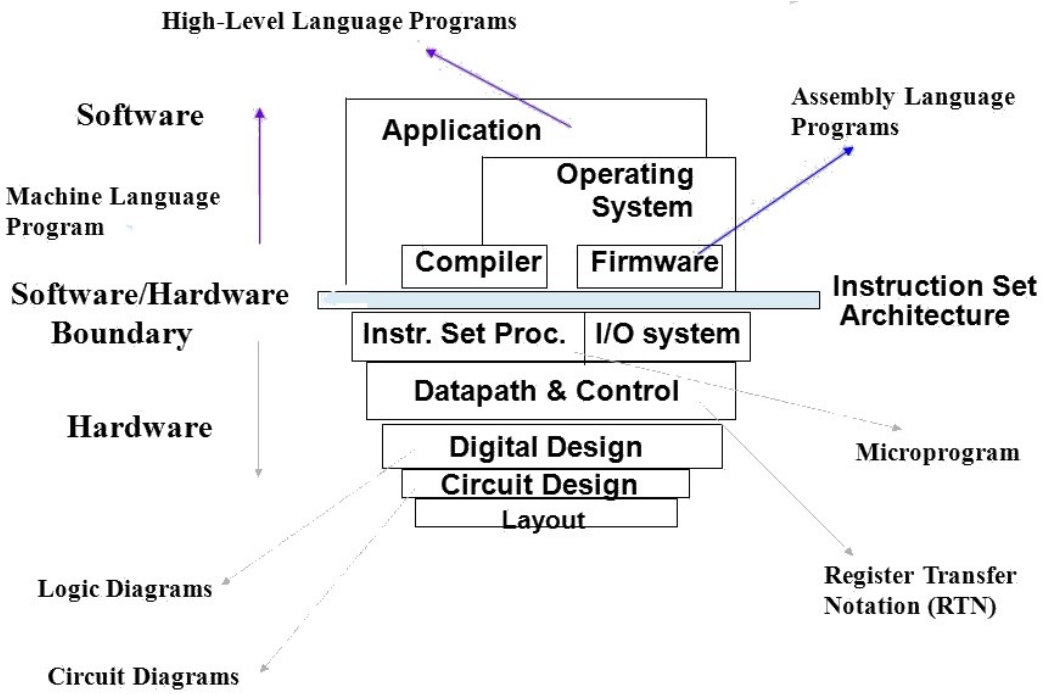
Architektury číslicových systémů

- zabývá se HW (s přesahem do SW) konstrukčními prvky číslicových systémů na všech úrovních (tranzistory, hradla, funkční bloky, aritmetické operace, sběrnice, řízení, procesory, instrukční sada, ..) jejich popisem (schema, stavový diagram, HDL (jazyky pro pro popis HW)) a algorimy pro návrh i realizaci.

- cíle návrhu číslicového systému – maximalizovat rychlost, výpočetní výkon, propustnost, … , minimalizovat složitost, plochu, spotřebu, ...

- naše oblast zájmu: Circuits design (návrh na úrovni hradel) – Instruction set Architecture (instr.sada)



Architektury číslicových systémů - Literatura

Tanenbaum, A. S.: Structured Computer Organisation, 4-6th Edition, Prentice Hall 1999-2013

Miloš D. Ercegovac, Tomás Lang, Digital Arithmetic, Morgan Kaufmann Publishers, 2004

Jean-Loup Baer: Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors, Cambridge University Press, 2009, ISBN 0521769922, 9780521769921

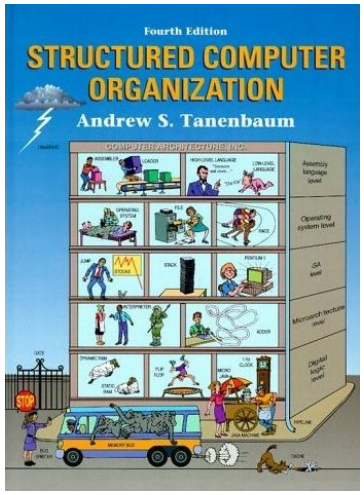
John L. Hennessy, David A. Patterson: Computer Architecture, Fifth Edition: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design), 2011, ISBN-13: 978-0123838728

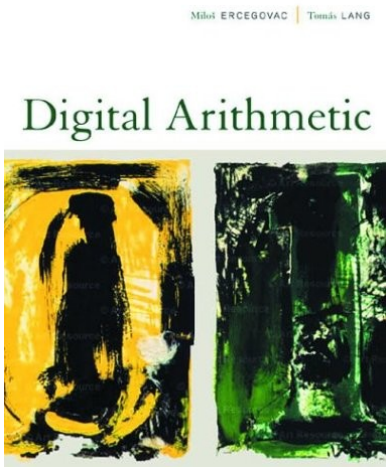
Gook, M.: Hardwarová rozhraní, Průvodce programátora, Computer Press 2006

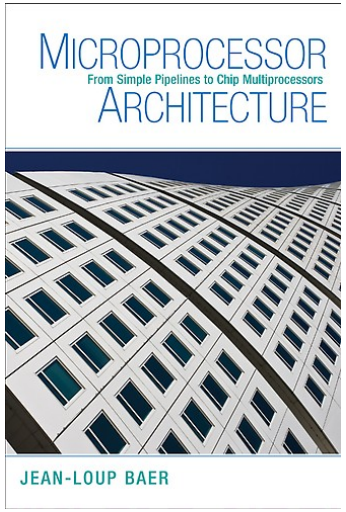
Pluháček A.: Computer Logic Design, skriptum ČVUT 2003

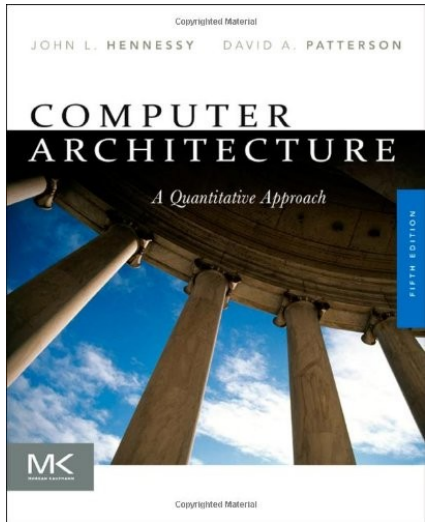
Douša J.: VHDL Language, skriptum ČVUT 2003

Pluháček, A.: Projektování logiky počítačů, ČVUT Praha, 2000









Číslicové systémy – Historie

Abakus (pomůcka) – starověké Řecko, Řím, Čína

– abakus – desticka vkládán kamínků (calculli)

Mechanické počítačí stroje

– Wilhelm Schickard, 1624: +, -, *, /

– Pascalina, 1642, vyr.50ks: +,-, pozdeji rozsirena o *,/

Děrné štítky - Joseph Marie, 1801 – tkalcovský stav kde vzorek látky bylo možné měnit změnou děrného šítku (ne přestavbou stroje). tj. první „program“

Programovatelné stroje

- Charles Babbage, 1833, „analytické stroj“ (mechanický) - první turingovsky úplný stroj

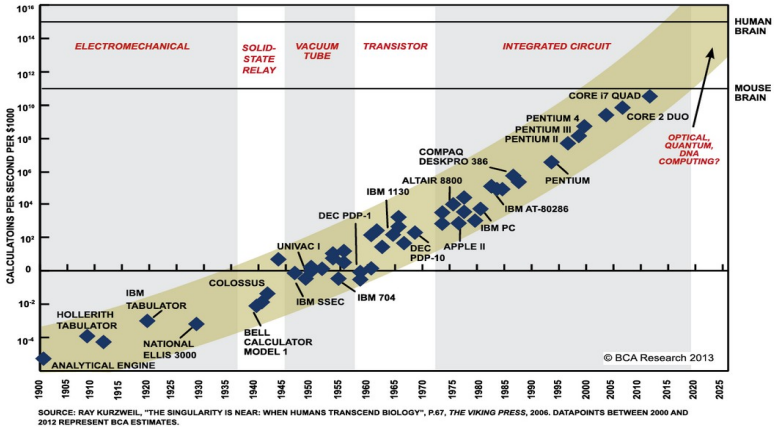
- Konrad Zuse (1934-1941) – reléový počítací stroj Z1,Z2,Z3 (200-2600relé)

Elektronkové počítače (I.gen) - ENIAC, 1944 (cca 20000elektronek, 30tun, 63m2, 150kW)

