B35APO: Architektury počítačů

Lekce 07. Vstup a výstup

Pavel Píša
pisa@fel.cvut.cz

Petr Štěpán
stepan@fel.cvut.cz

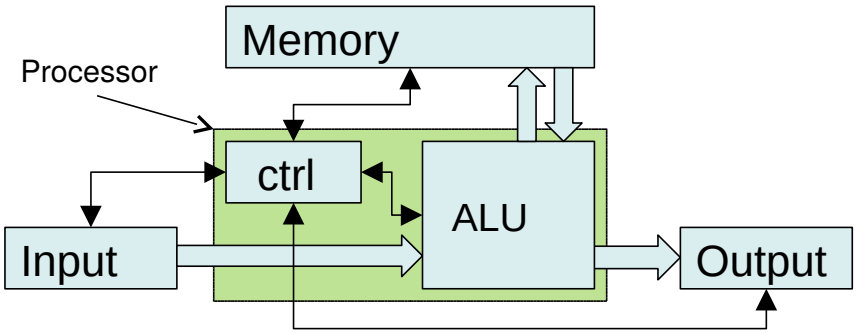
11. června, 2023

Obsah

Cíl dnešní přednášky

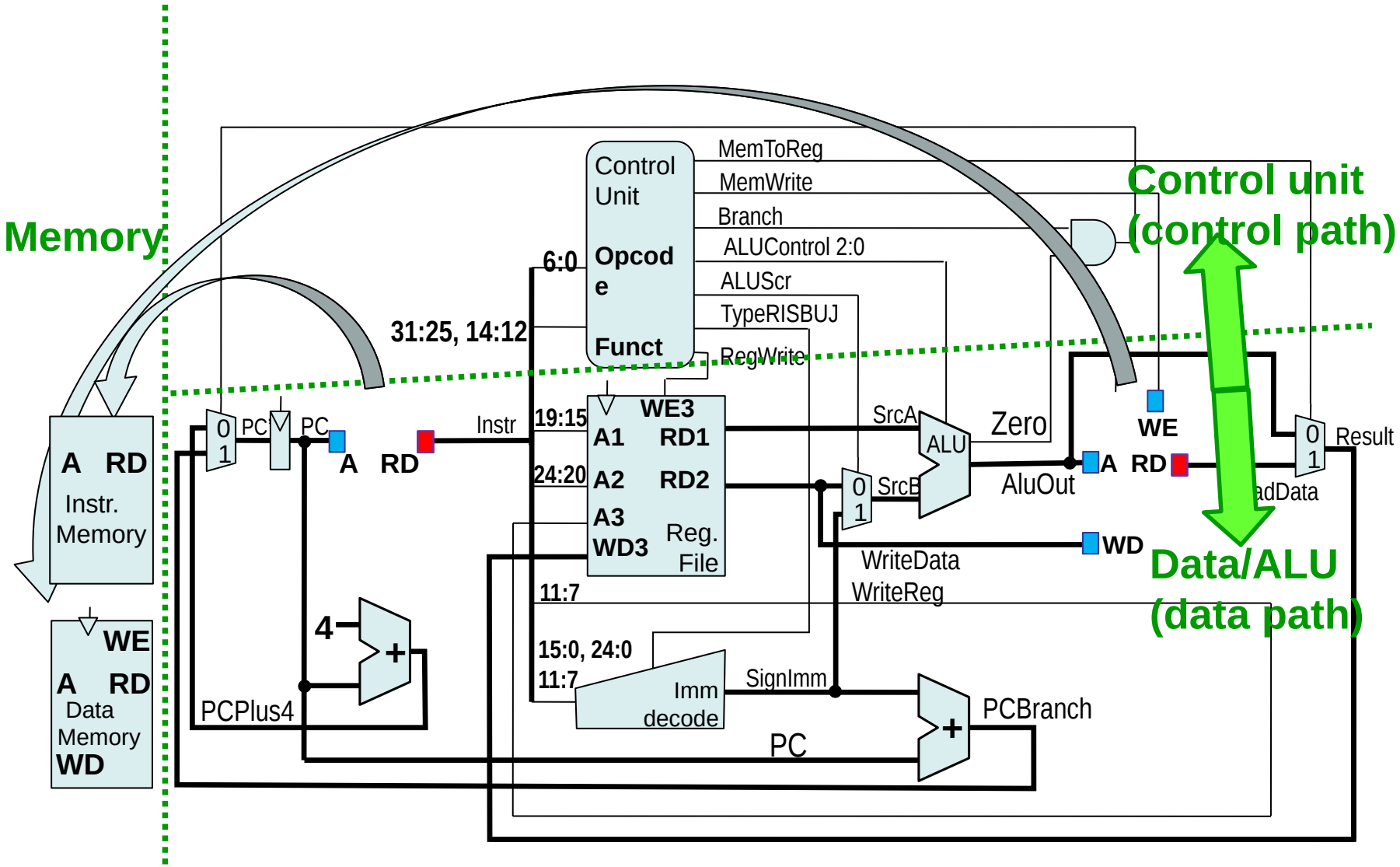
- Projít, jaké jsou možnosti vstupu a výstupu v počítači
- Periférie mapované do paměti
- Příklady v QtRvSim
- PCI a PCIe sběrnice

Architektura počítače – John von Neumann

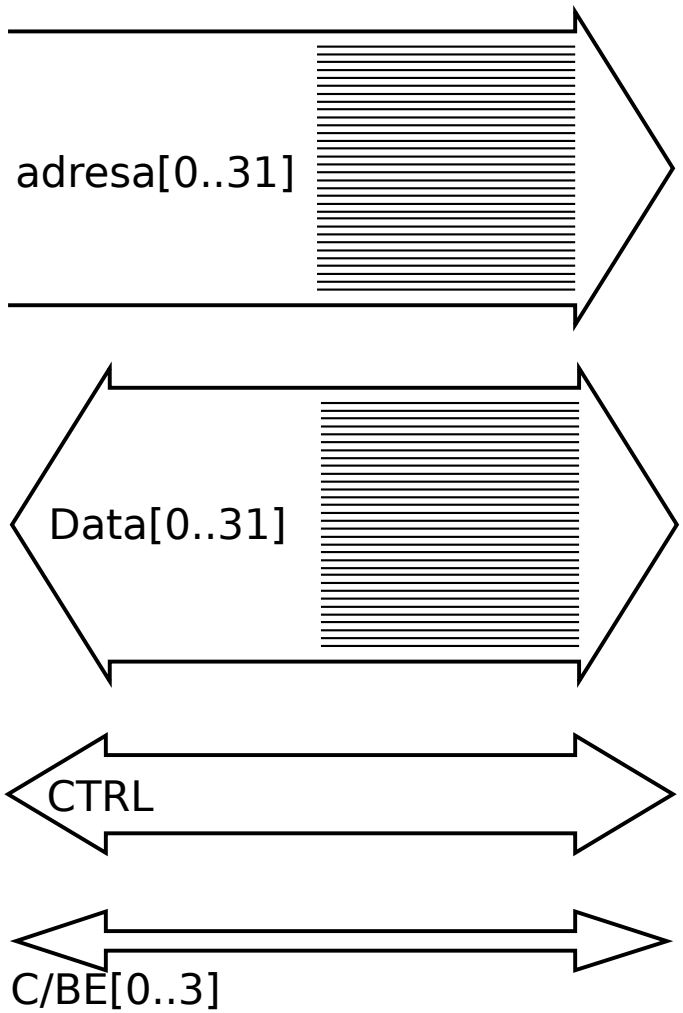


- 5 základních jednotek – řídicí jednotka, aritmeticko-logická jednotka, paměť, vstup (vstupní zařízení), výstup (výstupní zařízení)
- Architektura počítače by neměla být závislá na řešené úloze, měla by umět provádět program uložený v paměti. Program řídí, co počítač vykonává za instrukce a tím jaké dostane výsledky.
- Program a data jsou uložena ve stejné paměti, složené z buněk (jednotek) stejné velikosti. Oproti tomu Harvardská architektura měla jeden typ paměti pro program a jiný typ paměti pro data.
- Další instrukce, která se bude vykonávat je uložena v následující buňce paměti (mimo skoků v programu)
- Instrukce provádějí aritmetické a logické operace, přesuny dat z/do paměti, skoky a větvení programu a speciální řídicí instrukce.

Návrh CPU z přednášky 5



Propojení CPU s pamětí a periférie mi



- Adresová sběrnice (A0..A31) může být oddělená, nebo multiplexovaná, nebo sdílet stejné signály jako datová část
- Datová sběrnice (D0..D31) může být obousměrná, nebo oddělená pro každý směr zvlášť, paralelní nebo sériová
  - zde na obrázku je paralelní 32-bitová sběrnice typu half-duplex stejné vodiče pro oba směry
- Řídicí signály sběrnice
  - řídí komunikaci na sběrnici, co se bude kam přenášet, kdy přenos začne skončí, pozdrží.
- BE0 to 3 – pokud je povoleno čtení bajtů na sběrnici širší než 8 bitů.

