Minimální plocha
  - Počet log. členů: 6
  - Plocha: 12
  - Zpoždění: 4 t_d



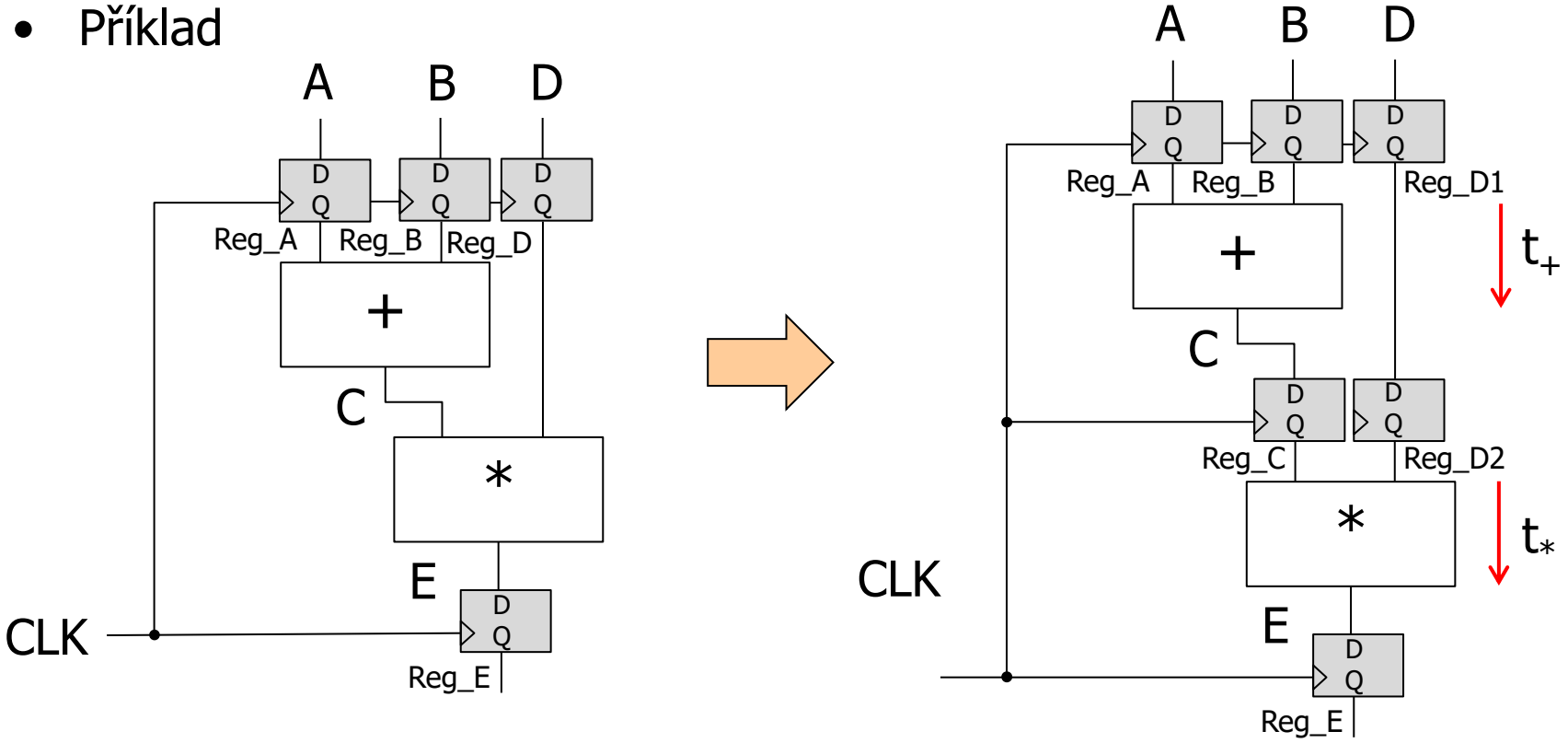
- Minimální zpoždění
  - Počet log. členů: 10
  - Plocha: 28
    Zpoždění: 3 t_d