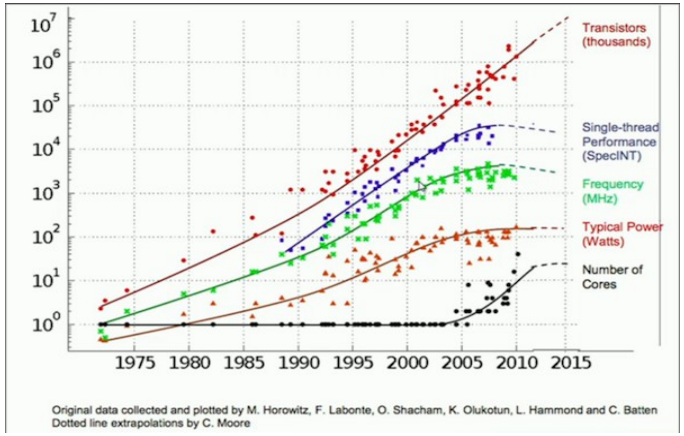
Tranzistorové počítače (II.gen) – 1951-1965 – počátek oper.systémů, assembleru a prog.jazyků (cobol, fortran, algol)

Integrované obvody (III.generace) – 1965-1980 – počátek mutitaskingu, první minipočítače (předtím mainframy (sálové počítače))

Procesory (IV.generace) – od.r.1981 – rozvoj sítí (internet), mutiprocesorové systémy



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.



Budoucnost (?)

Moorův zákon. Rychlost, spotřeba, cena.

Logické (digitální) obvody

- obvod pracuje s diskrétními stavy (vs.analogový), je tvořen log.členy(hradly). (rozhodovací úroveň, napěťové úrovně 0/1 In/Out)

- analogový - (+) rychlost, (-) přesnost, opakovatelnost, specializovanost) vs digitální - (+) opakovatelnost, univerzálnost

- kombinační (výstup fci vstupu) vs. Sekvenční (vnitřní stav) - Mealyho (Y=f(X,Z)), Moorův (Y=f(Z)

- asynchronní vs synchronní – úrovňové, hranové

- popis log.obvodu - výraz (logická funkce) , tabulka, Karnoughova mapa (K-mapa), log.obvod (schéma), diagram, program (HDL - Verilog, VHDL)

- Boolovská algebra – operátory a pravidla

- Základní log.fce (hradla) – NOT, AND/NAND, OR/NOR, XOR

- Základní Paměťové členy – Klopné obvody – RS, D, JK,

- Další stavební prvky – Buffery, drivery, registry, paměti, Dekodéry, Multiplexery, čítače

- Programovatelné log.obvody (PLD) – PAL/GAL, CPLD, FPGA

- procesory, mikrokontolery

- ASIC (zákaznické obvody – plně, polozákaznické)

- technologie – bipolární (e.g. TTL, IIL, ECL) , unipolární (PMOS, NMOS, CMOS)

- charakteristiky technologie – zpoždění(=rychlost), spotřeba, fanin (počet vstupů), fanout (počet připojitelných vstupů k výstupu), šumová odolnost, napájení, operační teplota, ...

Zobrazení čísel, aritmetika

- počítačová aritmetika – implementace aritmetických operací a funkcí – algoritmy pro HW i SW, kterou souvisí se zobr.čísel (přesnost, přetečení)

Zápis čísel

nejstarší 1:1 – záspis čísel počtem symbolů (6 mamutů = 6 kaménků), vytváření skupin po 5 nebo po 10

římský číselný systém – 1,5,10,50,100,500,1000 = I,V,X,C,L,D,M – nevhodný pro reprezentaci velkých a des.čísel, a pro aritm.operace

poziční číselný systém (poprvé v číně) – hodnota symbolu je dána polohou

smíšený poziční systém – základ není konstanta ale vektor, např. Čas H:M:S

poziční systém s pevným základem – základ R (radix) – typ.dekadická R=10 (123 = 1.10^2+2*10^1+3*10^0), **binární** R=2 (10=1*2^1+0*2^0)

optimální číselný základ (minimalizace stavů pro uchování informace) = e -> 3

vyvážená (symetrická) trojková číselná soustava – cifry {-1;0;1}, základ R=2, např. -2, -1, 0, 1,2,3,4, .. (dec) = $\bar{1}0$, $\bar{1}$, 0, 1, $1\bar{1}$, 10, 11, ..

redundatní systém se základem R=4 , cifry {-2, -1, 0, 1, 2}, napr. 6(dec) = 1 2 = 2 $\bar{2}$

systémy se záporným základem, systémy se zlomkovými základy, iracionální základy, základy s komplexním číslem,...

„Ahhh, what an awful dream. Ones and zeroes everywhere…[shudder] and I thought I saw a two.“ – Bender

„It was just a dream, Bender. There's no such thing as two“ – Fry

(Futurama)

Booleovská algebra

!x, -x, ~x, $\bar{x}$ = negace,inverze (~0=1, ~1=0)

x.y, x&y = log.soucin, and (0&0=0, 0&1=0, 1&0=0, 1&1=1)

x+y, x|y = log.součet, or (0+0=0, 0+1=1, 1+0=1, 1+1=1)

komutativita: x + y = y + x, x.y = y.x

distributivita: x + (y.z) = (x+y) . (x+z), x. (y+z) = (x.y) + (x.z)

neutralita 0/1: x+0 = x, x.1 = x

komplementarita: x + ~x = 1, x.~x = 0

(!pozor, + a . je jen symbol log.fcí OR a AND, není to arimetické + a *)

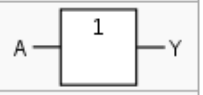
- asociativita: x + (y+z) = (x + y) + z, x.(y.z) = (x.y).z
- absorpce: x + (*x.y) = x*, *x.(x+y) = x*
- agresivita nuly,jedničky: x.0 = 0, x+1 = 1
- idempotence: x+x = x, x.x = x
- absorce negace: x+(~x.y) = x+y, x.(~x+y) = x.y
- dvojitá negace: ~(~x) = x
- komplementarita 0/1: ~0=1, ~1=0
- DeMorganovy zákony: ~x.~y = ~(x+y), ~x+~y = ~(x.y)
- DeMorganovy zákony: x.y = ~(~x+~y), x+y = ~(~x.~y)

Základní logické funkce (hradla)

Opakovač, Repeater, Driver, Budič	AND, log.soucin, konjunkce	OR, log.součet, disjukce
NOT, Invertor, Negace	NAND, negovaný log.součin	XOR, exkluzivní log.součet

Funkce $\mathbf{Y = A}$

Značení

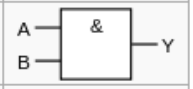
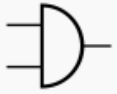
norma	symbol
ANSI/MIL	
IEC	
DIN	

Pravdivostní tabulka

$X(A)$	Y
0	0
1	1

Funkce $\mathbf{Y = A \cdot B}$

Značení

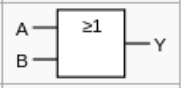
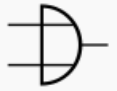
norma	symbol
ANSI/MIL	
IEC	
DIN	

Pravdivostní tabulka

$X_1(A)$	$X_2(B)$	Y
0	0	0
0	1	0
1	0	0
1	1	1

Funkce $\mathbf{Y = A + B}$

Značení

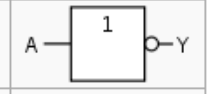
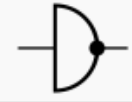
norma	symbol
ANSI/MIL	
IEC	
DIN	

Pravdivostní tabulka

$X_1(A)$	$X_2(B)$	Y
0	0	0
0	1	1
1	0	1
1	1	1

Funkce $\mathbf{Y = \overline{A}}$

Značení

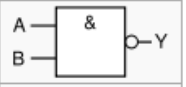
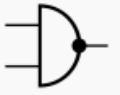
norma	symbol
ANSI/MIL	
IEC	
DIN	

Pravdivostní tabulka

$X(A)$	Y
0	1
1	0

Funkce $\mathbf{Y = \overline{A \cdot B} = \overline{A} + \overline{B}}$

Značení

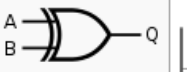
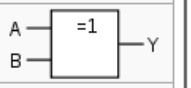
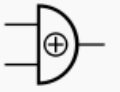
norma	symbol
ANSI/MIL	
IEC	
DIN	

