STU FIIT

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Dátové štruktúry a algoritmy

Výpočtovo intenzívne úlohy a HW akcelerácia

5. 5. 2021

letný semester 2020/2021

prednášajúci: Lukáš Kohútka

Výpočtovo intenzívne úlohy

- Algoritmy s vysokou výpočtovou **zložitosťou**
  - $O(N)$, $O(N.\log(N))$, $O(N^2)$, $O(N^3)$, $O(2^N)$, …
- Operácie, ktoré sa používajú **často**
- Možnosti riešenia príliš vysokej záťaže:
  - Použiť algoritmus s nižšou zložitosťou (ak existuje)
  - Optimalizovať implementáciu (zvyčajne nevýrazné)
  - Znížiť frekvenciu použitia alebo množstvo dát
  - Aplikovať algoritmy AI, evolučné algoritmy, štatistické modely, negarantovať správny výsledok na 100% alebo vedome znížiť kvalitu výsledku
  - Použiť výkonnejší HW
  - Paralelizmus, HW akcelerácia

Paralelizmus

- Existujú rôzne formy:
  - Viac-vláknovosť (multithreading)
  - Viac-úlohovosť (multiprocessing)
  - GPU
  - HW akcelerácia
- Namiesto toho, aby sme program vykonali inštrukciu za inštrukciou (sekvenčne), vykonáme ho (alebo nejakú jeho časť) **paralelne**
- Väčší problém rozdelíme na menšie (ideálne rovnako veľké) časti a každé vlákno/proces/jadro GPU vyrieši jednu časť

HW akcelerácia

- Nahradenie softvérovej implementácie vybranej funkcionality (algoritmu) implementáciou hardvérovou
  - Môže byť úplné alebo čiastočné (HW/SW Co-Design)
- SW sa vykonáva sekvenčne, krok za krokom, alebo inštrukcia za inštrukciou
- HW dokáže využívať **paralelizmus** v najväčšej možnej miere
- Možnosti realizácie:
  - **ASIC (**Application Specific Integrated Circuit) – vlastný čip vyrobený na mieru podľa návrhu
  - **FPGA** (Field Programmable Gate Array) – existujúci čip, ktorý sa konfiguruje podľa návrhu

Výhody a nevýhody HW akcelerácie na čipe

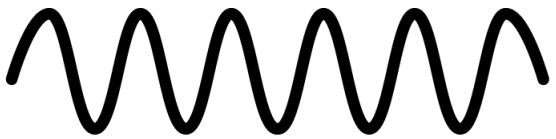
- Výhody:
  - Najvyššia priepustnosť
  - Najnižšia latencia
  - Najnižšia spotreba energie
  - Možnosť dosiahnutia lepších výsledkov/rozhodnutí vďaka realizácii komplexnejších a robustnejších algoritmov, ktoré môžu vo forme SW byť príliš pomalé
- Nevýhody:
  - Vývoj je náročnejší, zdĺhavejší a drahší
  - Nižšia flexibilita na zmeny požiadaviek (len ASIC)
  - HW navyše (niečo stojí)

HW/SW Co-Design

- Možnosti kombinácie HW a SW:
  - ASIC pozostávajúci z CPU a koprocesora (HW akcelerátor) – najdrahšie, ale najefektívnejšie riešenie
  - FPGA pozostávajúci zo soft-core CPU IP jadra a koprocesora – najlacnejšie riešenie, avšak CPU realizované v FPGA nedosahuje štandardný výkon (100-400 MHz)
  - FPGA SoC – hard-core CPU (zvyčajne ARM) a FPGA na jednom čipe – kompromis ASIC a FPGA
  - HW akcelerátor na samostatnom čipe (ASIC/FPGA) a existujúci CPU čip, spolu na jednej doske plošných spojov – pomalšia komunikácia, vysoká latencia

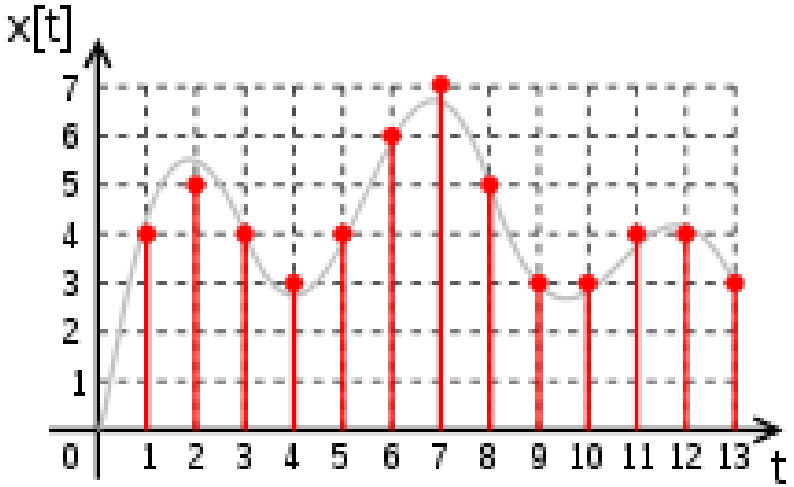
Integrované obvody

- Integrovaný obvod = mikročip
- Mikročipy z hľadiska signálu môžu byť:
  - Digitálne (digital design) – jednotky a nuly, čísla
  - Analógové (analog design) – napätie, prúd
  - Zmiešané (mixed-signal design) – kombinácia oboch svetov na jednom čipe