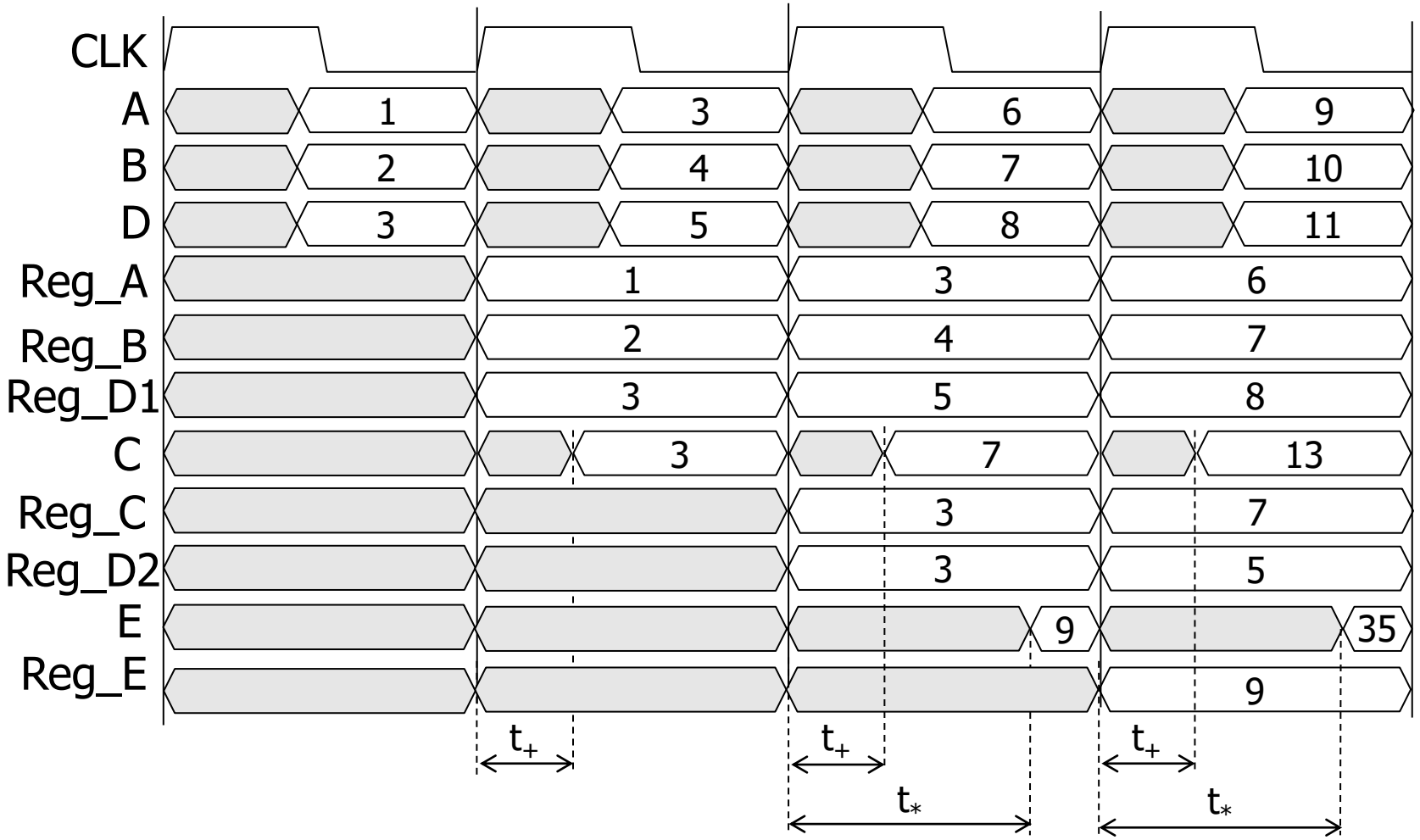
A	B	C	D	Y1	Y2
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	0

4) Zřetězené zpracování (pipelining)

- Vložením registrových stupňů mezi bloky kombinační logiky je možné složitější operaci rozdělit na více menších (rychlejších) částí
- Jednotlivé stupně zřetězení linky mohou současně (paralelně) zpracovávat oddělené části po sobě jdoucích dílčích operací
- Příklad