Pravdivostní tabulka

$X_1(A)$	$X_2(B)$	Y
0	0	1
0	1	1
1	0	1
1	1	0

Funkce $\mathbf{Y = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}}$

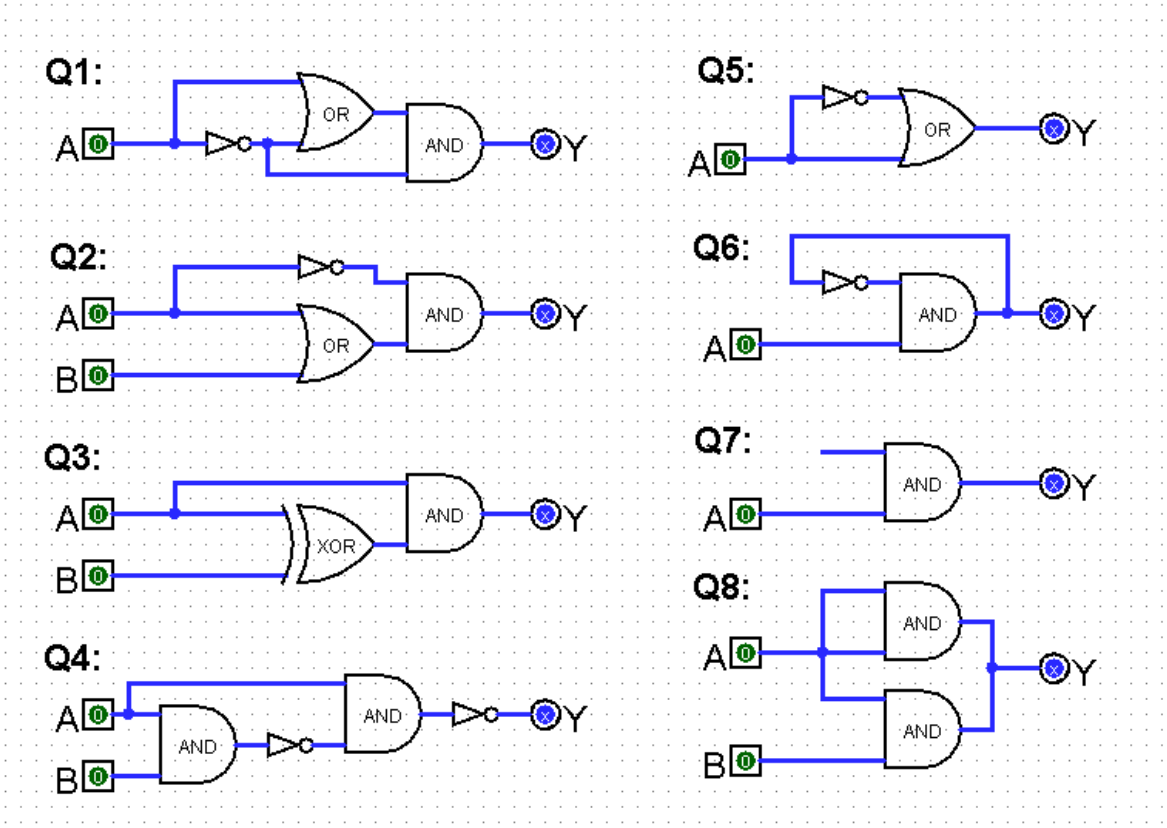
Značení

norma	symbol
ANSI/MIL	
IEC	
DIN	

Pravdivostní tabulka

$X_1(A)$	$X_2(B)$	Y
0	0	0
0	1	1
1	0	1
1	1	0

Příklady (kresleno v programu logisim)



Q1,Q2,Q3,Q4: kdy je Y=1? (Q1: A=0; Q2: AB=01; Q3: AB=10; Q4: AB=00/01/11)

Q5: Y=?, platí stále? Pozor na přechodový jev → hazard (glitch)

Q6: Y=?. Pozor! kmitá pro A=1.

Q7: Y=?. Pozor! nedefinovaný vstup.

Q8: Y=?. Pozor! zkrat na výstupu

Zobrazení celých čísel (v binární poziční soustavě)

Kladná čísla – 1Byte – bit7..bit0, 0-255 (2^8), 2Byte 0-65535 (2^16), …

Znaménková čísla (1 Byte)

- Přímý kód – nejvyšší bit znaménko, 0=+, 1=-, tj. 0X84 = - 4. je třeba testovat znaménko a podle toho sčítat odčítat, dvě nuly

- Inverzní kód – kladná čísla přímo, záporná inverzně, tj. -4 = -00000100 = 11111011. stále dvě reprezentace nuly, jen myšlenkový přechod k:

- Doplňkový kód - kladná čísla přímo, záporná dvokový doplněk (negace a přičtení 1) tj. -4 = - 00000100 = 11111011 +1 = 11111100

zobrazení -128 az 127 … 10000000 … 11111111 00000000 … 01111111

není dvojitá nula, vhodné při binární operace, přímá návaznost čísel -1 0 1 =snadná realizace aritmetiky (odčítání, sčítání)

- Kód s posunutou nulou

dohnutá „pozice“ 0, napr. 0=127. potom -4 = 123, 4=131. výhoda návaznosti, lepší porovnávání (vyšší číslo je vyšší), snadné sčítání

násobení je třeba ošetřit. Používá se pro reprezentaci exponentu pro reálná čísla

Zobrazení reálných čísel

Fixní desetinná tečka

0001.1100 = 1.75 = 28/16

Exponenciální formát (IEEE 754)

FORMÁT: S Exp Mantisa … S znaménko čísla (0=kladné, 1=záporné), Mantisa v Přímém kódu, Exponent v kód s posunutou nulou

Normalizovaný tvar Mantisy: 1.xxx .,První jedničku (která je tam vždy kromě výsledku 0) vynecháváme.

Přesnost, přetečení aritmetických operací

Celá čísla - číselný rozsah. Přenos (carry flag). Přetečení (overflow)

Pevná řádová čárka – číselný rozsah.

Pohyblivá řádová čárka – číselný rozsah/číselná osa. Přesnost (sčítání, násobení, zaokrouhlování)

a=1e10+1+1+1+1+1+...+1 //? int a, float a

a=10000*10000*10000*10000*10000; //? int a, float a

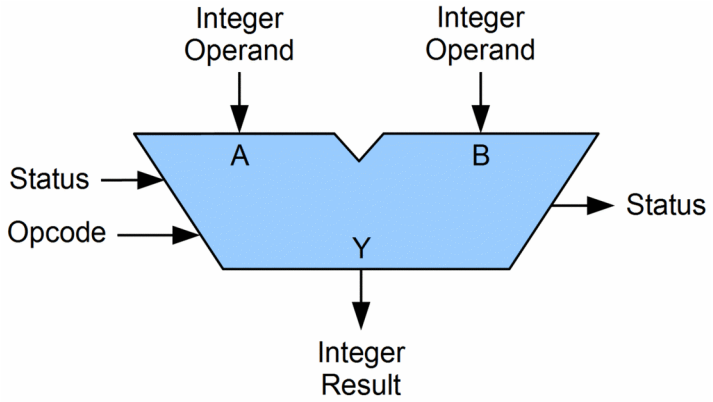
while(1) a+=1;   //? int a, float a

Důsledky chybné aritmetiky ...

Protiraketová střela Patriot v r.1991 – nepřesná kalkulace času – ochylka 0.34s/100hod → cca 500m odchylka (0.34s*1600m/s) → mimo dosah nepřátelské rakety -> nezničení, zabití 28lidí. Při testech krátký (několikahodinový) ‚up-time“ takže se neprojevilo.