Analog Signal

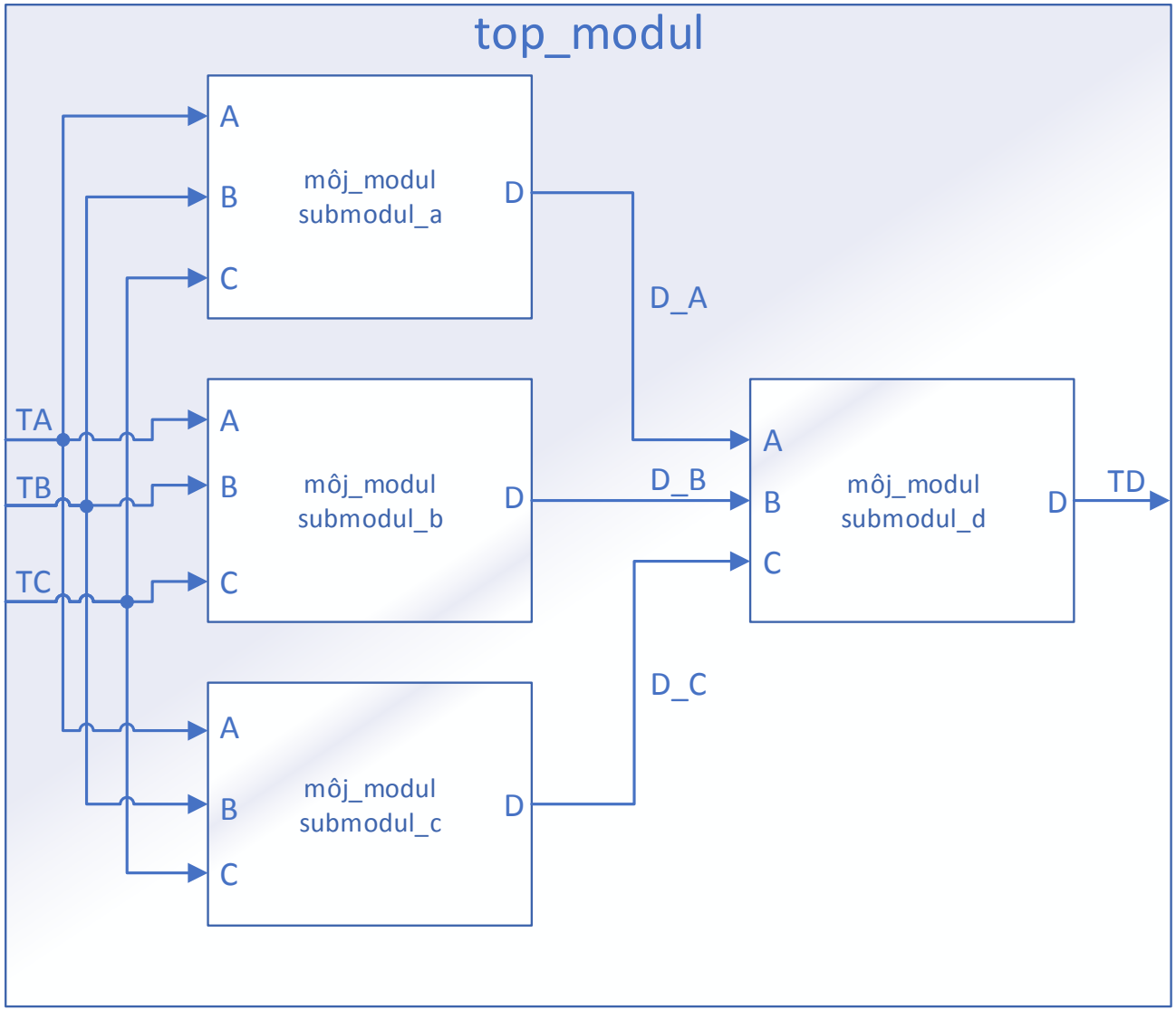
Digital Signal



Číslicový návrh a formy jeho opisu

- Pri návrhu sa často používajú **diagramy**, podobne ako pri návrhu softvéru, kde sa používajú najmä UML diagramy
  - Blokové diagramy – pre dekompozíciu časti dizajnu (modulu) na komponenty (submoduly)
  - Stavové diagramy – pre opis stavových automatov
  - Diagramy aktivít – pre vyjadrenie rozhodovacej logiky
  - Časové diagramy – pre opis komunikačných protokolov
- Okrem diagramov sa opisuje číslicový systém v podobe **RTL kódu** (podobne ako pri programovaní, len opisujeme HW, nie SW):
  - VHDL, Verilog, **SystemVerilog**, Chisel, SpinalHDL, ...

Blokový diagram - príklad



SystemVerilog - príklad

```
module top_modul (
  input TA,
  input TB,
  input TC,
  output TD
);
wire D_A, D_B, D_C;
môj_modul submodul_a (
  .A (TA),
  .B (TB),
  .C (TC),
  .D (D_A)
);
môj_modul submodul_b (
  .A (TA),
  .B (TB),
  .C (TC),
  .D (D_B)
);
môj_modul submodul_c (
  .A (TA),
  .B (TB),
  .C (TC),
  .D (D_C)
);
môj_modul submodul_d (
  .A (D_A),
  .B (D_B),
  .C (D_C),
  .D (TD)
);
endmodule
```

Zoraďovanie údajov

- Vstup: množina viac-bitových prvkov (items)
  - Pridávané prvkov paralelne (naraz)
  - Alebo sekvenčne (za sebou)
- Prvok (item) sa skladá z 2 častí:
  - Kľúč (data)
  - Payload (napr. ID)
- Výstup:
  - Zoradená množina prvkov
  - Alebo MIN/MAX prvok
- Realizovateľné softvérovo alebo hardvérovo

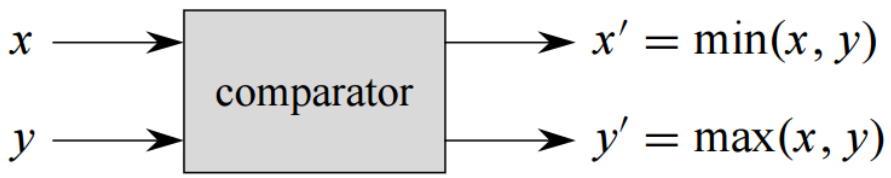
Dôležité parametre pri HW zoraďovaní

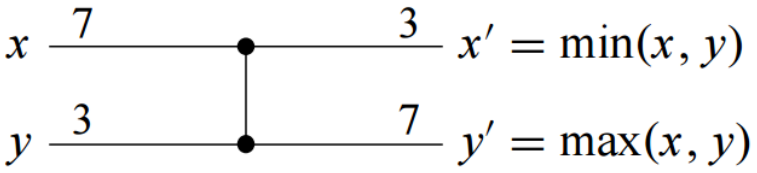
- Maximálny počet prvkov, ktoré je možné zoradiť
- Veľkosť prvkov = počet bitov na 1 prvok
- Smer zoradenia, t.j. MIN alebo MAX
- Či je vstup 1 prvok (sekvenčne) alebo všetky prvky (paralelne)
- Či je výstup len MIN/MAX prvok (sekvenčne) alebo kompletná množina prvkov (paralelne)
- Či je možné vybrať (odstrániť) len MIN/MAX prvok alebo ľubovoľný prvok podľa ID