Klasifikace periférií

Chování:

- jenom vstup
- jenom výstup
- vstup i výstup (v současnosti většina zařízení, i klávesnice má výstup signalizace led)

Připojení:

- přímé propojení CPU a periférie
- hierarchické – propojení přes jiné periférie (bridge, switch)

Partner:

- člověk – jiné parametry komunikace
- počítač – většinou rychlejší komunikace
- prostředí – senzory a aktuátory

Parametry přenosu:

- kapacita přenosové linky – maximální možnosti přenosu dat
- latence – čas, do kterého se provede přenos dat

Klasifikace periférií

Příklady periférií komunikujících s lidmi:

- klávesnice – sice jen vstup, ale moderní často i výstup led diody, velmi malá přenosová rychlost, latence až do 200ms (mimo hraní her)
- mikrofon/reproduktory – přenosová rychlost až 8Mb/s, latence záleží na aplikaci, pro interaktivní komunikaci se vyžaduje latence menší 500 ms, optimálně 150 – 300 ms
- tiskárny/scanery – přenosová rychlost podle připojení, na latency nezáleží (v řádech vteřin/minut)

Příklady periférií pro kounikaci mezi počítači

- modem – první modemy 115,2kb/s, LTE max 300 Mb/s, 5G až 500Mb/s
- sítě/WLAN – od 10 Mb/s přes 1Gb/s až po 1Tb/s
- datová úložistě – HDD, SSD, magnetopáskové jednotky, rychlosti komunikace podle připojení (dnes později), latence SSD nejlepší, HDD horší, magnetopáskové jednotky – použitelně pouze sekvenční zápis.

Klasifikace periférií

Příklady senzorů a aktuátorů:

- kamera, laserové hloubkoměry – rychlosti komunikace podle druhu připojení
  - USB 2.0 max 480 Mb/s,
  - USB 3.1 max 5Gb/s,
  - WLAN až 10Gb/s
- aktuátory – motory DC/PMSM
  - není důležitá rychlost přenosu, ale latence
  - latence je nejdůležitější parametr pro řízení
  - DC – latence 0.5-0.05ms,
  - PMSM – latence 0.05-0.01ms

Způsoby komunikace s periférií

Data pouze na dotaz:

- zařízení čeká, až ho CPU osloví a zašle data na výstup, nebo požádá o připravená data ze vstupu
- pomalejší komunikace, CPU musí aktivně zjišťovat, zda jsou připravená data k příjmu

Periférie využívá přerušení k signalizaci svého stavu:

- pokud dojde ke změně stavu, periférie vyvolá přerušení (přednáška 9)
- CPU pak aktivně čte, nebo zapisuje data na periférii

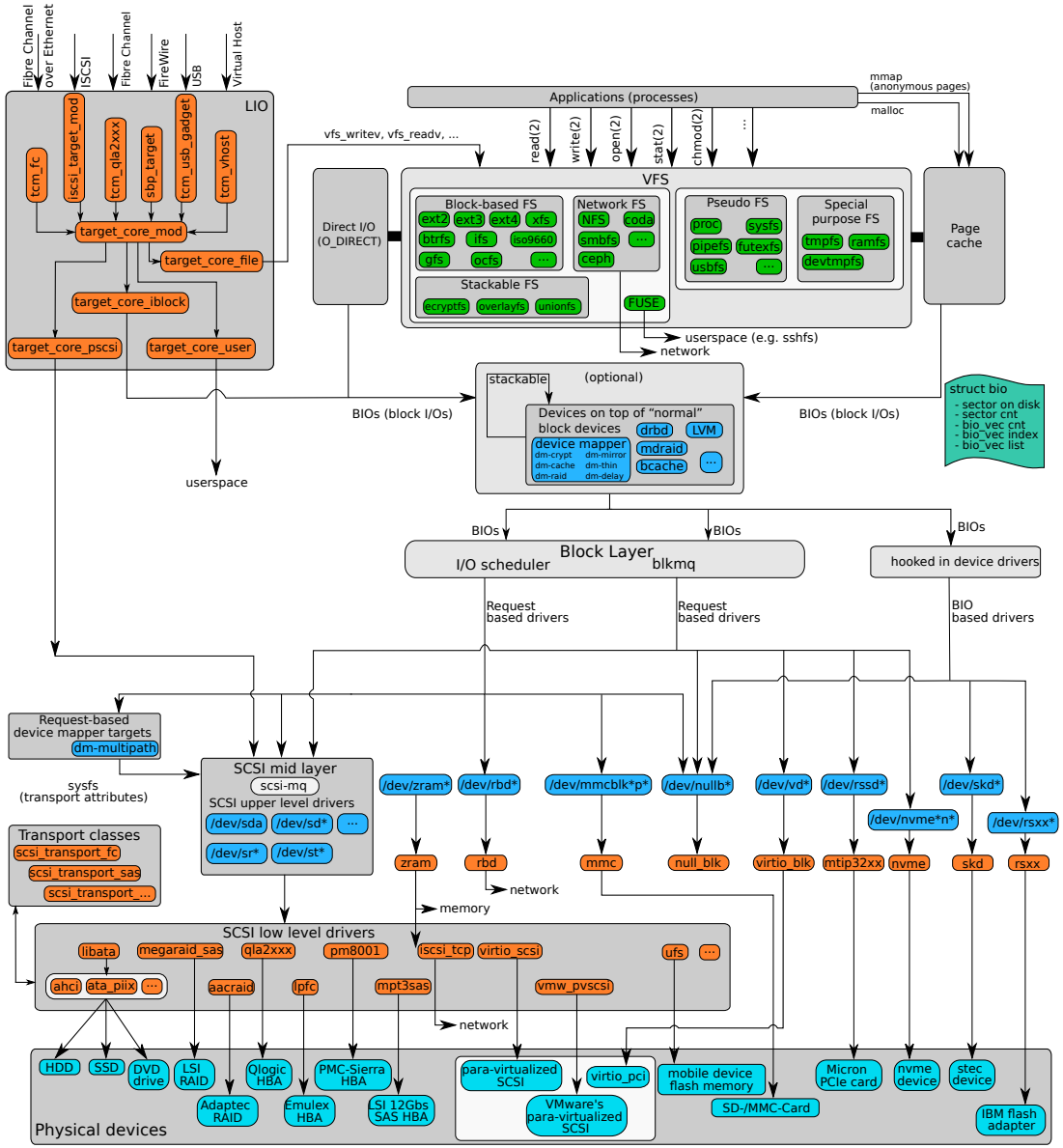
Periférie využívá přímý přístup do paměti:

- využívá také přerušení
- CPU nastaví pouze z/do jaké adresy v paměti se data budou číst/zapisovat a periférie sama řídí přenos dat
- po ukončení periférie vyvolá přerušení a CPU zkontroluje výsledek přenosu (zda bylo vše v pořádku, nebo došlo k chybě)

Zpracování periférií – Linux (zjednodušeno)

Programy komunikují s periferiemi pomocí operačního systému a systémového volání a ovladačů periferií(přehled v přednášce 10, podrobně v předmětu OSY).

Další možností je komunikace přes sdílenou paměť – náplň dnešní přednášky.