Exploze rakety Ariane5 v r.1996 – 64bit float byl konvertován do 16 bit integer v modulu navigačnho systému (toto spolehlivě pracovalo v Ariane4,ale v Ariane5 byla tato hodnota 5* větší) → vyjímka mimo rozsah → 37s po vypuštění - exploze → přímá ztráta $500 000 000 (raketa a náklad), nepřímá $7 000 000 000 (vývoj)

Aritmeticko-logická jednotka (ALU)



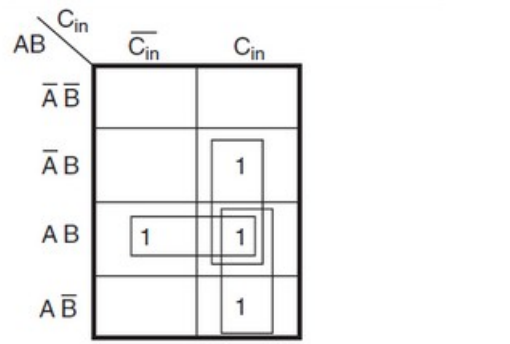
- základní prvek pro provádění aritmetických a logických operací
- podstatný vliv na celkový výkon procesoru
- základem ALU je (často) sčítačka
- vstupy (typ.2), výstup, řízení (control,opcode), status in/out (carry,overflow,zero)

Sčítačka 1bit

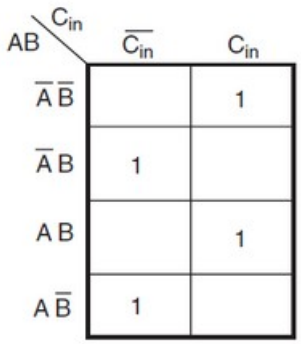
Půlsčítačka (half adder, HA) : A+B→ S,Cout (S=A xor B, C=A.B)

Úplná sčítačka (full adder, FA) : A+B+Cin→ S,Cout

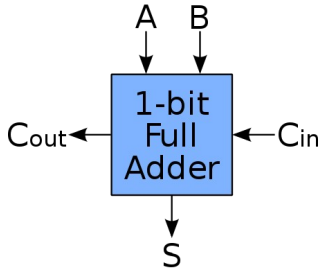
A	B	C	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$C_{out} = \overline{A}.B.C_{in} + A.\overline{B}.C_{in} + A.B.\overline{C}_{in} + A.B.C_{in}$



$S = \overline{A}.\overline{B}.C_{in} + \overline{A}.B.\overline{C}_{in} + A.\overline{B}.\overline{C}_{in} + A.B.C_{in}$



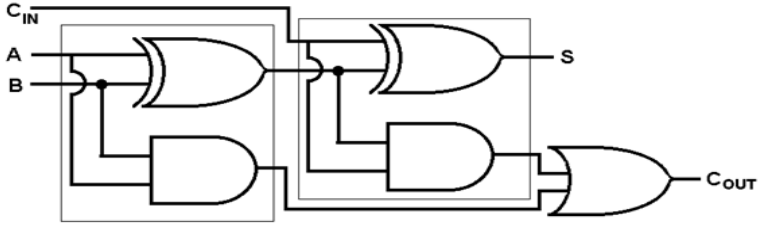
alternativně:

Cout = A.B + A.C + B.C S = $\overline{A}$ (B xor C) + A $\overline{(B \text{ xor } C)}$

Cout = A.B + C (A+B)

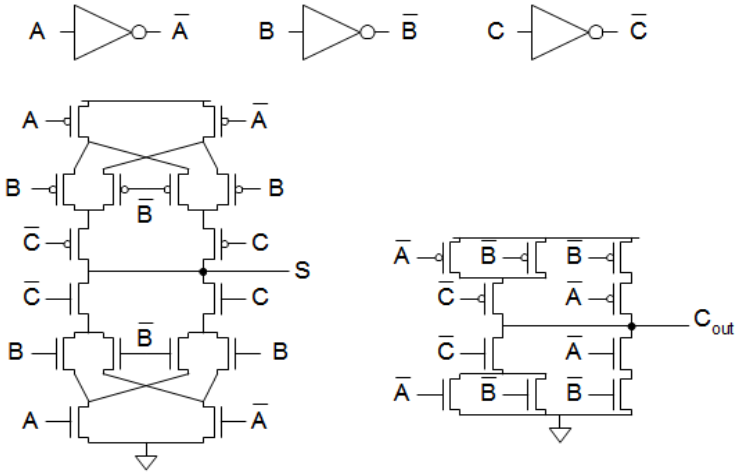
Cout = A.B + C. (A xor B) S=A xor B xor C

.... posl.radka - full adder z dvou half adderů + OR (viz obr)

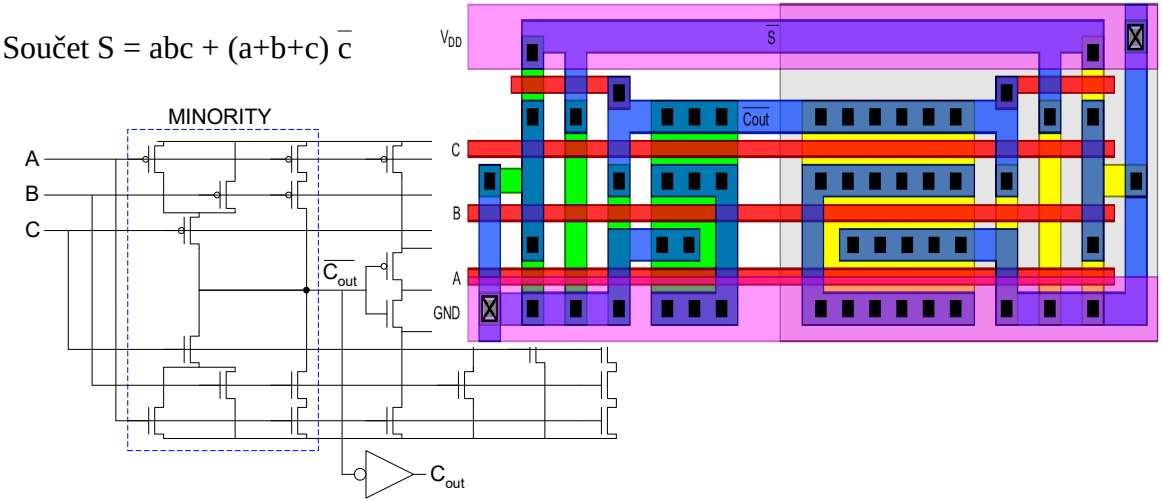


Sčítačka – technologie (pro představu)

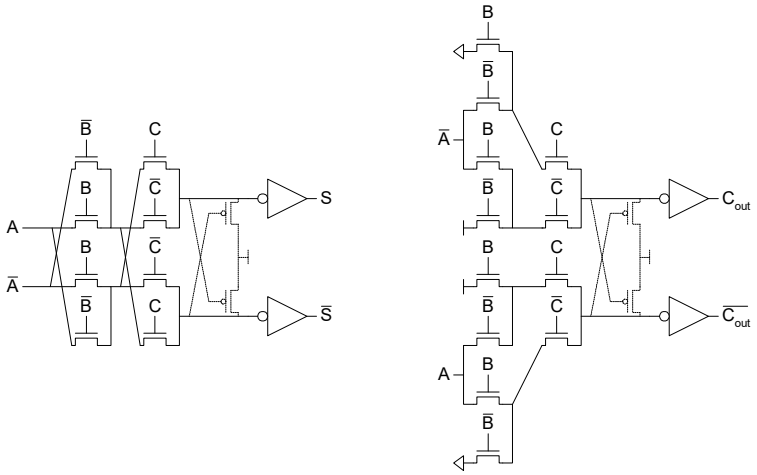
Implementace z rovnic S=A xor B xor C, Cout=a b + a c + b c



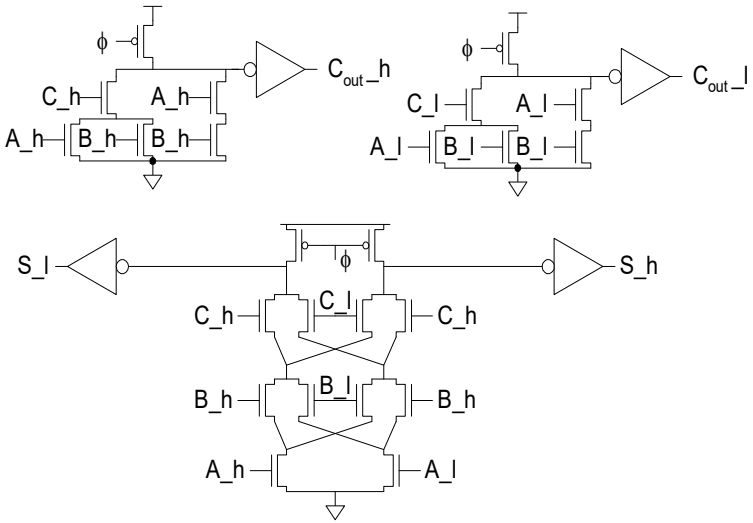
Součet S = abc + (a+b+c) $\bar{c}$



Komplementární tranzistory (rychlejší, ale zabírá více místa)



