1. Hradlové sítě *(zapsali Jirka Fajfr a Ján Černý)*

Na této přednášce se budeme zabývat jednoduchým modelem paralelního počítače, totiž hradlovou sítí, a ukážeme si alespoň jeden efektivní paralelní algoritmus, konkrétně sčítání dvojkových čísel v logaritmickém čase vzhledem k jejich délce.

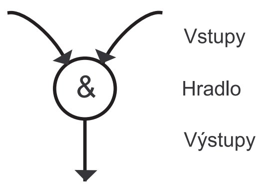
Hradlové sítě

Definice: *Hradlo* je zařízení, které počítá nějakou pevně danou funkci s k vstupy a jedním výstupem.

Příklad: Obvykle pracujeme s booleovskými hradly, ta pak odpovídají funkcím $f : \{0,1\}^k \to \{0,1\}$. Z nich nejčastěji potkáme:

- 0-vstupové: to jsou konstanty TRUE a FALSE,
- 1-vstupové: identita (ta je vcelku k ničemu) a negace (značíme $\neg$),
- 2-vstupové: logický součin (AND, &) a součet (OR, $\vee$).

Hradla kreslíme třeba následovně:



Hradlo provádějící logickou operaci AND se dvěma vstupy

Z jednotlivých hradel pak vytváříme hradlové sítě. Pokud používáme pouze booleovská hradla, říkáme takovým sítím *booleovské obvody,* pokud operace nad nějakou obecnější (ale konečnou) množinou symbolů (abecedou), nazývají se *kombinační obvody.* Každý vstup hradla je připojen buďto na některý ze vstupů sítě nebo na výstup nějakého jiného hradla. Výstupy hradel mohou být propojeny na výstupy sítě nebo přivedeny na vstupy dalších hradel, přičemž je zakázáno vytvářet cykly. Než si řekneme formální definici, podívejme se na obrázek.

TODO: *OBR*

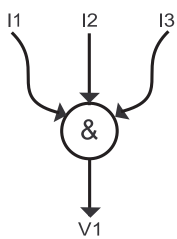
Definice: *Hradlová síť* je určena:

- *abecedou* Σ (to je nějaká konečná množina symbolů, obvykle $\Sigma = \{0,1\}$);
- množinou *hradel* H, *vstupních portů* I a *výstupních portů* O;
- acyklickým orientovaným grafem (V, E), kde $V = H \cup I \cup O$;
- zobrazením F, které každému hradlu $h \in H$ přiřadí nějakou funkci $F(h) : \Sigma^{a(h)} \to \Sigma$. To je funkce, kterou toto hradlo vykonává, a číslu $a(h)$ říkáme *arita* hradla h;
- zobrazením $z : E \to \mathbb{N}$, které každé hraně vedoucí do nějakého hradla přiřazuje některý ze vstupů tohoto hradla.

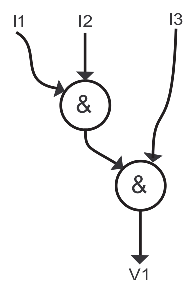
Přitom jsou splněny následující podmínky:

- $\forall i \in I : \deg^+(i) = 0$ (do vstupů nic nevede);
- $\forall o \in O : \deg^+(o) = 1$ & $\deg^-(o) = 0$ (z výstupů nic nevede a do každého vede právě jedna hrana);
- $\forall h \in H : \deg^+(v) = a(v)$ (do každého hradla vede tolik hran, kolik je jeho arita);
- $\forall h \in H, 1 \leq j \leq a(h)$ existuje právě jeden vrchol v takový, že $z(vh) = j$ (všechny vstupy hradel jsou zapojeny).

Pozorování: Kdybychom připustili hradla s libovolně vysokým počtem vstupů, mohli bychom libovolný problém se vstupem délky n vyřešit jedním hradlem o n vstupech, což není ani realistické, ani pěkné. Proto přijměme omezení, že všechna hradla budou mít maximálně k vstupů, kde k je nějaká pevná konstanta, obvykle dvojka. Následující obrázky ukazují, jak hradla o více vstupech nahradit dvouvstupovými:

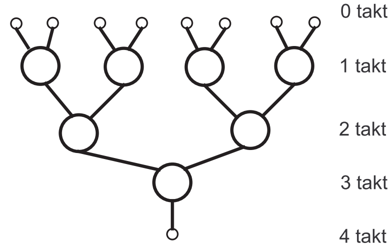


Trojvstupové hradlo AND



Jeho nahrazení 2-vstupovými hradly

Definice: *Výpočet sítě* probíhá v *taktech.* V nultém taktu jsou definovány právě hodnoty vstupních portů. V i-tém taktu vydají výsledek hradla, která jsou připojena na porty nebo na výstupy hradel, jejichž hodnota byla definována v $(i-1)$-ním taktu. Až po nějakém konečném počtu taktů budou definované i hodnoty výstupních portů, síť se zastaví a vydá výsledek.