Systémová volání

Systémová volání:

- pro normální uživatele jsou systémová volání zabalená ve funkcích knihovny libc – POSIX API
- funkce open
  - pro každou periferii lze vytvořit virtuální soubor
  - tento soubor slouží ke komunikaci s periferií
- funkce read
  - přečte data od periférie jako by je četlo ze souboru
  - blokující čtení
    - pokud nejsou data funkce read čeká na jejich příchod
    - při čekání je proces uspán a nezatěžuje CPU
  - neblokující čtení
    - pokud nejsou data, funkce read vrátí zápornou hodnotu
    - koordinace čtení je ovládána procesem
    - data z periférie jsou ukládána nezávisle ovladačem do vnitřních systémových bufferů

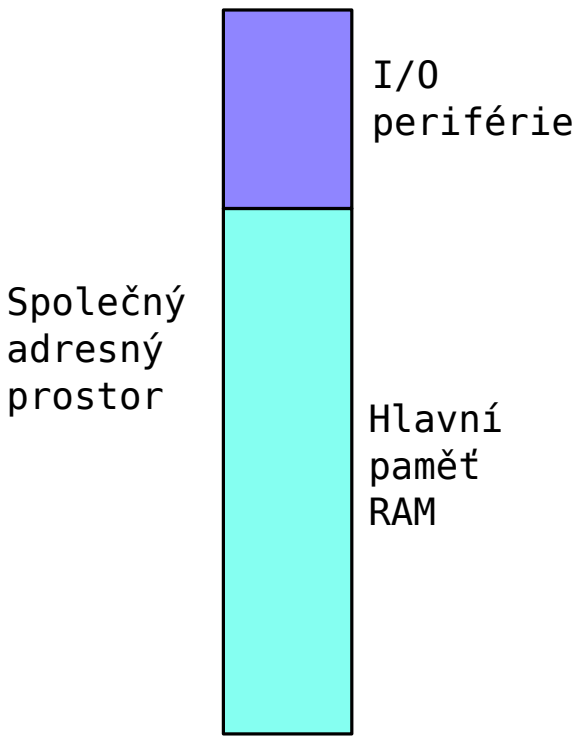
Přístup CPU k periférii

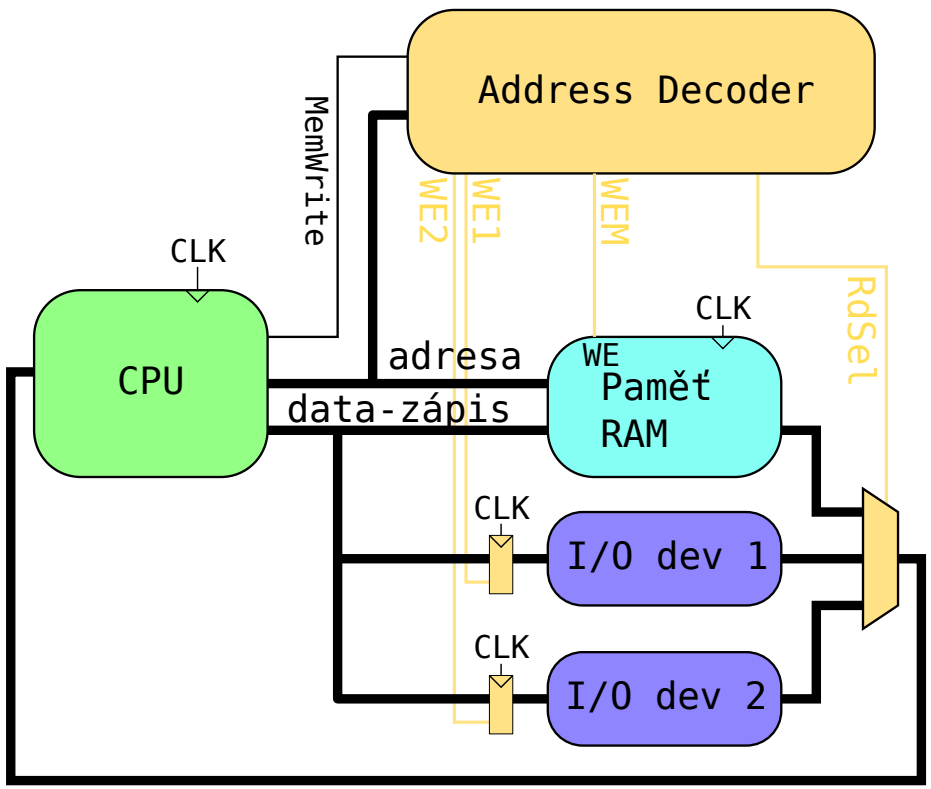
Prakticky existují dva různé přístupy:

- Speciální instrukce
  - x86 používá instrukce `in`, `out`.
  - Tyto instrukce mají podobný formát jako přístup do paměti, ale data čtou a zapisují na sběrnici kde jsou připojeny periférie.
  - Implementace těchto instrukcí potřebuje přístup k jiným sběrnicím.
  - Tyto instrukce se ukázaly neefektivní, nesystematické a moderní periférie je nepoužívají.
- Vyhrazení části adresového prostoru
  - RISC-V nemá speciální instrukce pro komunikace s periférierniemi, a proto využívá metodu, že čtení a zápis do určených oblastí paměti vede k čtení a zápisu z periferií.
  - Tento přístup je velmi jednoduše konfigurovatelný, periférie při inicializaci získají paměťový rozsah, který slouží k přesunu dat mezi CPU a periférierniemi.

Paměťově mapované periférie

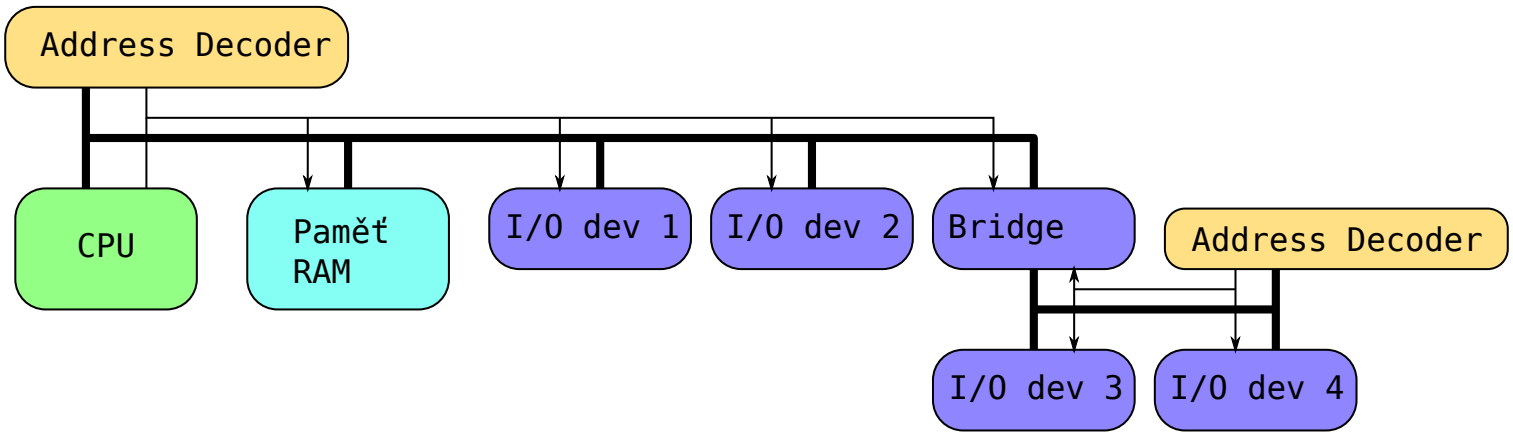
- RISC-V nemá speciální instrukce pro komunikace s periférierniemi
- pro komunikaci s periférierniemi se využívá ukládání a čtení z paměti
- Address Decoder – rozhoduje, kam se data přesměrují



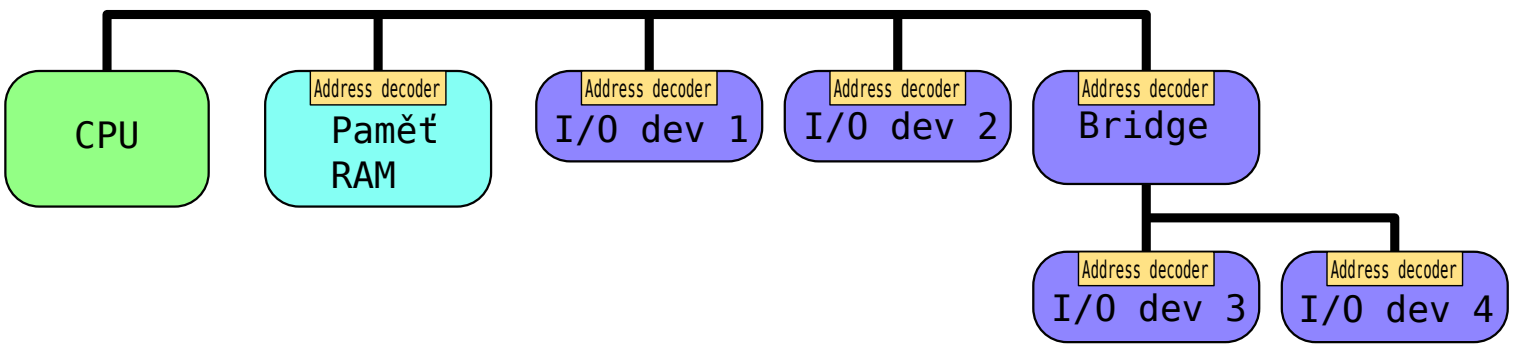


Typy adresových dekodérů

Centralizovaný dekodér



Decentralizovaný – autonomní dekodér



Kvíz otočné voliče

RISC-V přistupuje k perifériím

A speciální instrukcí `in`, `out`

B provedením dvojice instrukcí `sw`, `lw` hned po sobě

C normálním provedením instrukce `lw`, `sw` na speciální adresy

D zápisem mimo adresový prostor 32-bitového prostoru

Obsah

Asynchronní sběrnice vs. synchronní sběrnice

Asynchronní sběrnice:

- dvě základní varianty:
  - Začátek a konec každého bitu je detekovatelný druhou stranou
  - Je dohodnuta doba trvání vyslání jednoho bitu a jednotlivé bajty mají druhou stranou detekovatelný začátek, případně i konec
- Příkladem asynchronní komunikace je sériový port, USB, SATA disky

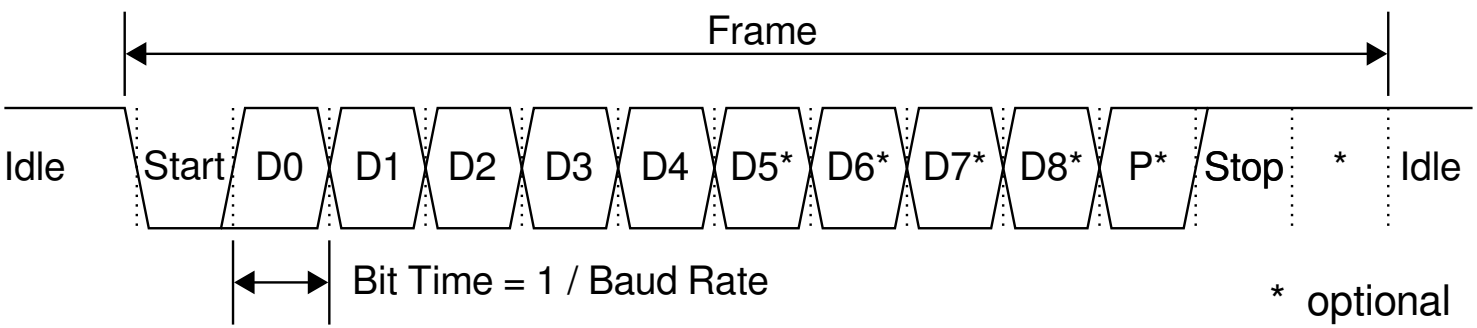
Synchronní sběrnice:

- Nejjednodušší je vyhradit jeden signál na přenos hodin vysílajícího k přijímajícímu
- Vyslání data se řídí hodinami, buď jen jedním typem hrany, nebo i na druhý typ hrany
- Příkladem synchronního přenosu jsou komunikace DDR s pamětí, PCI, PCI Expres

Sériová linka

Sériová linka (sériový port) je jedna z nejstarších způsobů komunikace využívaných dodnes.

- Asynchronní přenos bez hodin.
- Obě strany jsou nastaveny na stejnou rychlost, která definuje délku vyslání jednoho bitu
  - Přenos začíná vysláním start bitu (přechod z 1 $\rightarrow$ 0)
    - Vysláním a příjmem start bitu se synchronizují lokální hodiny všech zařízení
  - Pak se vysílají jednotlivé bity jednoho bajtu
  - Výběrově pak může následovat bit parity pro kontrolu chyb přenosu
  - Poslední je vyslán stop bit (přechod z 0 $\rightarrow$ 1, nebo přidržení 1)
- Vyslání jednoho bajtu tedy obsahuje vyslání 10-11 bitů
- Běžné rychlosti, dříve 9600 Bd až 115200 Bd, nově až 921600 Bd (Bd – Boud = bit per second)



Sériová linka

Základní specifikace RS 232:

- Navrženo pouze pro propojení dvou zařízení
- Obě zařízení jsou propojené signální zemí
- 0 reprezentována +3 – +15V, 1 reprezentuje -3 – -15 V
- Plně duplexní, tzn. jiné signály pro přenos tam a jiné pro přenos zpět
- Mohou být použité signály pro pozdržení vysílání, nebo příjmu při přeplněném lokálním bufferu

Základní specifikace RS 422:

- Plovoucí signál, Rx+, Rx- – vysílaná hodnota je napětí mezi dvěma vodiči, lze použít až na vzdálenost 1200m
- Plně duplexní, tzn. jiné signály pro přenos tam a jiné pro přenos zpět
- Může být více posluchačů pro jednoho vysílajícího

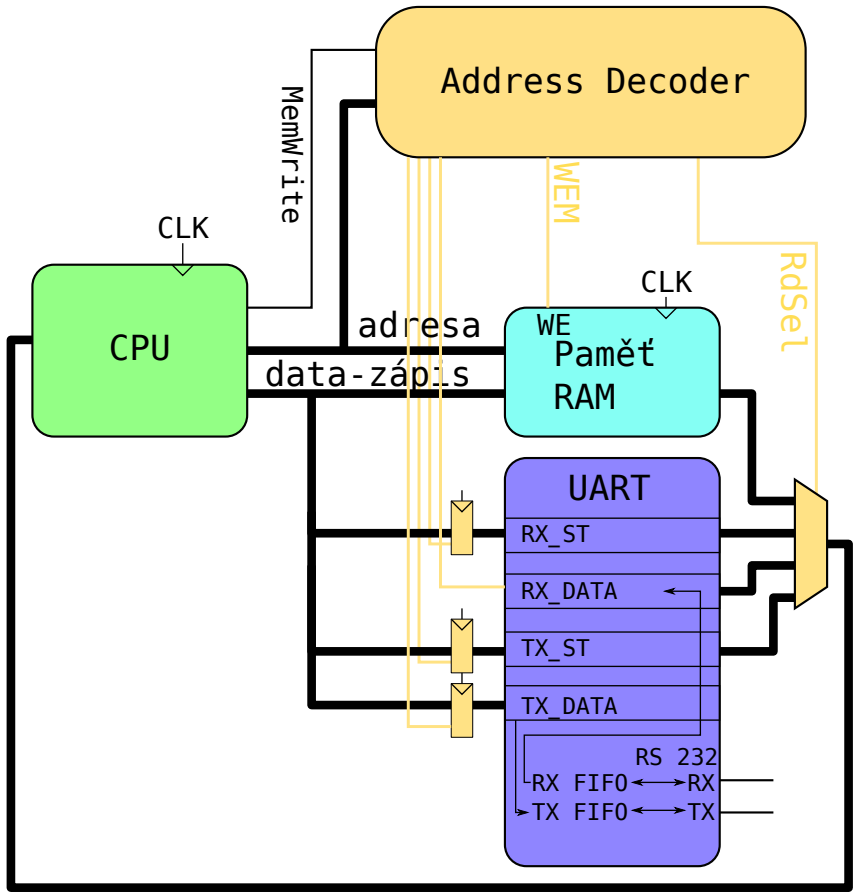
Základní specifikace RS 485:

- Plovoucí signál stejně jako 422
- Může být half-duplex - tedy pouze dva vodiče, je nutné po vyslání dat odpojit se od sběrnice a poslouchat zda někdo odpoví
- Lze více zařízení, většinou jeden poveluje, ostatní podle adresy odpovídají

UART – Universal asynchronous receiver-transmiter

UART – speciální zařízení, které přijímá a vysílá bajty po sériové lince

- RX_ST stav přijímání dat
  - bit 0 ready – byla přijata data
- RX_DATA přijatá data
  - Čtení z adresy RX_DATA současně odstraní čtená data z UARTu a vymaže příznak ready (pokud nejsou další data)
- TX_ST stav odesílání dat
  - bit 0 ready – můžeme zadávat data k odeslání
- TX_DATA data k odeslání
  - Zápis do TX_DATA vede rovnou k odesílání data



Sériový port v QtRvSim

```
.equ SERIAL_PORT_BASE,    0xffffc000
#base address of QtRVSim serial port

.equ SERP_RX_ST_REG,        0xffffc000  #Receiver status register
.equ SERP_RX_ST_REG_o,      0x0000      #Offset of RX_ST_REG
.equ SERP_RX_ST_REG_READY_m, 0x1 #Data byte is ready to be read
.equ SERP_RX_ST_REG_IE_m,     0x2 #Enable Rx ready interrupt

.equ SERP_RX_DATA_REG,    0xffffc004 #Received data byte in 8 LSB bits
.equ SERP_RX_DATA_REG_o,    0x0004   #Offset of RX_DATA_REG

.equ SERP_TX_ST_REG,      0xffffc008 #Transmitter status register
.equ SERP_TX_ST_REG_o,      0x0008   #Offset of TX_ST_REG
.equ SERP_TX_ST_REG_READY_m,  0x1 #Transmitter can accept next byte
.equ SERP_TX_ST_REG_IE_m,     0x2 #Enable Tx ready interrupt

.equ SERP_TX_DATA_REG,    0xffffc00c #Write word to send 8 LSB bits
.equ SERP_TX_DATA_REG_o,    0x000c   #Offset of TX_DATA_REG
```