- Časový diagram

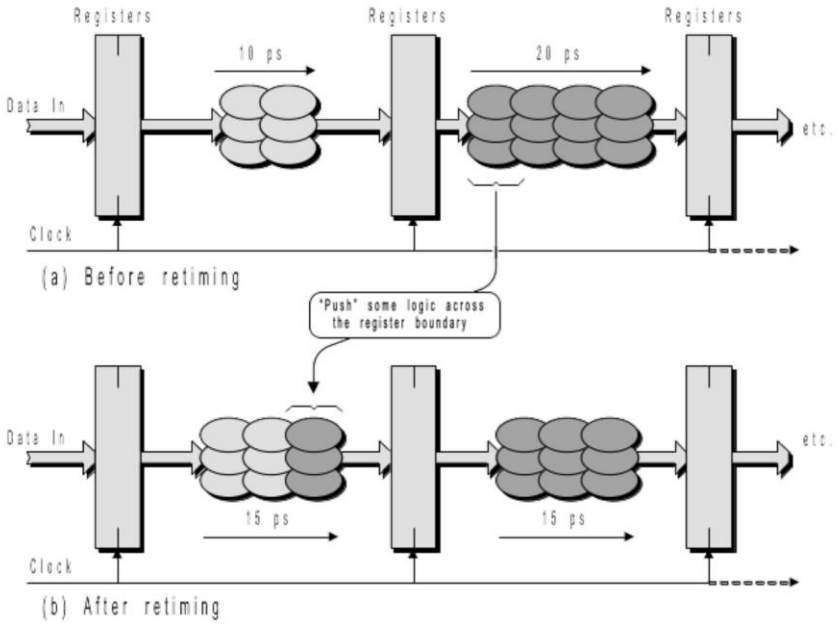


4) Zřetězené zpracování (pipelining)

- Poznámky:
  - Při vytváření stupně se musí registry vložit i doprostřed propojovacích vodičů (např. Reg_D2), jinak by výpočet nepracoval správně
- Důsledky:
  - Přidáním registrů dovnitř obvodu se dále prodlužuje jeho latence, tj. výsledek výpočtu je dostupný o takt později
  - Doba periody se zkracuje na $t=\max[t_+,t_*]$ + setup time + zpoždění vodičů, neboť jednotlivé příkazy (operace) se vykonávají nezávisle v oddělených stupních
  - Technika zřetězení je nejefektivnější, pokud je délka vypočtu v jednotlivých stupních vyvážená – vede na minimální délku periody tj. maximální rychlost výpočtu
- Problémy:
  - Techniku zřetězení nelze aplikovat na výpočty se zpětnou vazbou

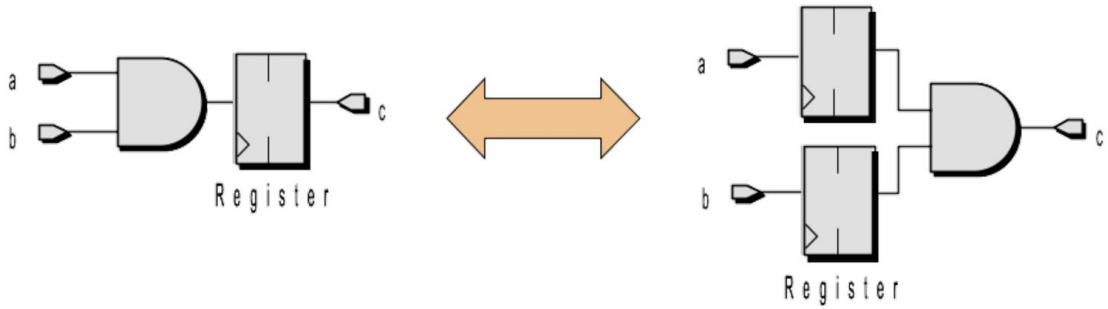
5) Retiming

- Metoda založena na přeuspořádání kombinační logiky mezi jednotlivými stupni zřetězené architektury
- Vede na vyvážení kombinační cesty mezi registry a možnost aplikovat vyšší pracovní frekvenci na celkový obvod
- Podle typu obvodu může metoda vést na zrychlení v řádu desítek procent výkonu
- Dostupná obvykle ve formě volitelné optimalizace v syntézních nástrojích



5) Retiming

- Základní pravidlo metody retiming
  - registr na výstupu kombinačního obvodu může být přesunut před tento prvek, pokud je aplikován na všechny jeho vstupy
  - podobným způsobem registr může být přesunut ze vstupu kombinačního obvodu na výstup, pokud je přesunut ze všech jeho vstupních portů
  - funkce obvodu zůstává v rámci této transformace zachována



Obsah

- Doposud představené optimalizační metody
- Nejčastější kritéria optimalizací
- Optimalizace časování
- Úvod do optimalizace příkonu

Kritéria optimalizace logických obvodů

- Plocha (area)
  - Část čipu, kterou obvod zabírá
  - Je ovlivněna řadou parametrů - počet log. členů (viz minimalizace), rozměry log. členů (technologie výroby, logický zisk), spoje
- Časování (timing)
  - Doba, za kterou se po změně vstupních proměnných ustálí výstupní proměnné obvodu
  - Je ovlivněno řadou parametrů - délka logické větve, logický zisk a zátěž log. členů, parazitní kapacity, délka spojů atd.
- Příkon (power)
  - Příkon obvodu ovlivňuje počet log. členů, pracovní frekvence, velikost napájecího napětí, parazitní kapacity, výrobní proces atd.
- Testovatelnost (test, scan)
  - Logické obvody je třeba navrhovat tak, aby bylo možno otestovat jejich správnou činnost