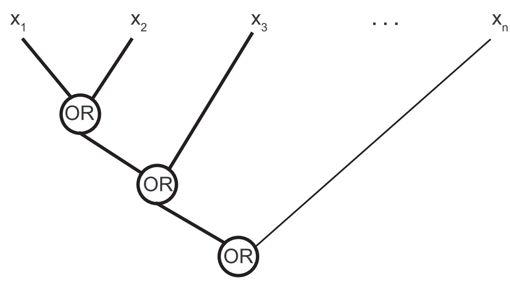
Výpočet hradlové sítě

Podle toho, jak síť počítá, si ji můžeme rozdělit na vrstvy:

Definice: *i-tá vrstva* obsahuje všechny vrcholy v takové, že nejdelší z cest z portů sítě do v má délku právě i. To jsou přesně vrcholy, které vydají výsledek poprvé v i-tém taktu výpočtu. Dává tedy smysl prohlásit za *časovou složitost* sítě počet jejích vrstev. Podobně *prostorovou složitost* definujeme jako počet hradel v síti.

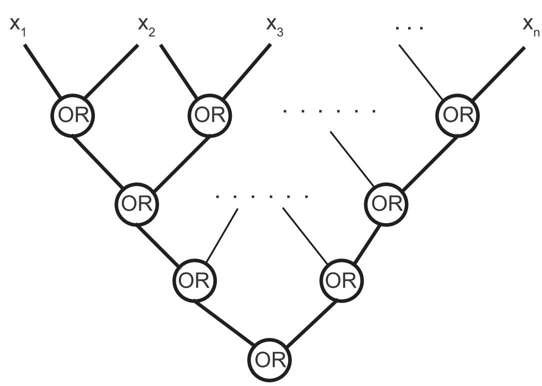
Příklad: Sestrojte síť, která zjistí, zda se mezi jejími n vstupy vyskytuje alespoň jedna jednička.

První řešení: zapojíme hradla za sebe (sériově). Časová a prostorová složitost jsou n. Zde vůbec nevyužíváme toho, že by mohlo počítat více hradel současně.



Hradlová síť, která zjistí zda-li je na vstupu alespoň jedna jednička

Druhé řešení: Budeme vrcholy spojovat do dvojic, pak výsledky z těchto dvojic opět do dvojic a tak dále. Tak dosáhneme časové složitosti $\Theta(\log n)$, prostorová složitost zůstane lineární.



Chytřejší řešení stejného problému

Sčítání binárních čísel

Pojďme se podívat na zajímavější problém: Mějme dvě čísla zapsané ve dvojkové soustavě jako $x_{n-1} \dots x_1 x_0$ a $y_{n-1} \dots y_1 y_0$. Budeme chtít spočítat jejich součet

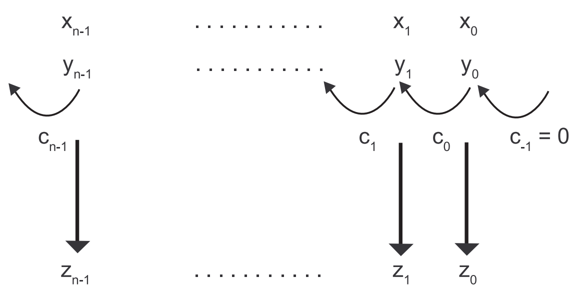
$z_n z_{n-1} \ldots z_1 z_0$.

Samozřejmě můžeme použít algoritmus „sčítání pod sebou“, který nás učili na základní škole. Formálně by se dal zapsat třeba takto:

$$z_i = x_i \oplus y_i \oplus c_{i-1},$$

kde $\oplus$ značí operaci XOR (součet modulo 2) a c_{i-1} je *přenos* z $(i-1)$-ního řádu do i-tého. Přenos přitom nastane tehdy, když ze tří xorovaných číslic jsou alespoň dvě jedničky:

$$\begin{aligned} c_{-1} &= 0 \\ c_i &= (x_i \mathbin{\&} y_i) \vee ((x_i \vee y_i) \mathbin{\&} c_{i-1}). \end{aligned}$$



Sčítání ze základní školy

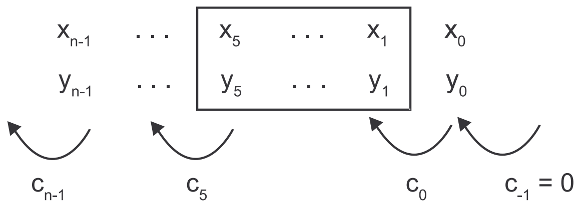
Bohužel na to, abychom spočítali c_i (a tedy z_i), musíme znát hodnotu c_{i-1}, tedy mít spočítané hodnoty pro všechny čísla menší než i. To dává lineární časovou složitost. Zamysleme se nad tím, jak by se proces sčítání mohl zrychlit.