Příklad vyslání znaku

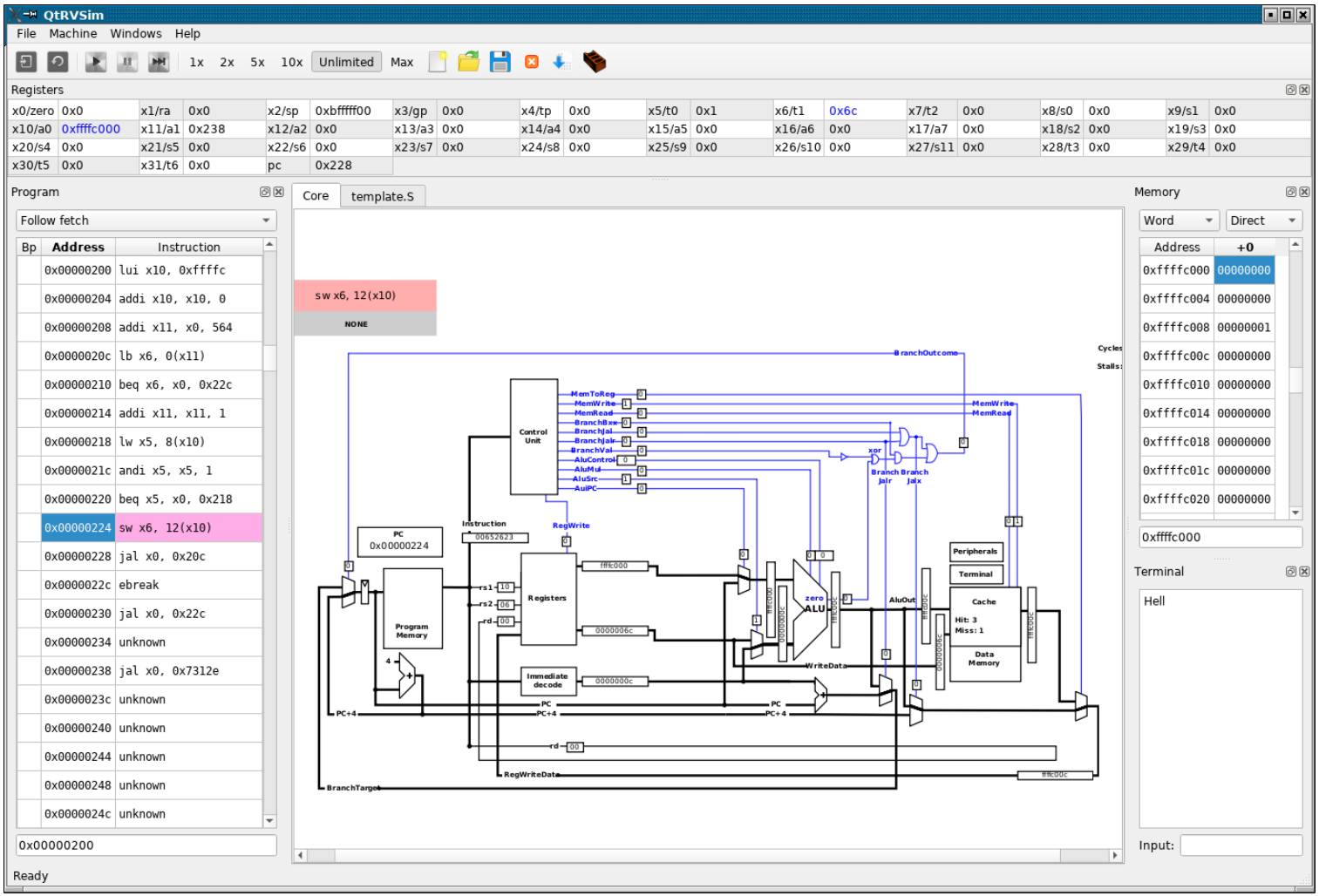
```
write:
    li   a0, SERIAL_PORT_BASE # a0 ukazuje na zacatek pameti UART
    la   a1, text_1           # nacti ukazatel na retezec
next_char:
    lb   t1, 0(a1)            # nacti znak k odeslani
    beq  t1, zero, end_char   # 0 ukoncuje retezec
    addi a1, a1, 1            # posun ukazatel
tx_busy:
    lw   t0, SERP_TX_ST_REG_o(a0)       # zjisti statu odesilaci fronty
    andi t0, t0, SERP_TX_ST_REG_READY_m # vymaskuj bit READY
    beq  t0, zero, tx_busy    # pokud neni volno v UARTU cekej zkus to znov
    sw   t1, SERP_TX_DATA_REG_o(a0) # je volno - zapis bajt
    j    next_char            # posli dalsi znak
end_char:
    ebreak                    # ukonci program

    .data
text_1:
    .asciz  "Hello world.\n"  # retezec ukonceny 0
```

Příklad přijetí znaku

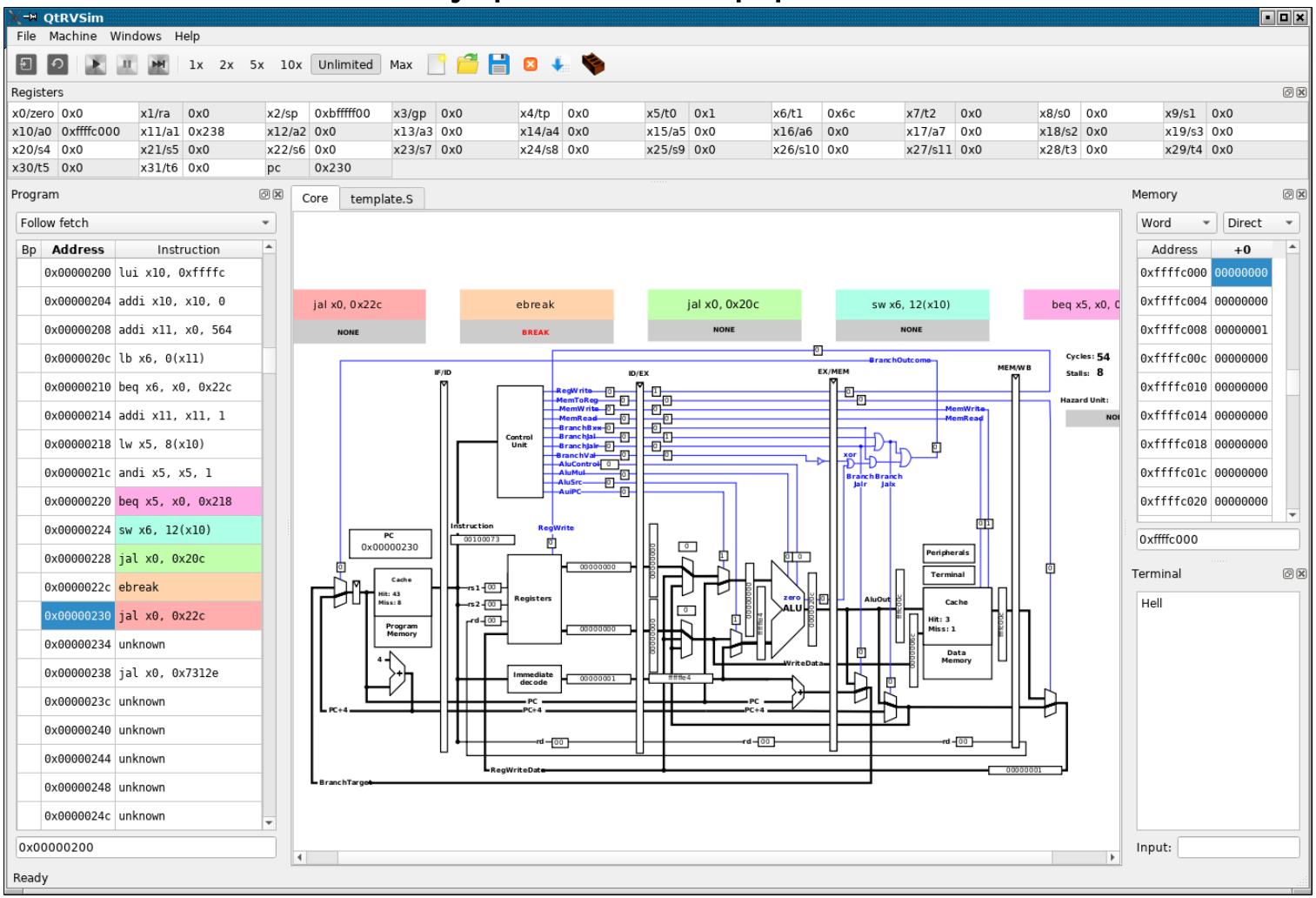
```
gets: li   a0, SERIAL_PORT_BASE # a0 ukazuje na zacatek pameti UART
    la   a1, text_1             # nacti ukazatel na buffer
    addi t2, zero, 40
next_char:
rx_not_ready:
    lw   t0, SERP_RX_ST_REG_o(a0)       # zjisti statu prijimaci fronty
    andi t0, t0, SERP_RX_ST_REG_READY_m # vymaskuj bit READY
    beq  t0, zero, rx_not_ready    # pokud neni znak UARTU cekej zkus to zn
    lw   t1, SERP_RX_DATA_REG_o(a0) # je znak - precti ho a tim odstran
    sb   t1, 0(a1)              # uloz znak do bufferu
    addi t1, t1, -13            # test je to novy radek?
    beq  t1, zero, end_char     # ukoncuje cteni
    addi a1, a1, 1              # posun ukazatel
    addi t2, t2, -1              # kontroluj kolik muzeme nacist
    bne  t2, zero, next_char
end_char:
    ebreak                      # ukonci program
    .data
text_1:
    .word 0,0,0,0,0,0,0,0,0,0
```