Dual-Rail Domino – velmi rychlé, ale velká spotřeba

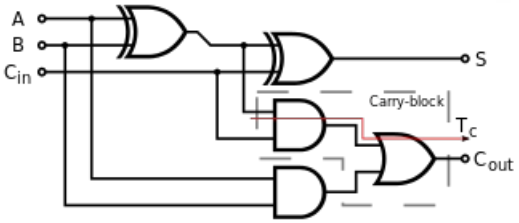


Riple Carry Adder (RCA) (=Carry Propagate Adder (CPA))

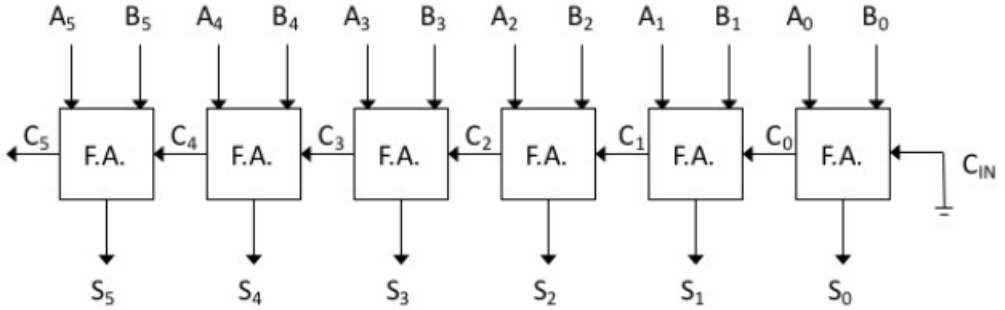
- jednoduché spojení N 1bitovývh sčítaček

- přenos prochází od nejnižšího řádu k nejvyššímu

- doba součtu je úměrná počtu bitů, tj. nejdelší doba součtu je N

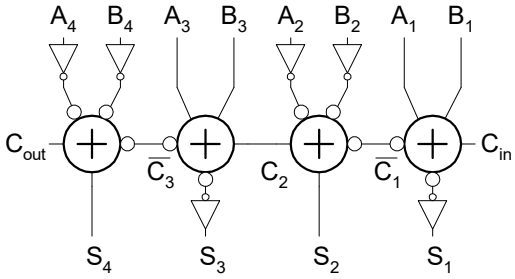


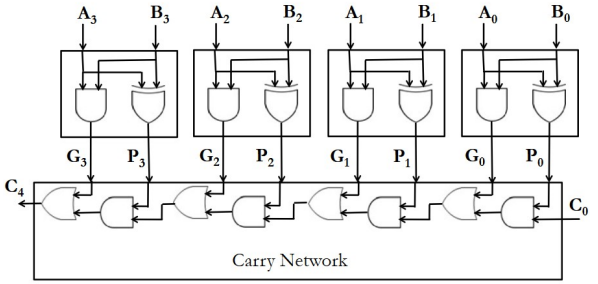
- Zpoždění 3+(N-1)*2 (3 výpocet Cout0 + 2 přenos carry v dalších)



Urychlení („technologické“)

Kritickou cestu lze urychlit využitím invertovaných signálů u suchých sčítaček





Přenos Carry (je fce A,B)

Generace: Cout=1 nezávisle na Cin: G = A.B (tj. A=B=1)

Propouštění/propagace: Cout=Cin: P = A xor B

Anihilace(kill): Cout=0 nezávisle na Cin: K = ~A.~B (tj. A=B=0)

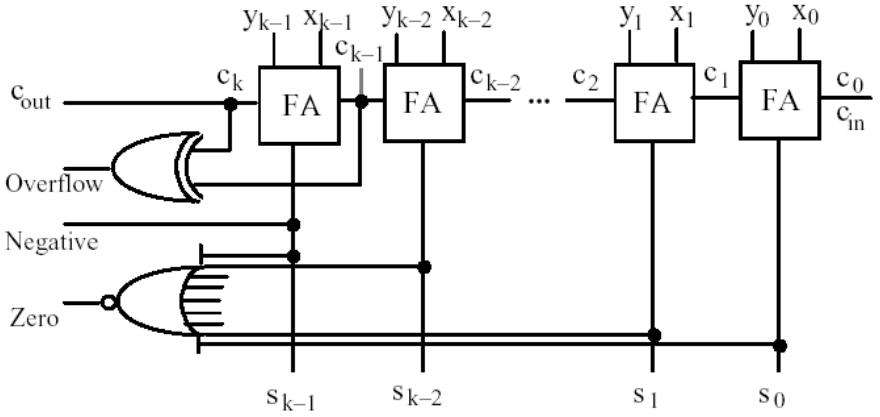
Podmínkové/příznakové signály (Flagy)

Carry = Ck

Overflow = Ck xor Ck-1

Negative = Sk-1

Zero = S0.S1.S2….Sk-1



Paralelní asychnronní sumátor

- Základní myšlenka - doba součtu sčítačky se sériově probíhajícím přenosem (RCA) nemusí vždy odpovídat nejhoršímu případu.

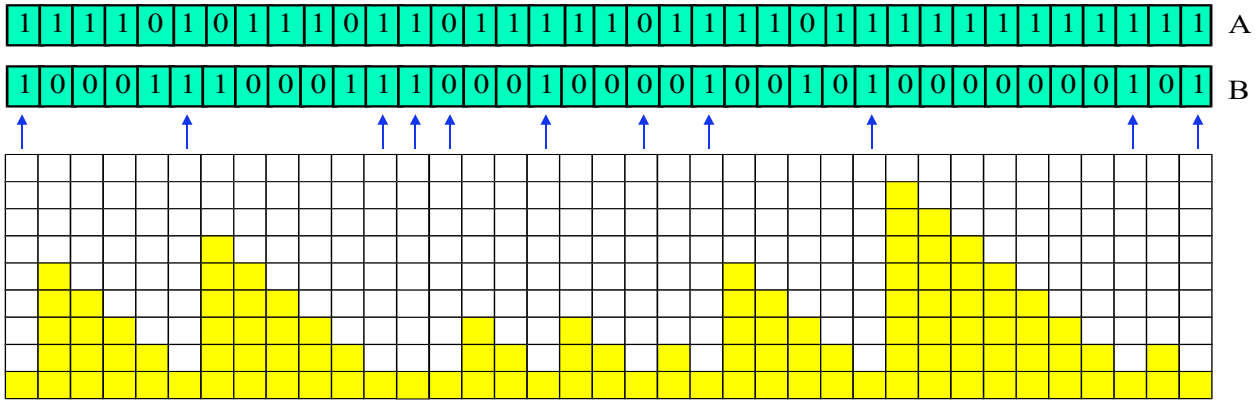
- Doplnění obvodů pro indikaci dokončení operace součtu

- Doba součtu je proměnná a závisí na operandech

- Kdy? Je-li generace nebo anihilace

- tj. ai=bi=1 nebo ai=bi=0

- výskytem se sumátor „rozpadá“ na větší počet samostatně pracujících částí



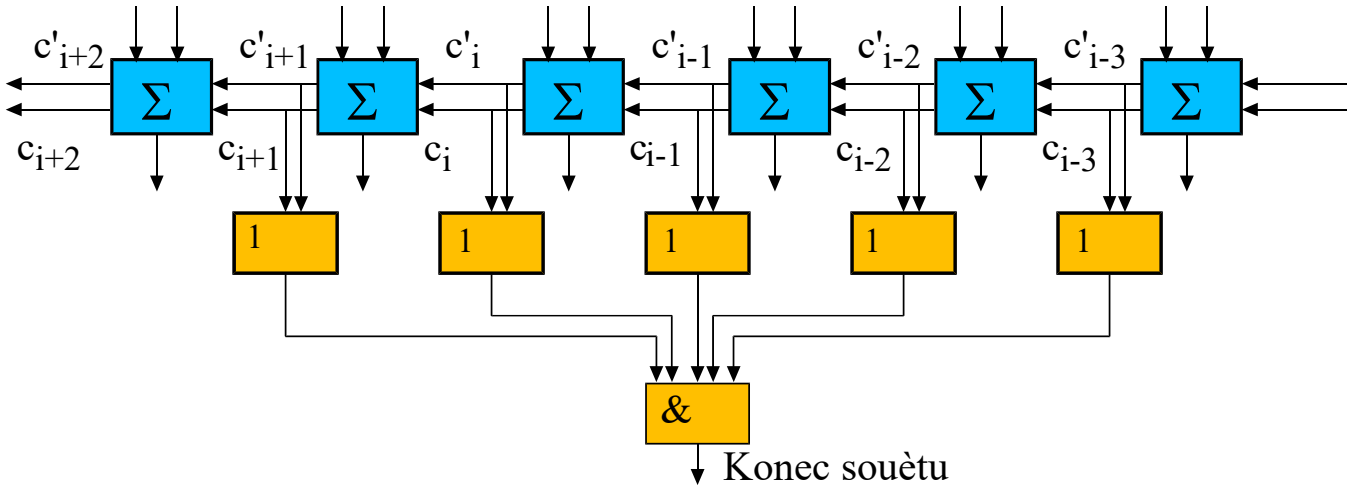
Příklad: (výška žlutých polí vyjadřuje zpoždění)