Použitelnost optimalizačních metod

	Architektura	Kódování informace	Minimalizace	Pipelining	Retiming
Plocha	Ano	Ano	Ano	Ne	Nepřímo
Časování	Ano	Ano	Nepřímo	Ano	Ano

- Optimalizace příkonu viz dále
- Testovatelnost a jiná kritéria nyní neuvažujeme

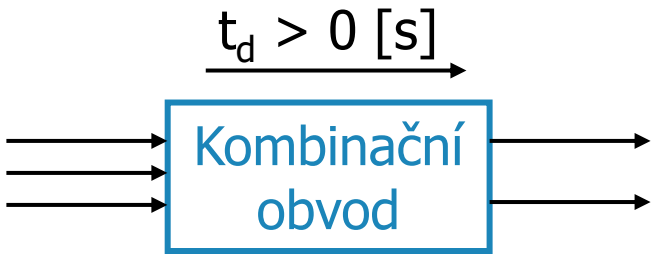
- Kódování informace, minimalizaci a retiming lze aplikovat i automatizovaně v rámci procesu syntézy
- Změnu architektury a pipelining naopak musí vývojář aplikovat ve vlastní režii

Obsah

- Doposud představené optimalizační metody
- Nejčastější kritéria optimalizací
- Optimalizace časování
- Úvod do optimalizace příkonu

Chování logických obvodů v čase

- Specifikace chování (behaviorální, funkce)
  - Definice hodnoty (logické úrovně) každého výstupu pro všechny kombinace vstupních hodnot (logických úrovní)



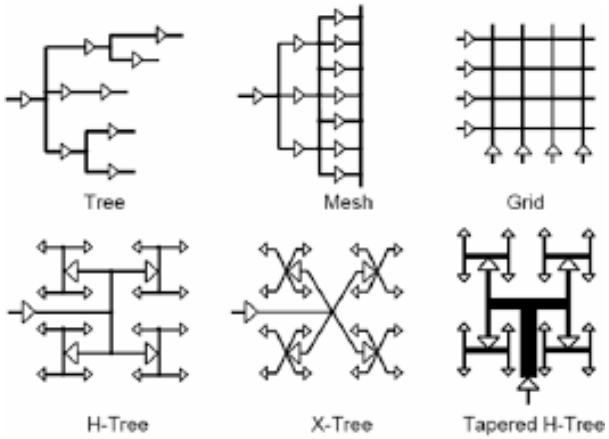
- Specifikace časování
  - Každý fyzicky realizovaný obvod má zpoždění t_d
    - Definováno např. jako nejhorší potřebná doba pro výpočet (vygenerování) platných logických hodnot výstupů od okamžiku, kdy na vstupech budou platné a stabilní logické hodnoty
  - Při návrhu se zpoždění často zanedbává ($t_d = 0$)
    - Zpoždění je třeba při implementaci obvodů zohlednit (rychlost výpočtů, vliv prostředí atd.)

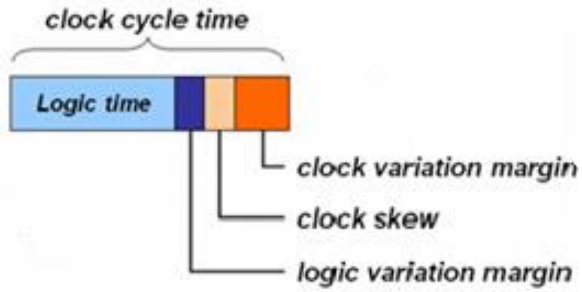
Synchronní sekvenční obvody

- 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 (hazardů) - díky tomu se hazardy nešíří dále do dalších stupňů KLS
  - Lze automatizovat syntézu logických systémů z popisu jejich chování na vysoké úrovni abstrakce (VHDL, Verilog atd.)

Synchronní sekvenční obvody: Časování

- 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 – největší problémem 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





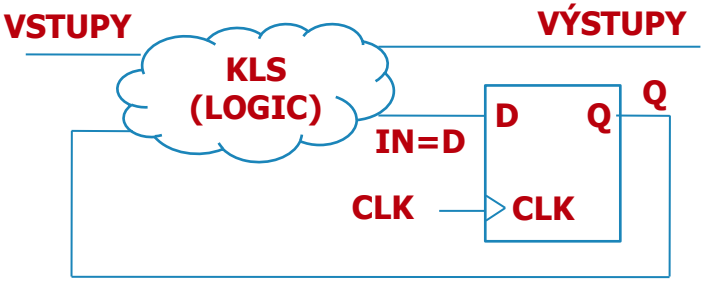
[Zdroj: Synopsys]