QtRvSim Terminal – serial port



QtRvSim Terminal – serial port

Zřetězený procesor – pipelined verze



Shrnutí komunikace periferií

- uvedený způsob komunikace je tzv. polling
  - program se neustále dotazuje, zda se něco nezměnilo, nepřišel znak, nebo se uvolnila odesílací fronta
  - toto je velmi neefektivní, zatěžuje zbytečně CPU, které by mohlo dělat něco užitečného
- na přednášce 9 bude využití přerušení
  - program může dělat něco jiného, přerušení nastane, pokud je povoleno a změní se stav periférie
  - při přerušení se začne provádět jiný program, který zjistí, jaká událost se stala a zpracuje ji
  - informace o tom co se stalo předá programu pomocí synchronizačních prostředků (bude v podrobně v předmětu OSY)

QtRvSim – otočné voliče, led

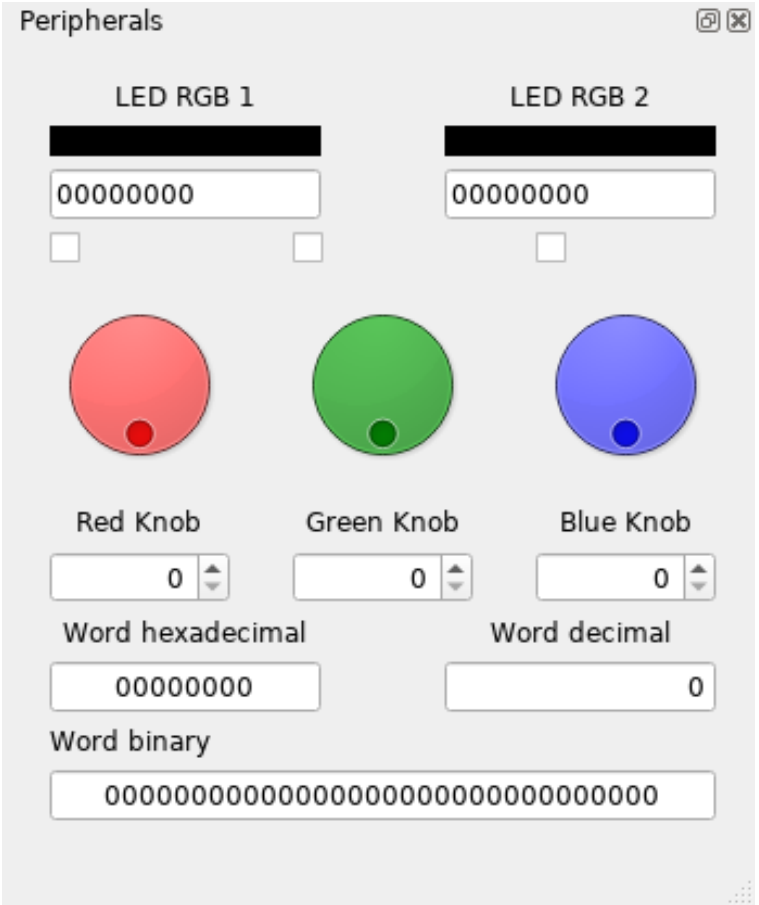
```
#base of SPILED port region
.equ SPILED_REG_BASE,        0xffffc100

#RGB LED 1 barevne slozky - 8 bitu kazda
.equ SPILED_REG_LED_RGB1,    0xffffc110
.equ SPILED_REG_LED_RGB1_o,    0x0010

#RGB LED 2 barevne slozky - 8 bitu kazda
.equ SPILED_REG_LED_RGB2,    0xffffc114
.equ SPILED_REG_LED_RGB2_o,    0x0014

#Tri 8 bitove otocne volice
#nejvyssi bajt informace o stitsknuti
.equ SPILED_REG_KNOBS_8BIT, 0xffffc124
.equ SPILED_REG_KNOBS_8BIT_o, 0x0024

#32 LEDek kazdy bit jedna LED dioda
.equ SPILED_REG_LED_LINE,    0xffffc104
.equ SPILED_REG_LED_LINE_o,    0x0004
```



Příklad použití informace od otočných voličů

```
    li   a0, SPILED_REG_BASE # a0 ukazuje na zacatek pameti pro I/O
    ori  t2, t2, -1
loop:
    lw   t0, SPILED_REG_KNOBS_8BIT_o(a0)   # nacti data od otocnych volicu
    sw   t0, SPILED_REG_LED_RGB1_o(a0)
    xor  t1, t0, t2
    sw   t1, SPILED_REG_LED_RGB2_o(a0)
    srli t0, t0, 24
    andi t0, t0, 4
    beq  t0, zero, loop    # pokud nebyl zmacknut cerveny volic

    ebreak                 # ukonci program
```

Kvíz otočné voliče

Pokud máme v proměnné `unsigned int v;` načtené slovo z adresy `SPILED_REG_BASE+SPILED_REG_KNOBS_8BIT_o` pak informace o stavu zeleného voliče získáme jako:

A `((v<<24) & 0x00ff00)`

B `((v>>8) & 0xff)`

C `(v & 0x30300030)`

D `((v>>24) & 0xf0)`

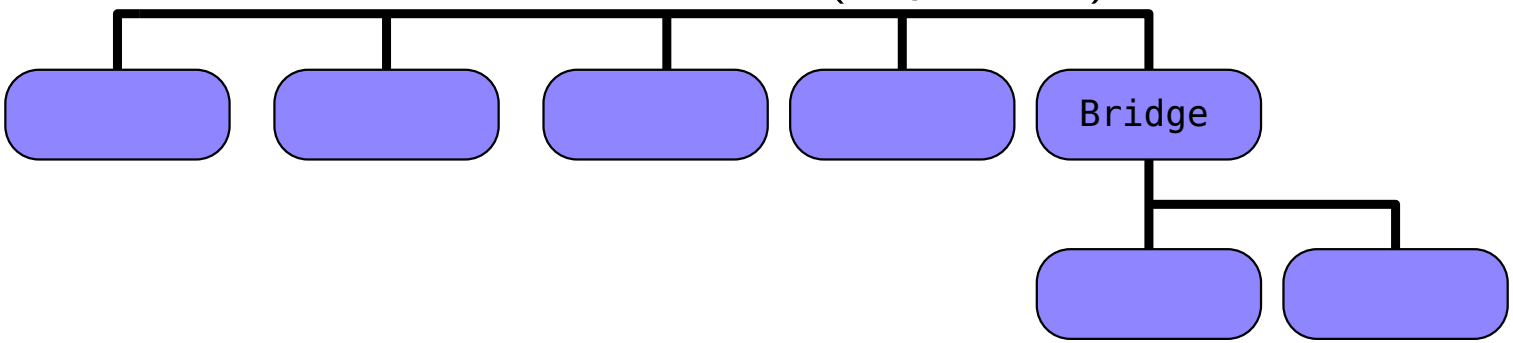
Obsah

Stručná historie vnitřních sběrnic

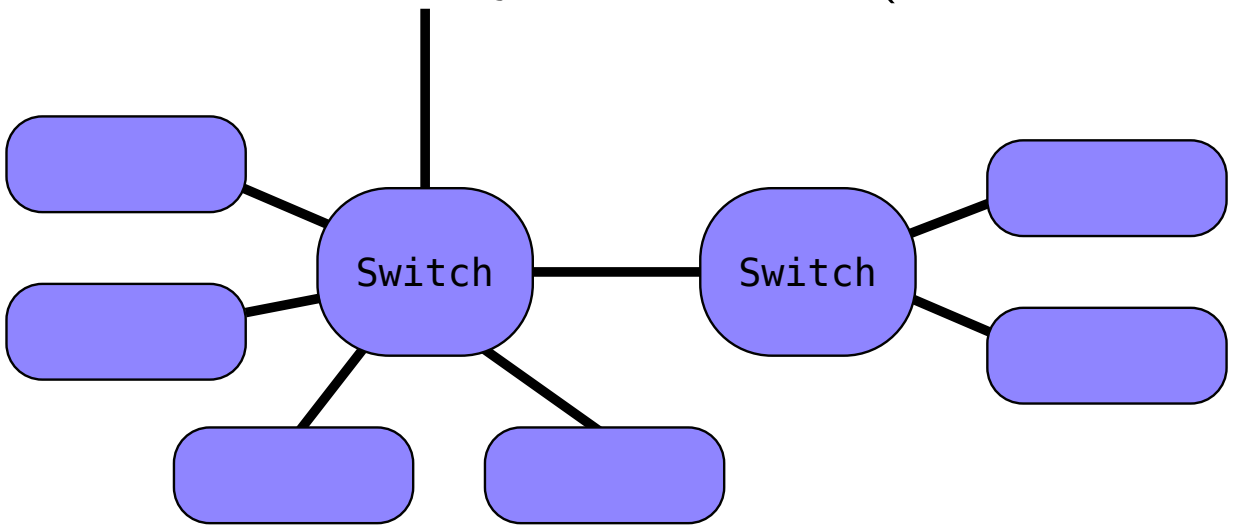
- ISA – starší typ pasivní sběrnice, šířka 8 nebo 16 bitů, přenosová rychlost maximálně 8 MB/s
- PCI – novější typ „inteligentní“ sběrnice, šířka 32 nebo 64 bitů, burst režim, přenosová rychlost až 530 MB/s
- AGP – jednoúčelová sběrnice určená pro připojení grafického karty přes severní můstek k CPU, přenosová rychlost 260 MB/s – 2 GB/s
- PCI-Express (PCIe) – nová sériová implementace sběrnice PCI

Topologie sběrnic

Sdílená sběrnice (např. PCI)



Spojení peer-to-peer s využitím switch (např. PCIe, USB)



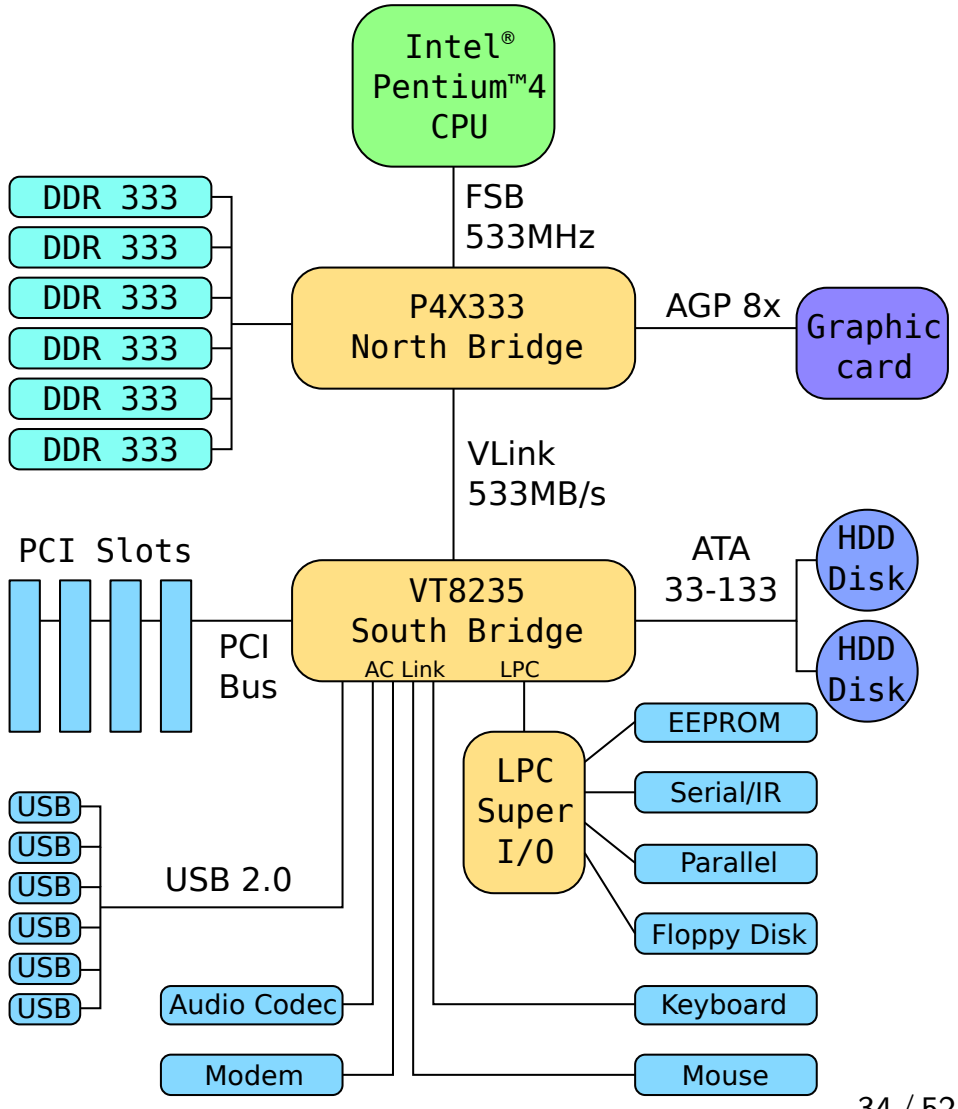
PC sběrnice

Stará architektura Pentium 4 (90. léta minulého století)

Severní můstek je připojen přímo k CPU a nejrychlejším periferiím – paměti a grafické kartě

Jižní můstek komunikuje je připojen k severnímu a má síťové karty, HDD, PCI sloty.

Nejpomalejší periferie jako Floppy Disk, nebo sériová a paralelní port (tiskárny) bývají připojeny přes další můstky.



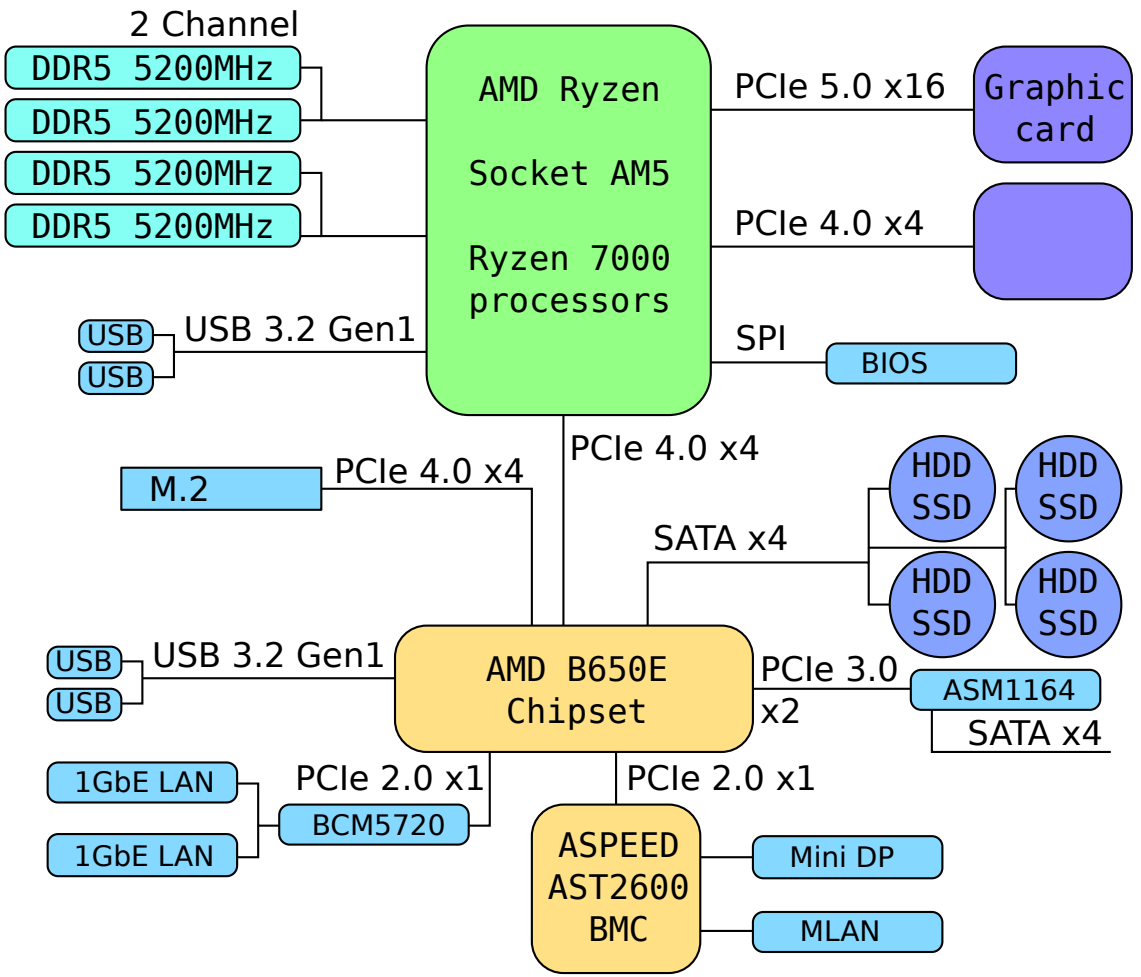
PC sběrnice

Moderní s ovladačem paměti na čipu.