Odhad průměrné doby výpočtu

P(generace)=0.25, P(anihilace)=0.25, P(průchodu,propagace)=0.5

Lze po té určit že střední délka nejdelšího přenosu je L=log2(1.25 N) (N=délka sčítačky) (např. N=32b -> L=5.32b)

Tento konec je třeba detekovat



Sčítačka s Carry Lookahead (“predikcí” přenosu) (CLA)

- Fci sčítačky, resp. fci přenosu (carry) můžeme teoreticky rozvinout do libovolné úrovně jako dvouúrovňovou fci - Ci= f(Ai,Bi,Ai-1,B-1,...A0,B0,C0)

- Zavedeme Ci+1 = Gi + Ci . Pi (fce G .. přenos generován, P – přenos propuštěn jsme definovali již dříve a je fcí A,B), pak např.

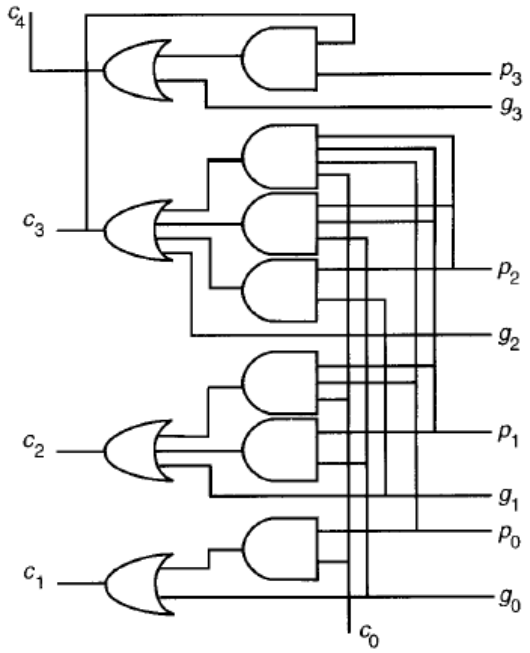
C1 = g0 + p0.c0

C2 = g1 + p1.c1 = g1 + p1.g0 + p1.p0.c0

C3 = g2 + p2.c2 = g2 + p2.g1 + p2.p1.g0 + p2.p1.p0.c0

C4 = g3 + p3.c3 = g3 + p3.g2 + p3.p2.g1 + p3.p2.p1.g0 + p3.p2.p1.p0.c0

(tedy C4 je funkcí a0-a3,b0-b3 a c0)



Tedy (v příkladu) predikujeme(vypočítáváme) všechny přechny přenosy pro 4 bitovou sčítačku

Pokud tento obvod přidáme ke sčítačče

- nemusíme genrovat přenosy (zjednodušení vlastní sčítačky)

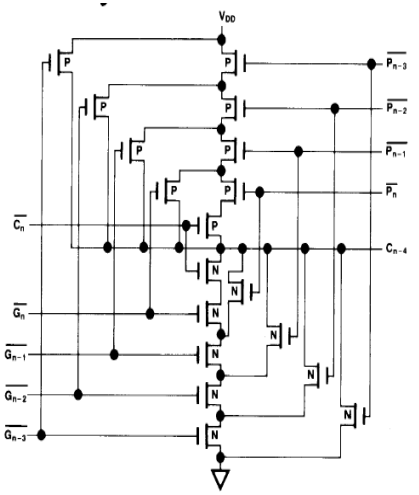
- výsledná fce je složitější (tj. větší cena, spotřeba(?-absolutni/relativni))

- všechny přenosy jsou „okamžitě“ (se zpožděním 1+2 (zpoždění 1 za generaci Pi,Gi + zpoždění za generaci Ci)) známé, tj. Výpočet Carry proběhne za O(3)

- další zpoždění 2 (někdy 3) za propagaci Carry do sumací – tj. Výpočet O(5)

- pro širší vstupy je dle omezení technologie (fanin) je nutno přidat další vrtstvu (zpoždění +1)

- typická šířka carry lookahead je 4 bity



Serializace Carry LookAhead (L za sebou jdoucích urovní Carry lookahead)

- zpoždění 1+2L+2 (1 generace Pi,Gi + 2 propagace carry zkrz jeden Lookahead + 2 propagace carry do sumaci)

- pro 16bit sčítačku se (čtyřmi) 4bitovými look ahead (tj. L=16b/4b=4) = 1+2*4+2 = 11

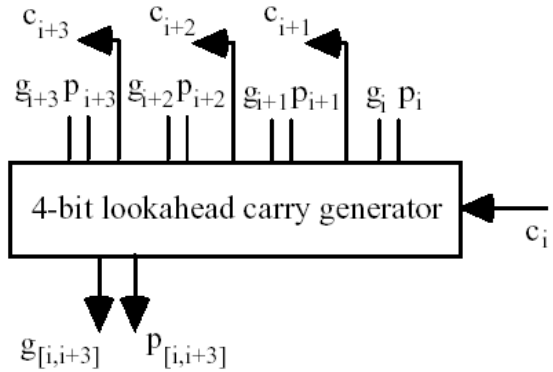
Stromová struktura Carry LookAhead (T – úrovní stromu)

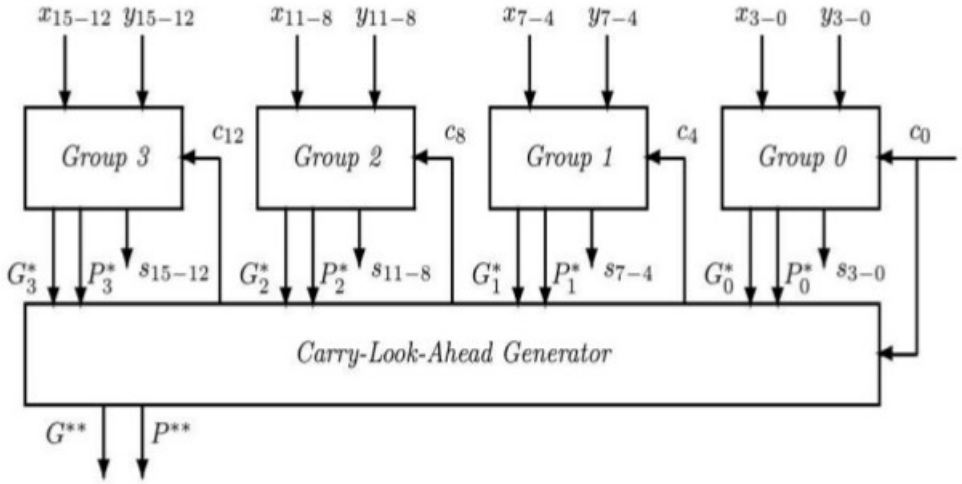
- pro dvouúrovňovou struturu (viz obr): (lookahead obvod upravíme tak ze ještě generuje výstupy G a P pro skupinu)

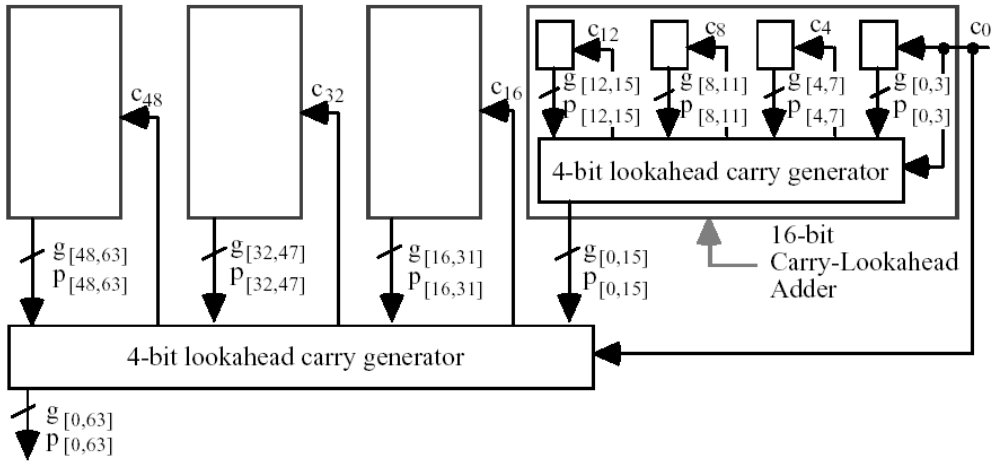
zpoždění 1 + 4 + 4 (1 generaci Pi,Gi, 4 progragace caary pres dve vrstvy + 4 propagace carry do sumaci pres dve vrstvy) = 9

(jestliže lookahead obvod negeneruje pomocne signaly G a P pro další vrstvu pak ještě +2)

Rychlost O(log N). Plocha O (N.log N).