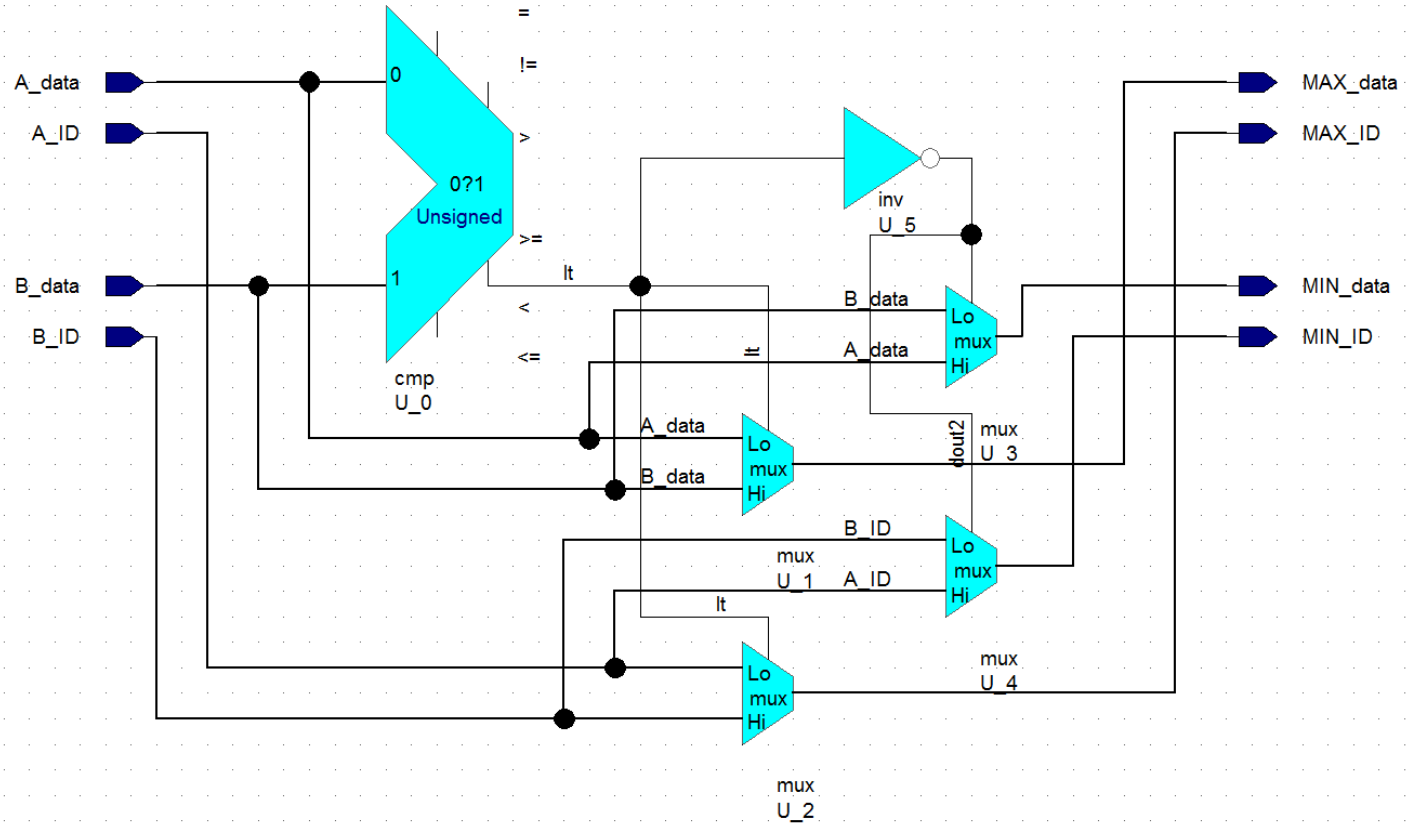
Dôležité vlastnosti

- Výkon, ktorý pozostáva z:
  - Max. pracovná frekvencia
  - Priepustnosť
  - Latencia
- Množstvo potrebného hardvéru (plocha čipu)
  - Cena (výrobné náklady)
  - Veľkosť (fyzické rozmery)
  - Spoľahlivosť (odolnosť voči poruchám)
- Spotreba energie
  - Absolútna / relatívna (na výkon)
  - Statická / dynamická

Sorting node

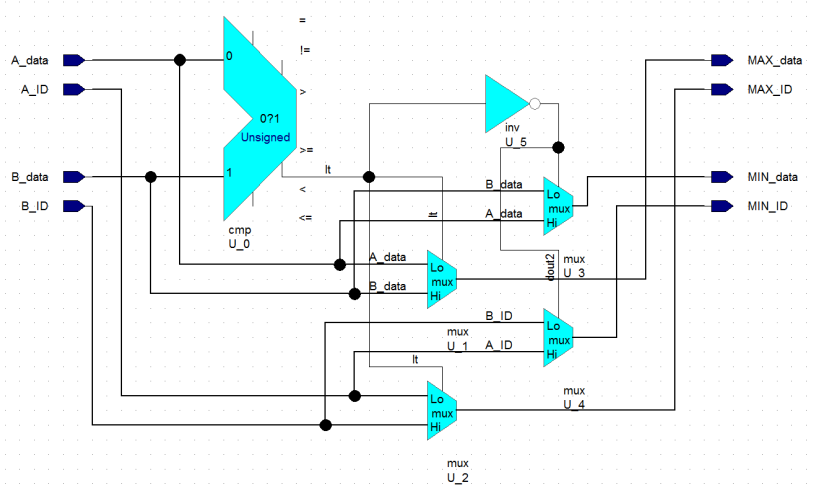


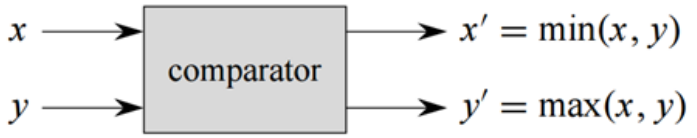




Sorting node - SystemVerilog

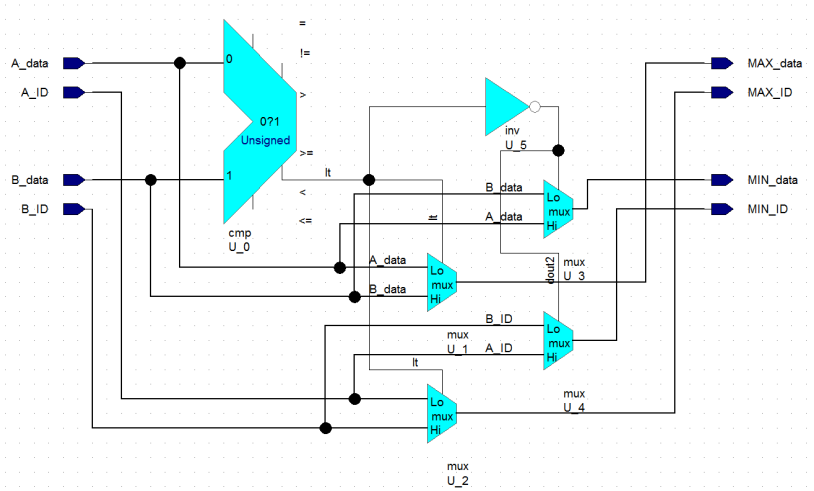
```
1  module sorting_node #(
2    parameter DATA_W = 16,
3    parameter ID_W = 8
4  )(
5    input  logic [DATA_W-1:0] A_data,
6    input  logic [  ID_W-1:0] A_ID,
7    input  logic [DATA_W-1:0] B_data,
8    input  logic [  ID_W-1:0] B_ID,
9    output logic [DATA_W-1:0] MAX_data,
10   output logic [  ID_W-1:0] MAX_ID,
11   output logic [DATA_W-1:0] MIN_data,
12   output logic [  ID_W-1:0] MIN_ID
13 );
14
15 wire lt;
16
17 assign lt = B_data >= A_data;
18
19 assign MAX_data = lt ? B_data : A_data;
20 assign MAX_ID   = lt ?   B_ID :   A_ID;
21
22 assign MIN_data = !lt ? B_data : A_data;
23 assign MIN_ID   = !lt ?   B_ID :   A_ID;
24
25 endmodule
```