Synchronní sekvenční obvody: Časování

- Příklad: Kombinční 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
- Stejné pro sekvenční sítě bez i se zpětnou vazbou

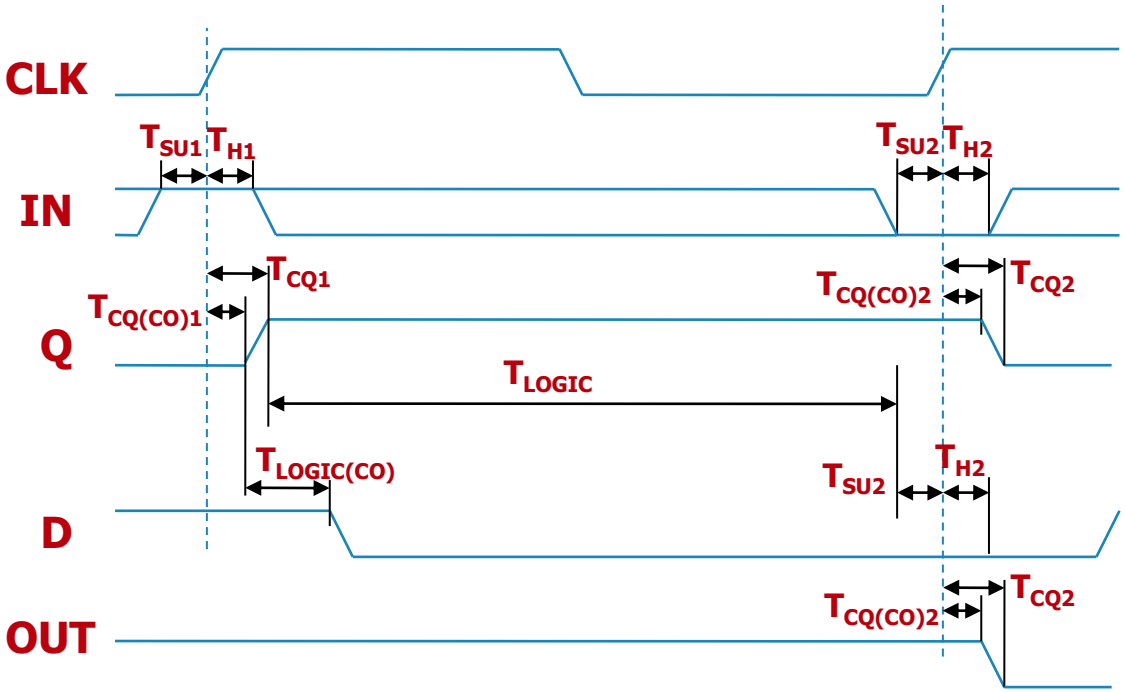
Sekvenční síť bez zpětné vazby

Sekvenční síť se zpětnou vazbou - konečný automat



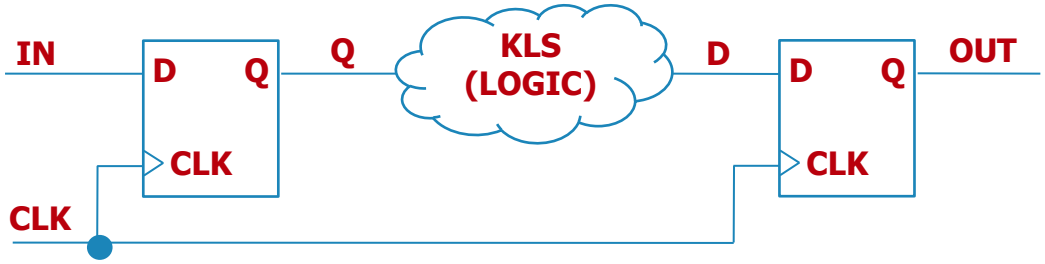
- Minimální perioda hodinového signálu
  - Pro jednoduchost předpokládáme, že parametry klopných obvodů v sekvenční síti jsou stejné ($T_{SU1}=T_{SU2}$ atd.)
  - Pro správnou činnost musí platit, že perioda hodinového je dostatečně dlouhá