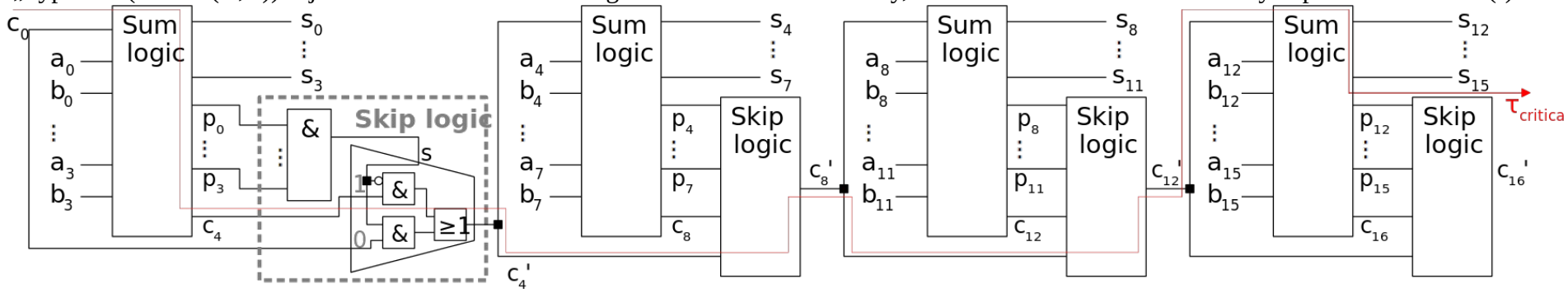


Carry-skip adder (CSA) (=Carry Bypass Adder)

Sčítačku rozdělujeme na jednotlivé bloky ve kterých zjištujeme dochází-li k propagaci a nebo je C vypočítán, tj. dva možné případy:

- propagace (Cout=Cin) – jestli dojde k propagaci lze zjistit na zaklade ai,bi – tj. s konstantním malým zpožděním po příchodu Cin

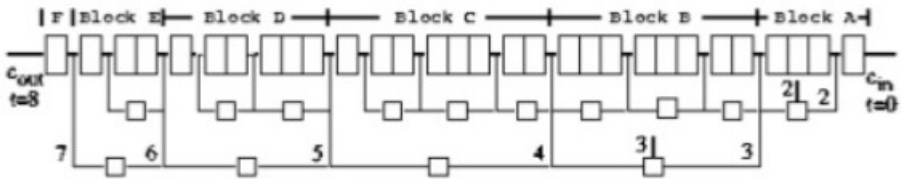
- „výpočet“ (Cout=f(A,B)) - tj. někde v bloku dochází ke generaci neho anhilaci carry, ale neřešíme kde – ale víme že výstupní Cout nezávisí(!) na Cin



Kritická (nejpomalejčí cesta) výpočtu pro bloky stejné velikosti n – v prvním bloku je C vypočítáván (čas nT), v dalších přeskakován (m*1T) (pokud by byl vypocitavan tak je již nT znám) + v posledním bloku je ještě třeba počkat na výpočet sumace jenž je závislý na Cin (čas nT) - Velikost bloků nemusí být stejná (naopak to vede k lepším výsledkům – ideálně nejkratší první a poslední sčítačka, uprostřed největší)

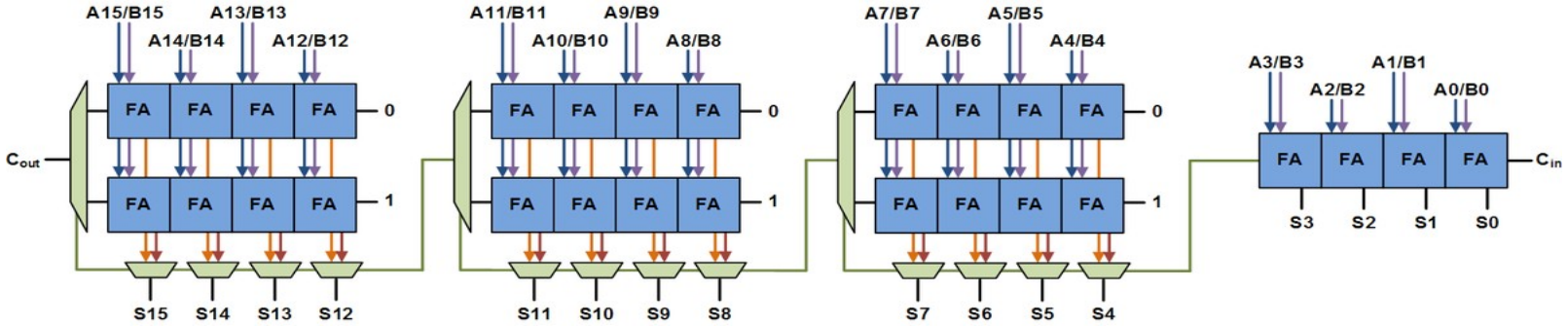
(první bloky jsou malé kvůli rychlému výpočtu carry, postupne se mohou bloky o 1 zvetšovat protože i bypass má zpoždění, odprostřed se bloky zase zmenšijí kvůli výpočtu z došlého carry, aby se na výsledek nečekalo než se přepočte dlouhý blok)

I tento typ sčítačky lze stavět stromově (tj. Carry skip logic přes Carry skip logic)



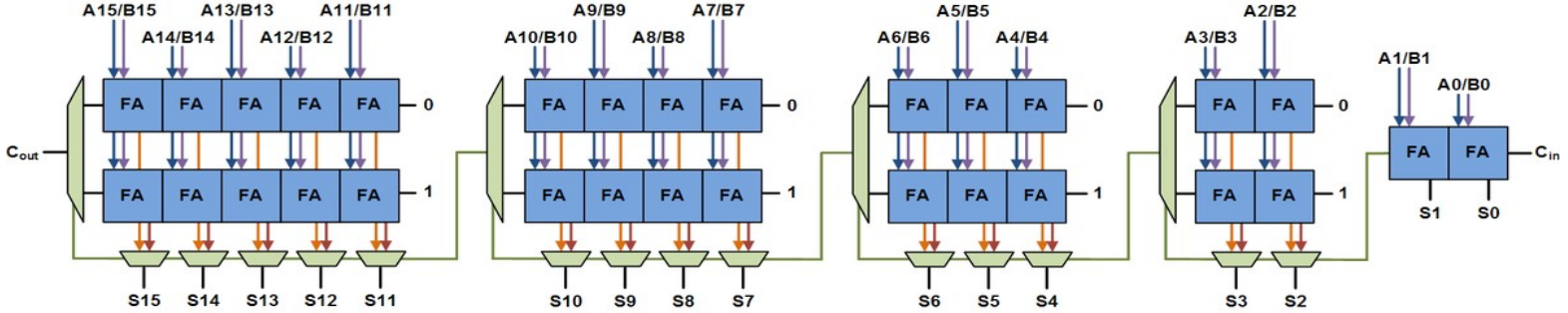
Carry-Select Adder (CSA)

- vypočítává n-bitové součty pro oba možné přenosy – po příchodu přenosu z předcozího bloku už se provede jen výběr správného výsledku



- pro optimální rozložení je výpočetní složitost O(sqrt(N)). Plocha O(n).