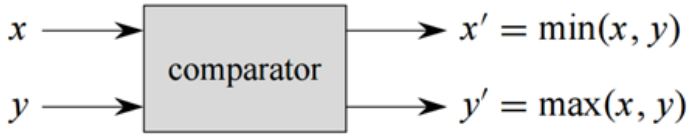




Sorting node - SystemVerilog (alternatívne)

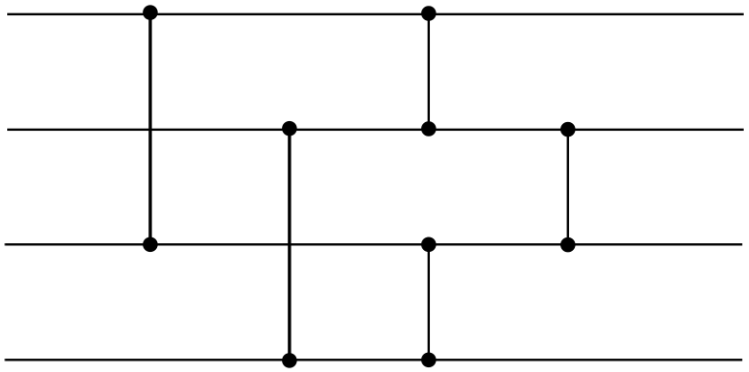
```
1  module sorting_node #(
2    parameter DATA_W = 16,
3    parameter ID_W = 8
4  )(
5    input  logic [DATA_W-1:0] A_data,
6    input  logic [  ID_W-1:0] A_ID,
7    input  logic [DATA_W-1:0] B_data,
8    input  logic [  ID_W-1:0] B_ID,
9    output logic [DATA_W-1:0] MAX_data,
10   output logic [  ID_W-1:0] MAX_ID,
11   output logic [DATA_W-1:0] MIN_data,
12   output logic [  ID_W-1:0] MIN_ID
13 );
14
15 always_comb begin
16   if (B_data >= A_data) begin
17     MAX_data = B_data;
18     MAX_ID   = B_ID;
19     MIN_data = A_data;
20     MIN_ID   = A_ID;
21   end else begin
22     MAX_data = A_data;
23     MAX_ID   = A_ID;
24     MIN_data = B_data;
25     MIN_ID   = B_ID;
26   end
27 end
28
29 endmodule
```

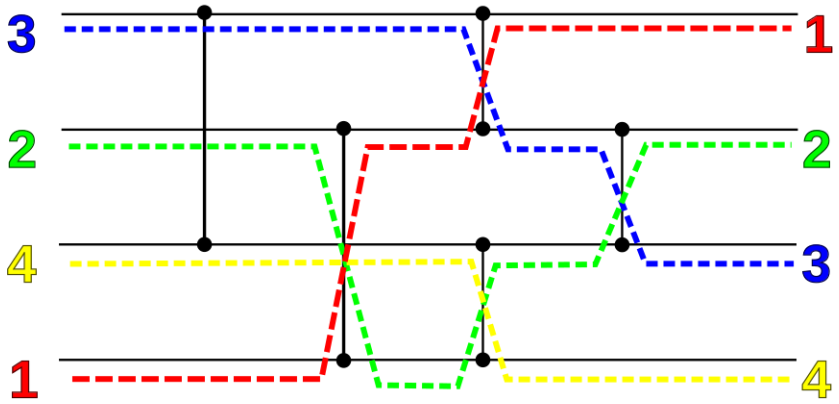




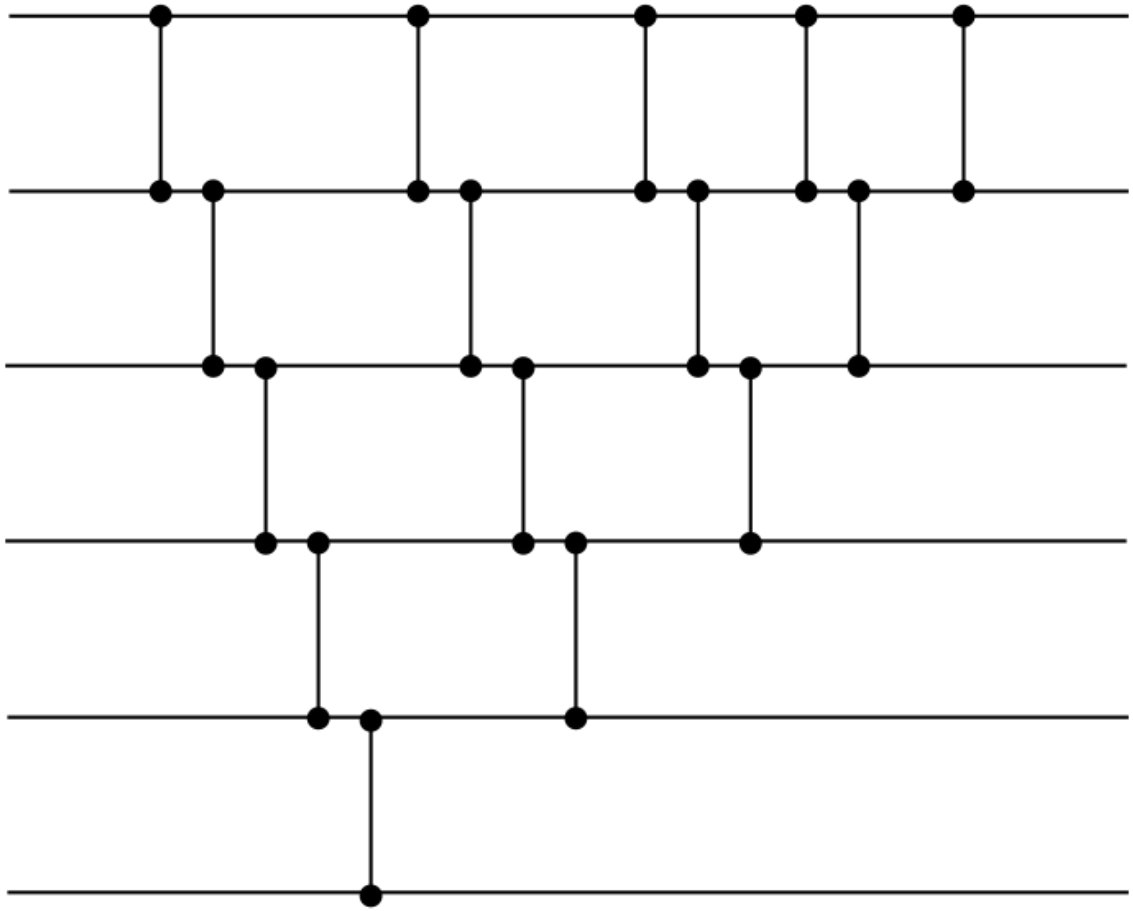
Sortovacia sieť (Sorting net)