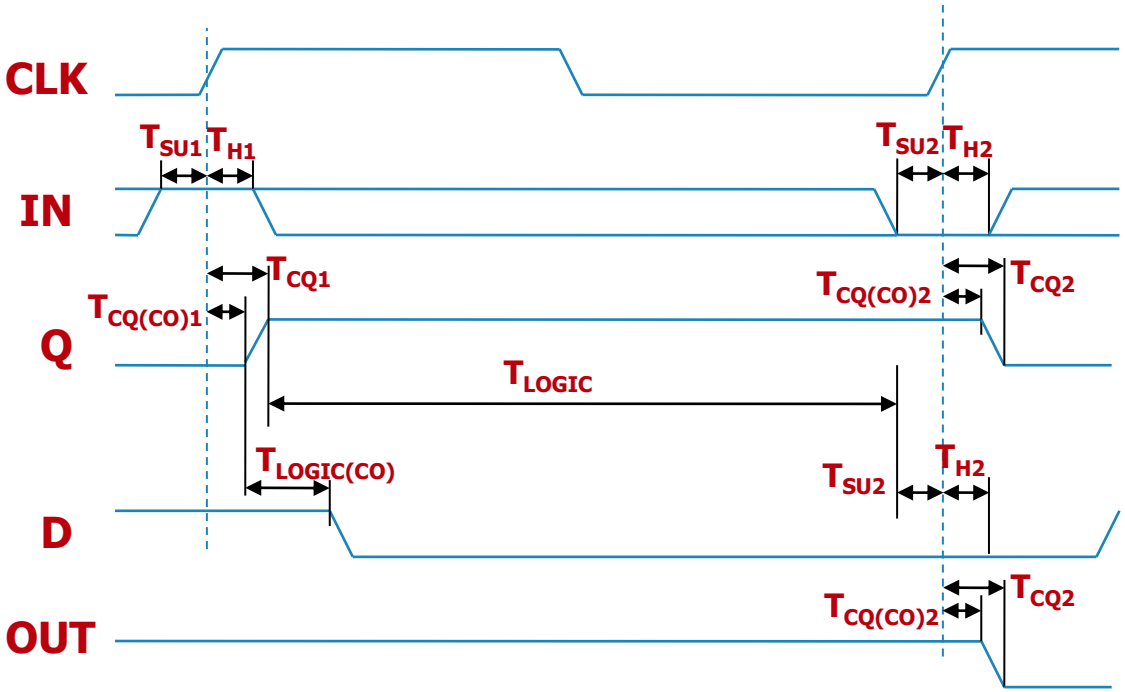
$$T_{CLK} > T_{CQ} + T_{LOGIC} + T_{SU}$$



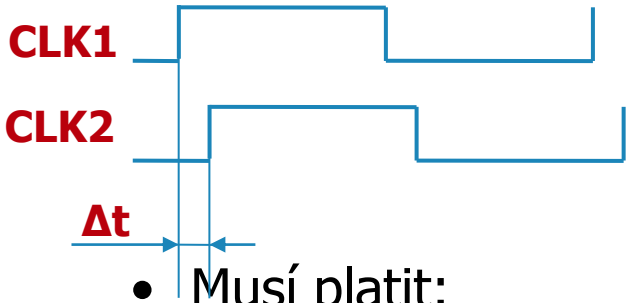
- Minimální zpoždění komponent
  - Pro správnou činnost musí být dodržen „hold time” klopných obvodů

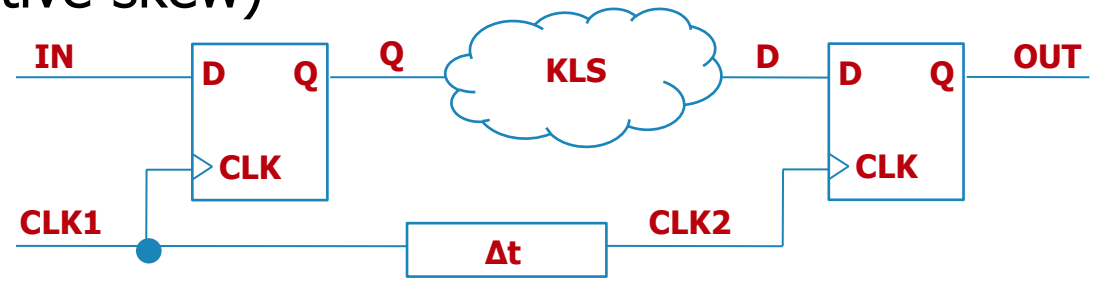
$$T_H < T_{CQ(CO)} + T_{LOGIC(CO)}$$





- Pozitivní zpoždění (positive skew)