Přenosy v blocích

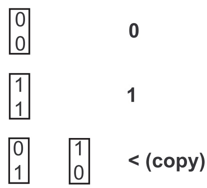
Jediné, co nás při sčítání brzdí, jsou přenosy. Kdybychom je dokázali spočítat rychle (řekněme v logaritmické hloubce), součet už zvládneme dopočítat v konstantním čase.

Podívejme se na libovolný *blok* v našem součtu. Tak budeme říkat číslům $x_a \ldots x_b$ a $y_a \ldots y_b$ v nějakém intervalu indexů $\langle a, b \rangle$. Přenos c_b vystupující z tohoto bloku závisí mimo hodnot sčítanců už pouze na přenosu c_{a-1}, který do bloku vstupuje. Záviset může pouze třemi možnými způsoby:

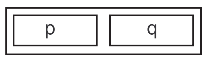
1. generuje přenos: $c_a = 1$,
2. pohlcuje přenos: $c_a = 0$,
3. kopíruje přenos: $c_a = c_{b-1}$.



Blok součtu



Tabulka triviálních bitů



p	q	B
0	*	0
1	*	1
copy	0	0
copy	1	1
copy	copy	copy

Skládání chování bloků

Cvičení: Rozmyslete si, jak přesně vypadají bloky s jednotlivými typy chování.

Jednobitové bloky se chovají velice jednoduše:

Pokud máme nějaký větší blok B složený z menších bloků p a q, jejichž chování už známe, můžeme z toho odvodit, jak se chová velký blok:

Všimněme si, že skládání chování bloků je asociativní operace (je to vlastně úplně obyčejné skládání funkcí), takže pro libovolný blok můžeme jeho chování spočítat v čase $\mathcal{O}(\log n)$ postupným skládáním („stromečkovým" způsobem).

To nám dá nějaký kombinační obvod nad trojprvkovou abecedou, ale samozřejmě můžeme chování bloků kódovat i binárně dvojicí bitů:

- $(1, *) = <$,
- $(0, 0) = 0$,
- $(0, 1) = 1$

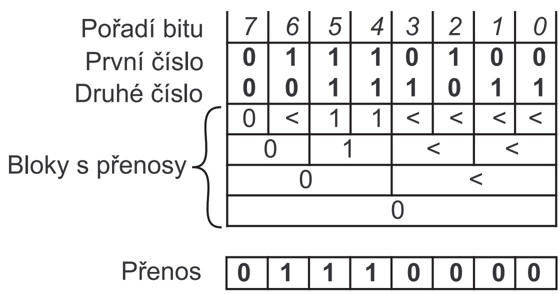
Operaci skládání $(a, x) \odot (b, y) = (c, z)$ pak definujeme takto:

$$\begin{aligned} c &= a \mathbin{\&} b, \\ z &= (\neg a \mathbin{\&} x) \vee (a \mathbin{\&} y). \end{aligned}$$

Paralelní sčítání

Paralelní algoritmus na sčítání už zkonstruujeme poměrně snadno. Bez újmy na obecnosti budeme předpokládat, že počet bitů vstupních čísel n je mocnina dvojky, jinak si vstup doplníme nulami.

1. Spočteme chování bloků velikosti 1. ($\mathcal{O}(1)$ hladin)
2. Postupně počítáme chování bloků velikosti 2^k na pozicích dělitelných 2^k. ($\mathcal{O}(\log n)$ hladin, na nichž se skládají bloky)
3. $c_{-1} \leftarrow 0$
4. Určíme c_n podle c_{-1} a chování (jediného) bloku velikosti n.
5. Postupně počítáme přenosy na hranicích dělitelných 2^k „zahušťováním“: jakmile víme c_{2^k-1}, můžeme dopočítat $c_{2^k+2^{k-1}-1}$ podle chování bloku $\langle 2^k + 2^{k-1} - 1, 2^k \rangle$. ($\mathcal{O}(\log n)$ hladin, na nichž se dosazuje)
6. $\forall i : z_i = x_i \oplus y_i \oplus c_{i-1}$.



Výpočet přenosu

Algoritmus pracuje v čase $\mathcal{O}(\log n)$. Hradel je dokonce lineárně: na jednotlivých hladinách kroku 2 počet hradel exponenciálně klesá od n k 1, na hladinách kroku 5 exponenciálně stoupá od 1 k n, takže se sečte na $\mathcal{O}(n)$.