- 1954 – Armstrong, Nelson and O'Connor
- Sieť sortovacích uzlov (sorting nodes)
- Vhodné na realizáciu paralelného sortovania
  - t.j. keď je vstup paralelný – keď sa môžu meniť všetky vstupy v rovnakom čase
- Existujú rozličné topológie

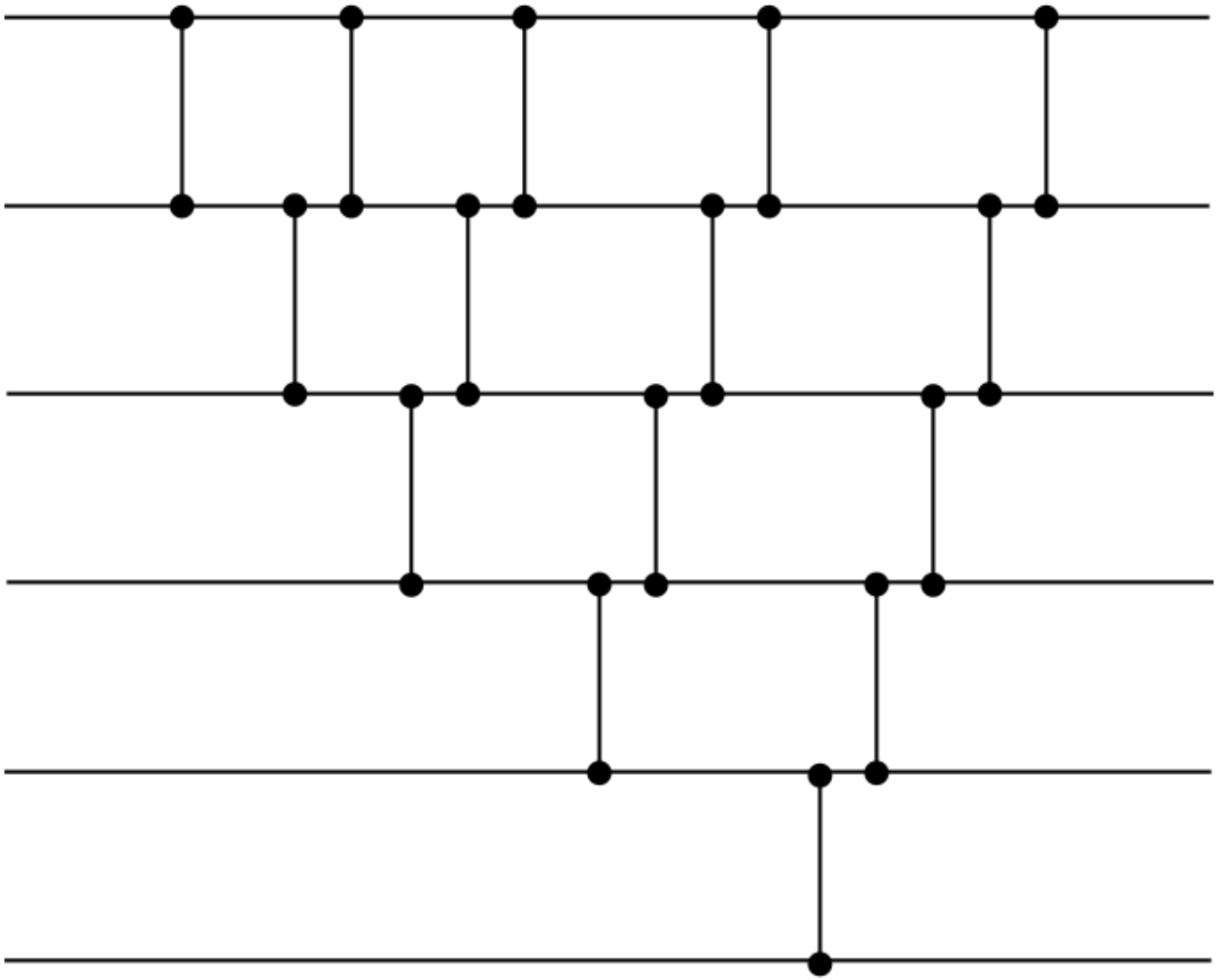




Bubble-sort topológia

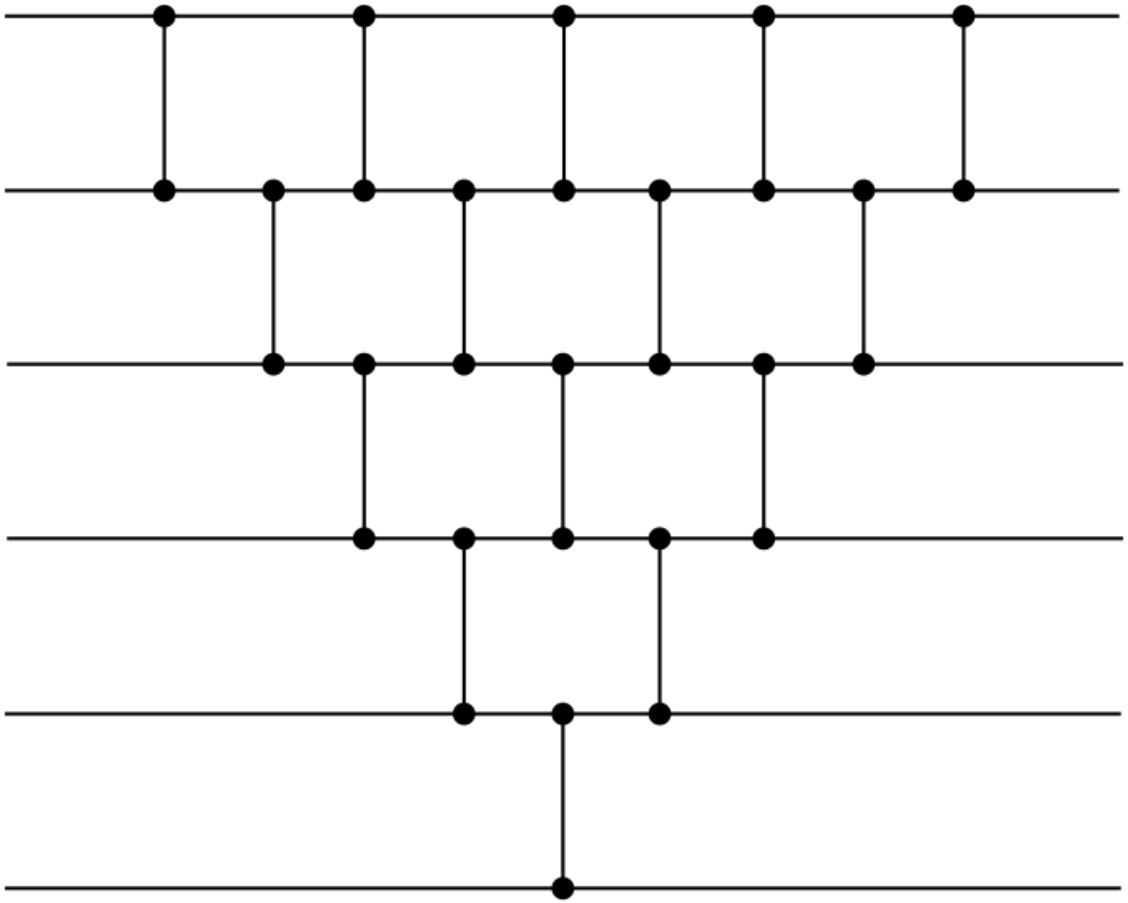


Insertion sort topológia



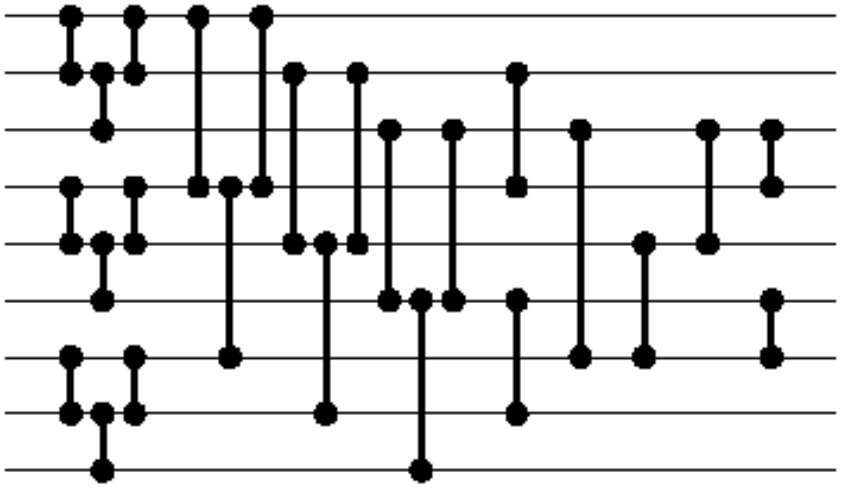
HW topológia

- HW verzia bubble-sort a insert sort je rovnaká



Optimálna sortovacia sieť

- Minimálny počet sortovacích uzlov
- Závisí od počtu vstupov