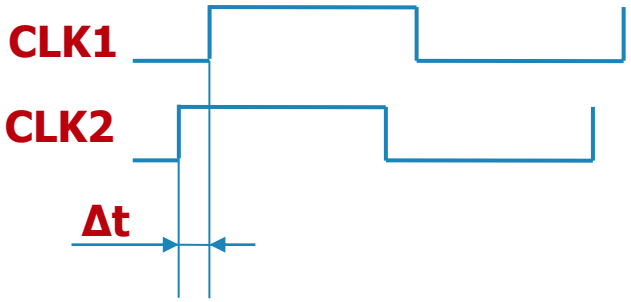


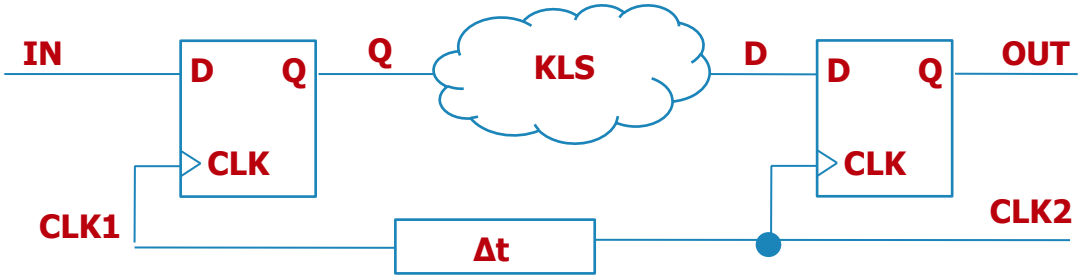


  - Musí platit:

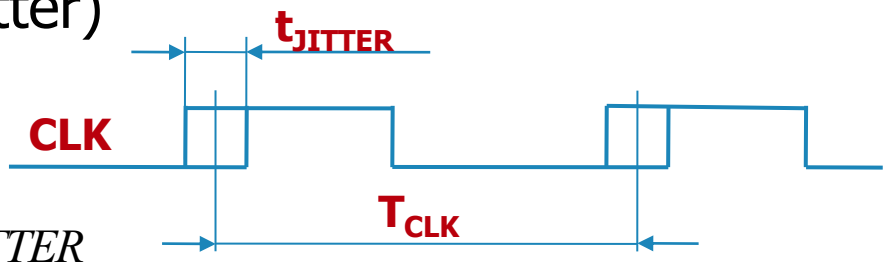
$$T_H + \Delta t < T_{CQ(CO)} + T_{LOGIC(CO)}$$

- Negativní zpoždění (negative skew)





- Nestabilita hodinového signálu (jitter)
  - Musí platit:

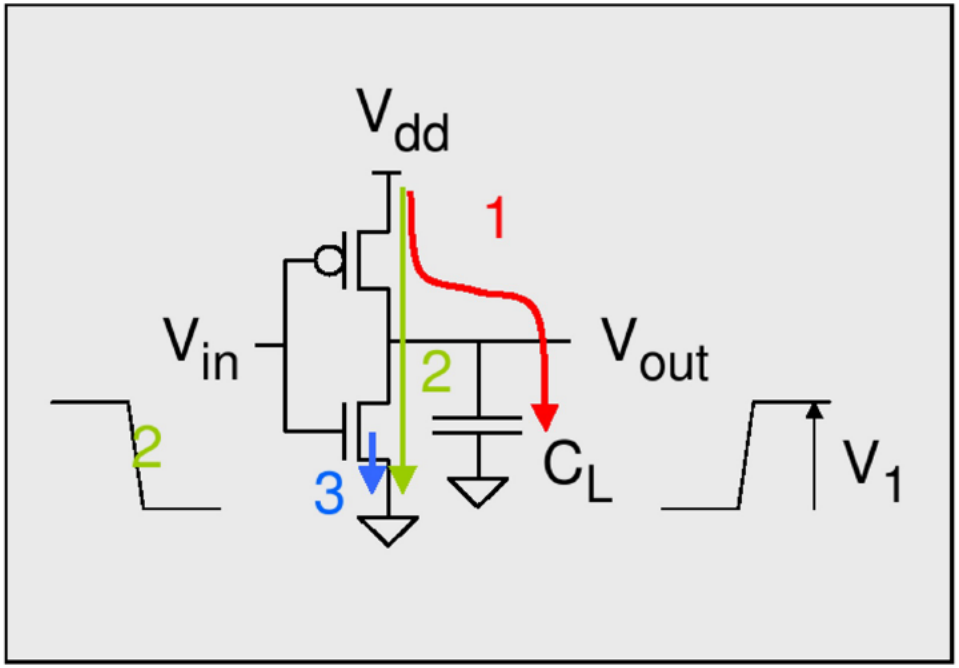


$$T_{CLK} > T_{CQ} + T_{LOGIC} + T_{SU} + 2 \cdot t_{JITTER}$$

Obsah

- Doposud představené optimalizační metody
- Nejčastější kritéria optimalizací
- Optimalizace časování
- Úvod do optimalizace příkonu