- Velikost bloků nemusí být stejná (naopak to vede k lepším výsledkům):



Carry-Inc Adder (CIA)

- obdobný princip. Ale místo výpočtu obou výsledků a výběru přes multiplexer je informace kódována jednodušeji (G,P signály) a konečný výběr je dělán fcí XOR, tedy kritická cesta je rychlejší a (díky absenci multiplexerů) je potřeba cca polovina hradel.

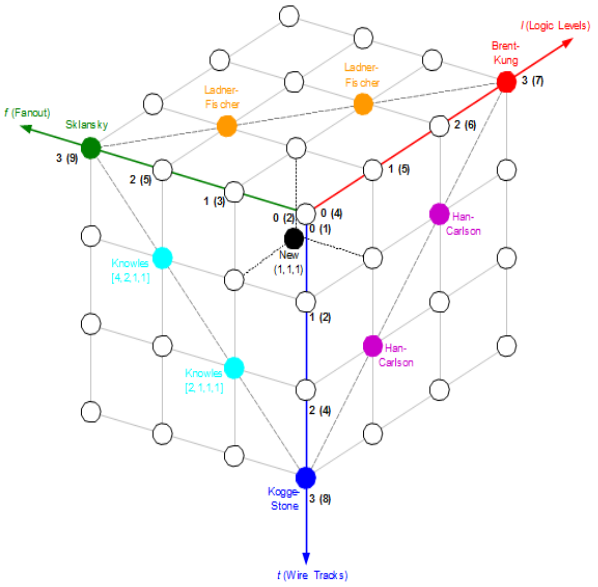
Další Sčítačky

Lingovy sčítačky (faktorizace Carry lookahead – vyšší rychlost, uniformí zatížení fann/fanout)

Optimalizae dle počtu log.úrovní, větvění (fanin/out), počtu spojů – Sklansky, Brent-Kung, Koge-Stone.

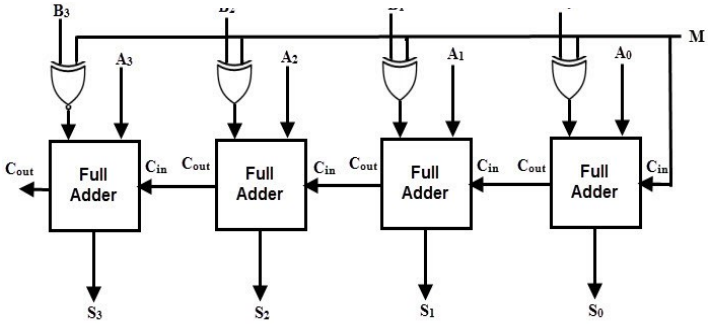
Dále mezi nimi: Knowles, Han-Carlson, Ladner-Fisher

Architektura	Klasifikace	# Logických úrovní	Max # větvení	Spoje	# Buněk
Carry-Ripple		N-1	1	1	N
Carry-Skip n=4		N/4 + 5	2	1	1.25N
Carry-Inc. n=4		N/4 + 2	4	1	2N
Brent-Kung	(L-1, 0, 0)	$2\log_2 N - 1$	2	1	2N
Sklansky	(0, L-1, 0)	$\log_2 N$	N/2 + 1	1	0.5 $N\log_2 N$
Kogge-Stone	(0, 0, L-1)	$\log_2 N$	2	N/2	$N\log_2 N$



Odčítačka (subtractor)

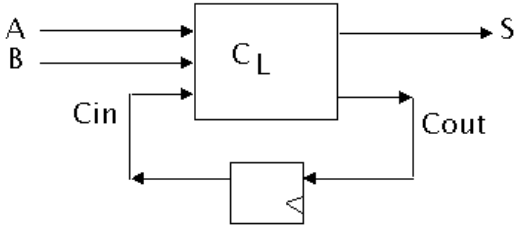
Sečítíme A a -B, tj. (uděláme dvojkový doplněk B) A + ~B +1, kde +1 můžeme nahradit vstupním Carry=1



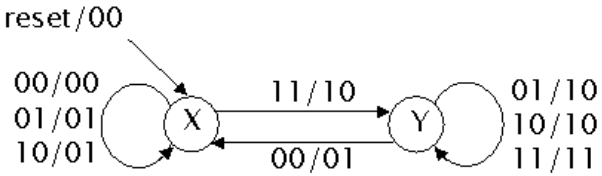
Sériová sčítačka (serial adder)

A,B přichází sériově (v taktu hodin), odebíráme S, sečítáme čísla libovolné délky.

Časová složitost O(N), velikost O(1).



		P.S.	N.S.	
A	B	Cin	Cout	S
0	0	X:0	X:0	0
0	0	Y:1	X:0	1
0	1	X:0	X:0	1
0	1	Y:1	Y:1	0
1	0	X:0	X:0	1
1	0	Y:1	Y:1	0
1	1	X:0	Y:1	0
1	1	Y:1	Y:1	1



X: carry is 0
Y: carry is 1

Čítače (Counters)

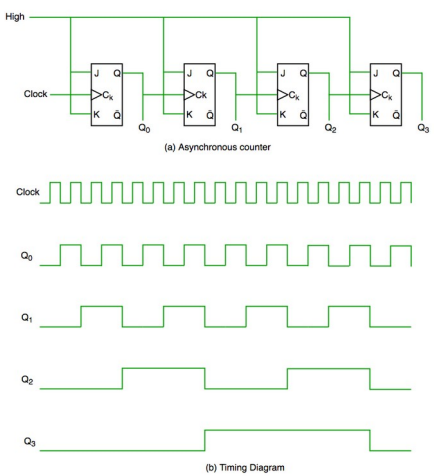
- nejsou sčítačky(!), slouží např. pro počítání cyklů 0..N

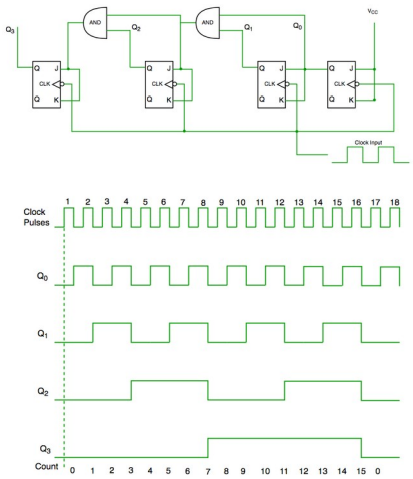
- Jako čítač LZE použít sčítačku ale občas zbytečně náročné

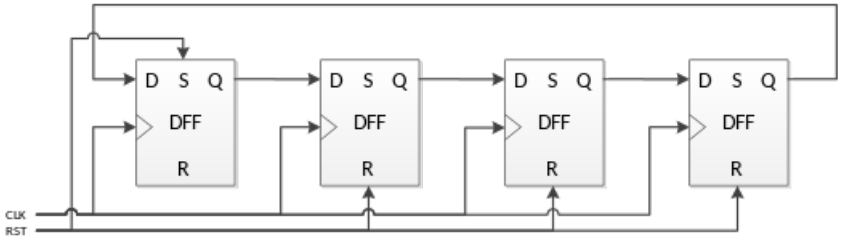
- asychronní (nejstejný clock), synchronní

Straight Ring Counter (**Overbeck Counter)**

- používá do kruhu spojený posuvný registr kde cirkuluje jedna jednička. Nevýhoda je nutnost přednastavení jedničky po resetu

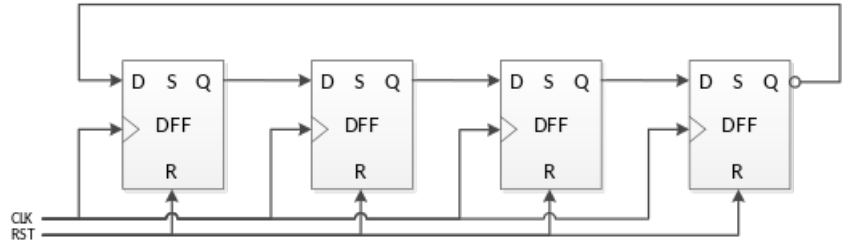






Twisted Ring Counter (Johnson Counter) (Mobius Counter)

- do kruhu spojený registr s invetovaným přenosem mezi nejvyšším a nejnižším bitem



Straight ring/*Overbeck counter*				
State	Q0	Q1	Q2	Q3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
0	1	0	0	0

Twisted ring/*Johnson counter*				
State	Q0	Q1	Q2	Q3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
0	0	0	0	0