n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Depth[10]	0	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9	10
Size, upper bound[11]	0	1	3	5	9	12	16	19	25	29	35	39	45	51	56	60	71
Size, lower bound (if different)[11]											33	37	41	45	49	53	58

Sekvenčný vstup

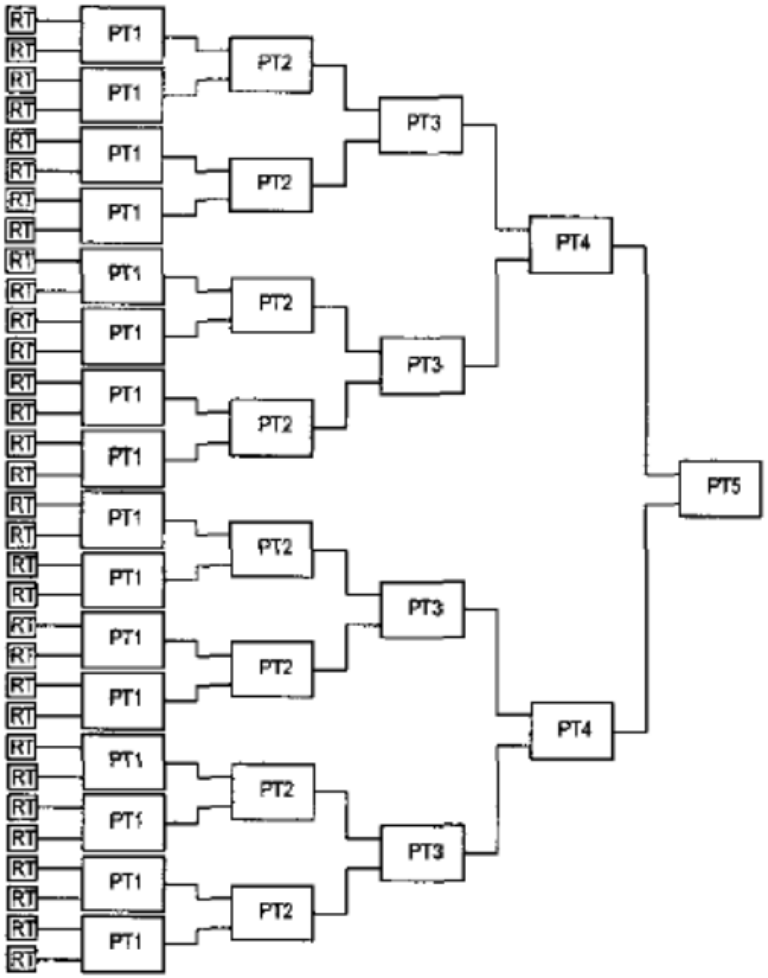
- Keď sa mení max. 1 prvok (vstup) v čase
  - inštrukcia INSERT (vloží nový prvok) alebo REMOVE (vymaže prvok podľa ID)
- Existujú viaceré architektúry
- Každá ma svoje výhody a nevýhody
- Najpopulámejšie architektúry:
  - FIFO Array
  - Sorting Tree
  - Pipelined Sorting Tree
  - Shift Registers
  - Systolic Array