Příkon v technologii CMOS



- Dynamický / Aktivní příkon
  - Nabíjení a vybíjení parazitní kapacity
- Statický / Leakage příkon
  - Zbytkové proudy tranzistorů
- Příkon způsobený „zkraty“
  - Při přepnutí tranzistoru vzniká přímá cesta mezi VDD a GND

Dynamický/aktivní příkon

- Dynamický příkon

$$P_{dynamic} = \alpha \cdot C \cdot {V_{DD}}^2 \cdot f_{clk}$$

- Spotřeba energie

$$E_{dynamic} = \alpha \cdot C \cdot {V_{DD}}^2 \cdot S$$

- Zpoždění

$$\tau_c\,(U) = k' \cdot C_L \cdot \frac{V_{DD}}{(V_{DD} - V_T)^2}$$

- α – pravděpodobnost přepnutí logické úrovně hradla
- C_L – zátěžová kapacita
- V_{DD}– napájecí napětí
- V_T – prahové napětí
- f_{clk} – frekvence hodin
- s – počet taktů hodin

- Příklad: snížení napájecího napětí na polovinu
  - Příkon klesne 4x

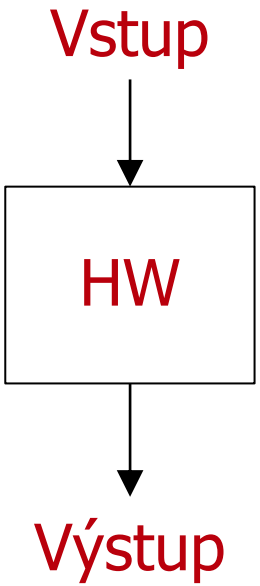
$$P_{(V_{DD}/2)} = \alpha \cdot f \cdot C \cdot (V_{DD}/2)^2 = 1/4 \cdot \alpha \cdot f \cdot C \cdot V_{DD}^{\ 2} = P/4$$

  - Zpoždění vzroste 2x

$$\tau_{(V_{DD}/2)} \approx \frac{1}{\frac{V_{DD}}{2}} = 2 \cdot \frac{1}{V_{DD}} = 2 \cdot \tau$$

CMOS technologie: Napětí, příkon a frekvence

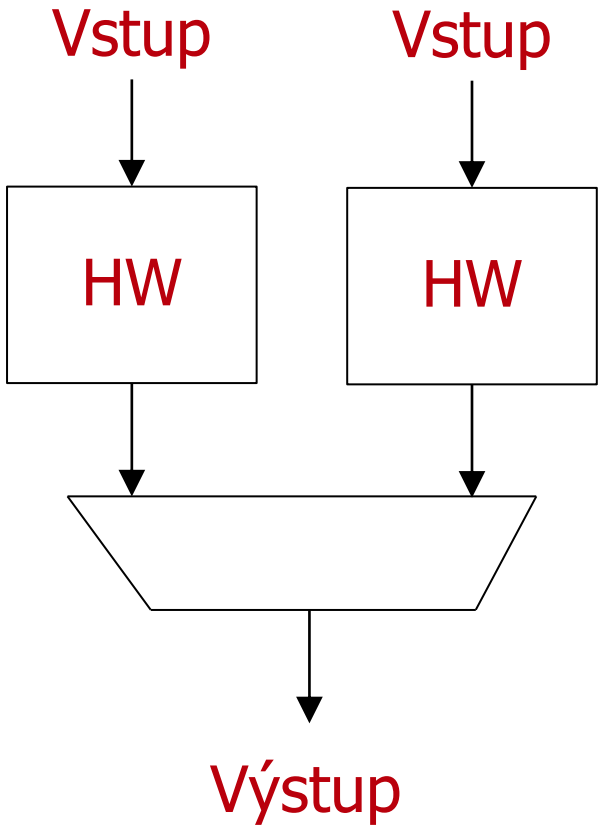
- Příklad - výpočet jednou výpočetní jednotkou
  - f = 1 GHz, V_{DD} = 2 V



$$P_1 = C \cdot 2^2 \cdot f = 4 \cdot C \cdot f$$

$$P_2 = 2 \cdot C \cdot 1^2 \cdot f/2 = C \cdot f = P_1/4$$

- Výpočet dvěma identickými jednotkami s polovičním napájecím napětím
  - f = 500 MHz, V_{DD} = 1 V



Metody redukce příkonu

- Dynamický příkon
  - Clock gating
  - Snížení napájecího napětí
  - Snížení frekvence hodin
- Statický příkon
  - Vytvoření domén s nezávislým napájením + power gating
  - Snížení počtu napájených zařízení
  - Použití vhodnější technologie