FIFO Array

- Pre každú možnú hodnotu existuje 1 FIFO buffer
- Hĺbka (kapacita) každého FIFO buffra je rovná maximálnemu množstvu prvkov, ktoré chceme zoradiť
- Pridávanie prvkov:
  - Nový prvok sa vždy vloží do FIFO buffra prislúchajúceho hodnote (data) prvku
- Výstup:
  - Použije sa výstup prvého neprázdneho FIFO buffra
  - FIFO buffre sú fixne usporiadané
- Vhodné pre malý počet možných hodnôt

Sorting Tree

- V podstate sortovacia sieť, ktorá má tvar binárneho stromu
- Každý vstup je prvok pre jedno konkrétne ID, ktorý je uložený v D-FF
- Vhodné pre väčší rozsah hodnôt, ale malý počet prvkov

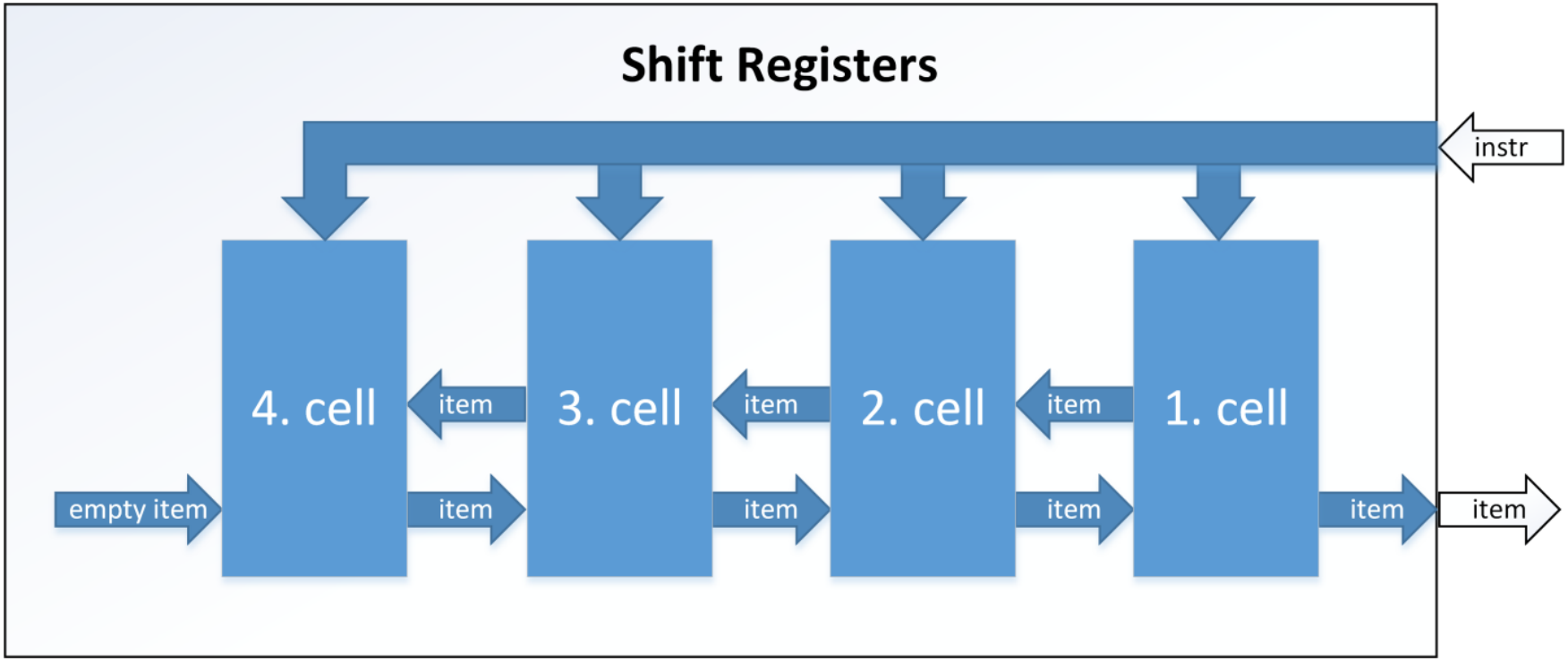


Pipelined Sorting Tree

- To isté ako Sorting Tree, ale každý sortovací uzol je registrovaný
- Vyriešený problém dĺžky kritickej cesty (funguje aj pre vyššie pracovné frekvencie)
- Vyššia spotreba registrov (väčšia plocha aj spotreba energie)
- Spracovanie vstupu už netrvá 1 takt, ale $log_2(N)$

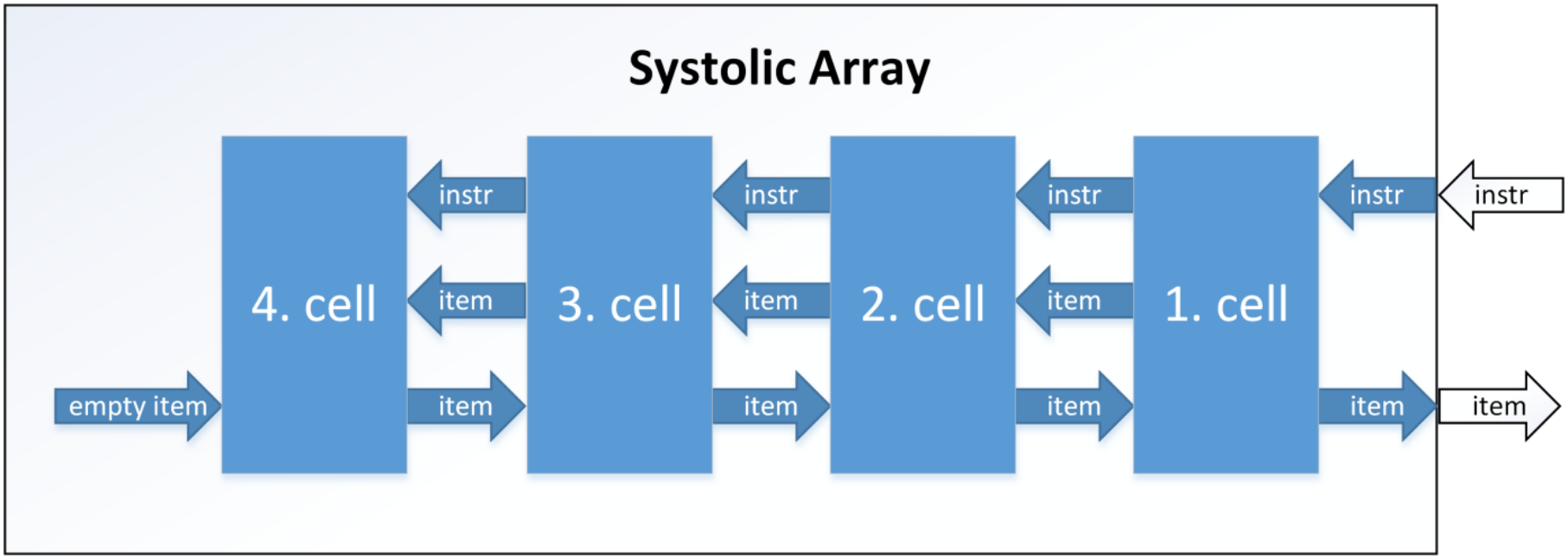
Shift Registers

- Bunky so spoločnou zbernicou na vstupe
- Výmena prvkov medzi susednými bunkami tak, aby boli prvky zoradené



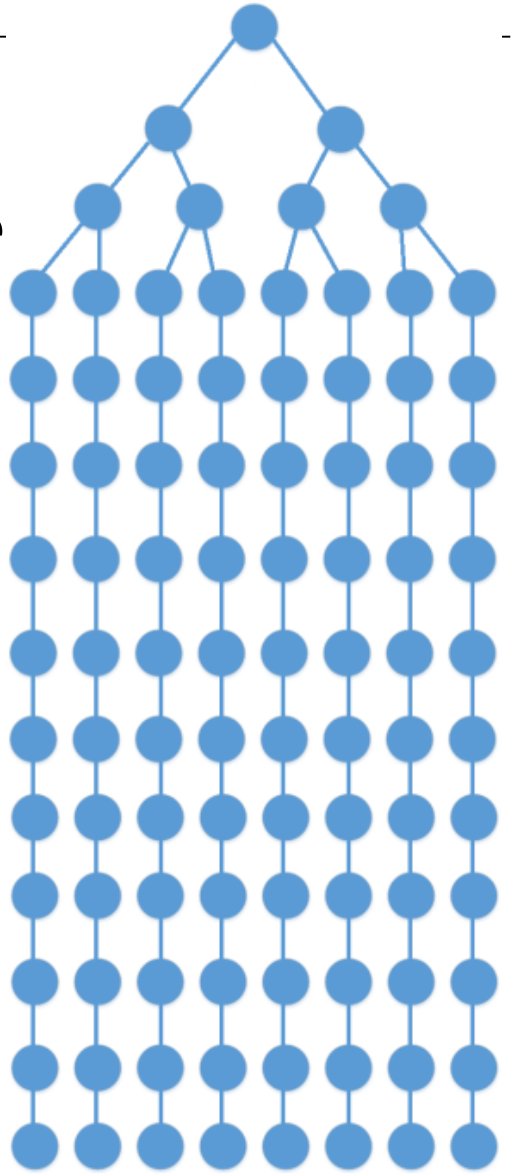
Systolic Array

- Ako Shift Registers, len nemá spoločnú zbernicu
- Inštrukcia s novým prvkom (alebo mazanie) sa posúva postupne, o 1 bunku ďalej v každom takte



Rocket Queue

- Vstup aj výstup zhora
- 2 typy buniek: duplikujúce a zlúčené
- Organizované do levelov
- Bunky rovnakého levelu zdieľajú komparátor
- Inštrukcia prechádza zhora nadol 1 level za 1 takt
- Pridávanie prvkov je vyvažované



Heap Queue

- Modifikácia Rocket Queue
- Používajú sa iba duplikujúce bunky
- Prvky v bunkách sú uložené do SRAM pamäte
- Možnosť odstrániť len MIN/MAX prvok (na samotnom vrchole Heap Queue)
- Ideálna architektúra pre zoraďovanie veľkého množstva dát (t.j. veľká kapacita) a pre veľký rozsah dát
- Kapacita môže byť len $2^N - 1$

Porovnanie ceny implementácie v FPGA

Kapacita	Heap Queue (LUTs)	Heap Queue (RAM)	Rocket Queue (LUTs)	Systolic Array (LUTs)	Shift Registers (LUTs)	Sorting Tree (LUTs)	Pipelined Sorting Tree (LUTs)
7	742	144	456	478	438	346	440
15	1103	296	822	1148	955	828*	980
31	1444	912	1490	2461	1955	1721*	2152
63	1868	2184	3488	4114	3767	3570*	4402
127	2196	4864	7494	8418	7826*	X	9157
255	2813	10344	15540	21769	X	X	18738
511	3303	21584	31743	44416	X	X	37354
1023	3701	44600	67322	X	X	X	X
2047	4212	91680	X	X	X	X	X
4095	4392	187976	X	X	X	X	X
8191	4992	384568	X	X	X	X	X
16383	5492	785960	X	X	X	X	X

Celkové porovnanie

	ID based Remove	Stable Sorting	Linear	Memory Type	Stable Clock Freq.	Latency (clock cycles)
Heap Queue	No	No	No	RAM	Yes	2
Rocket Queue	Yes	No	No	Reg	Yes	2
Systolic Array	Yes	Yes	Yes	Reg	Yes	2
Shift Registers	Yes	Yes	Yes	Reg	No	2
Sorting Tree	Yes	No	No	Reg	No	2
Pipelined Sorting Tree	Yes	No	No	Reg	Yes	$log_2(N)+1$
Software Min Max Heap	No	No	No	-	Yes	$A . log_2(N)+B$
Software Select Sort	Yes	Yes	Yes	-	Yes	$C . N + D$

Zaujala Vás dnešná prednáška a HW?

- Môžete sa zapísať na predmet na to zameraný
  - **Špecifikačné Prostriedky**
  - Povinne voliteľný predmet
  - Naučíte sa viac o vývoji mikročipov, mikroprocoserov a pod.
  - Budete vedieť hardvérovo akcelerovať
  - Rozšírite si obzory
  - Môžete sa rozhodnúť pre túto špecializáciu v rámci svojej budúcej kariéry
  - Môžete sa venovať vývoju softvéru so zameraním na podporu vývoja hardvéru (EDA)

Skúška

- skúška RT – 26.05. o 10:30
- skúška OT – 30.06. o 10:30
- forma:
  - písomnou formou
  - online
  - test v AIS-e

Spätná väzba

- Pre zlepšenie predmetu do budúcna...

- Čo sa vám **páčilo** na predmete?
- Čo sa vám **nepáčilo** na predmete?
- Čo bolo podľa vás málo preberané/vysvetľované?
- Čo bolo podľa vás duplicitné (voči ostatným predmetom)?
- Čo by ste zmenili, pridali alebo odstránili z predmetu?

- Akékoľvek iné nápady alebo postrehy?

- Využite tiež **evaluáciu predmetov v AIS-e**

Ďakujem za pozornosť