Severní můstek se stal součástí procesoru.

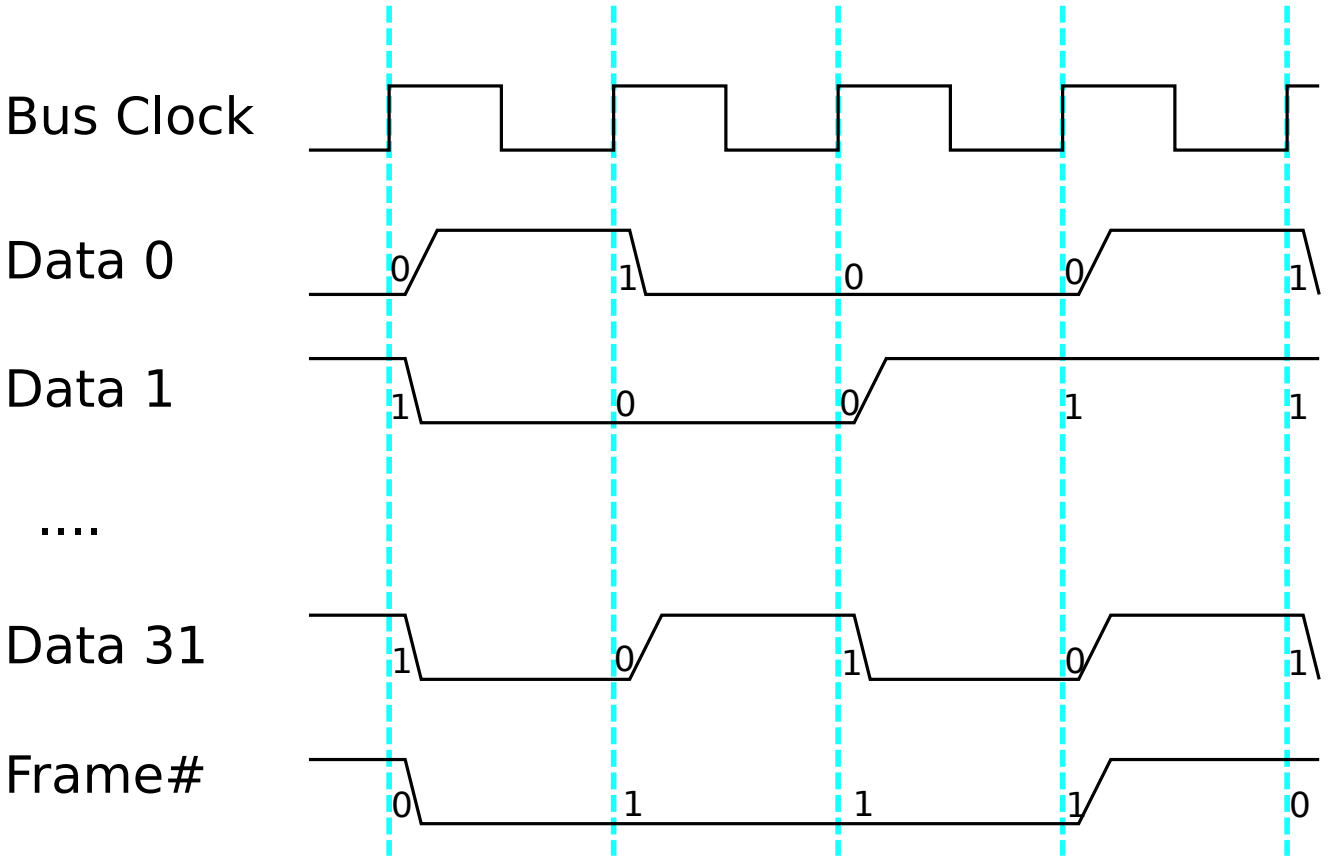
Jižní můstek komunikuje přímo s procesorem.

Většina periferií je připojena přes USB.



PCI

Sběrnice PCI je řízena hodinami.
Pro správnou činnost je nutná co nejpřesnější synchronizace podle vysílaných hodin.
Signály označené # jsou negované, protože sestupná hrana je rychlejší.



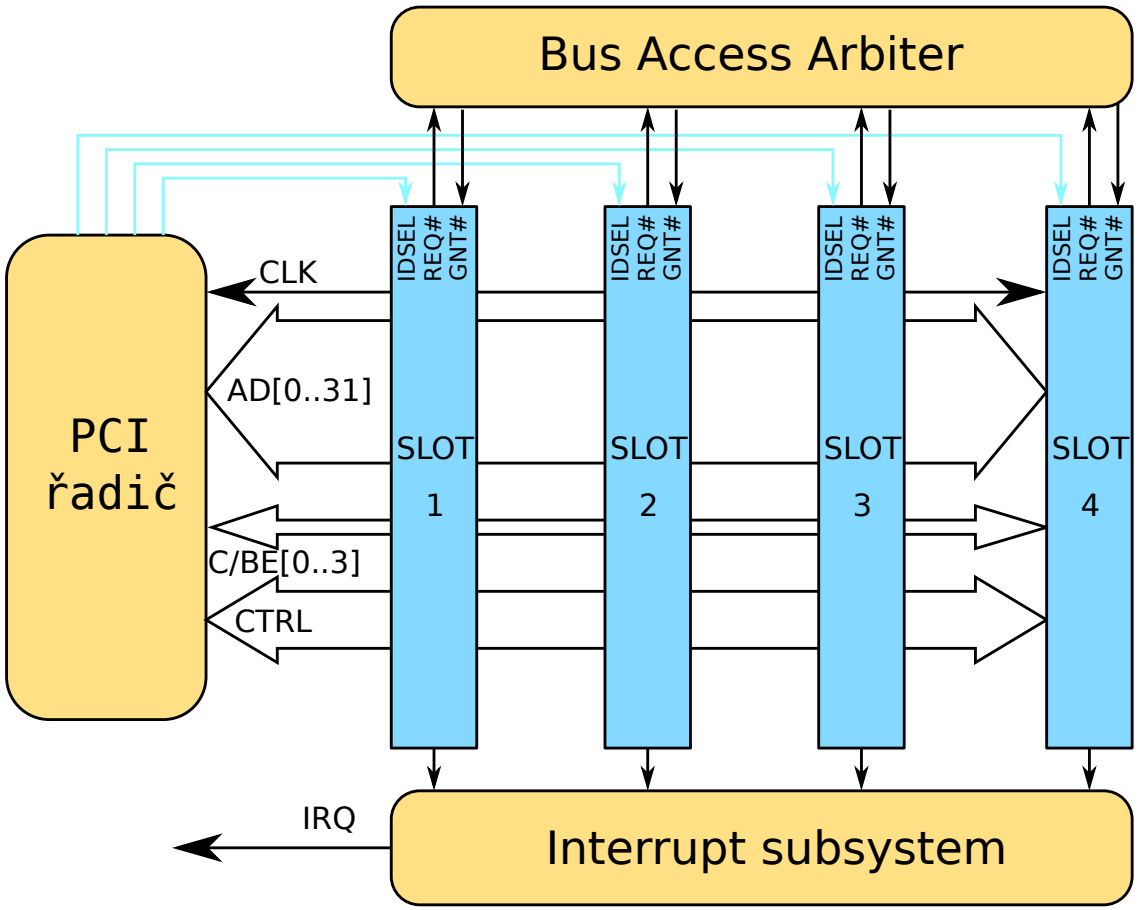
PCI architektura

Signál IDSEL je pouze pro inicializaci, zjištění, jaké zařízení je připojeno.

AD je 32 signálů používaných pro adresu a pro data

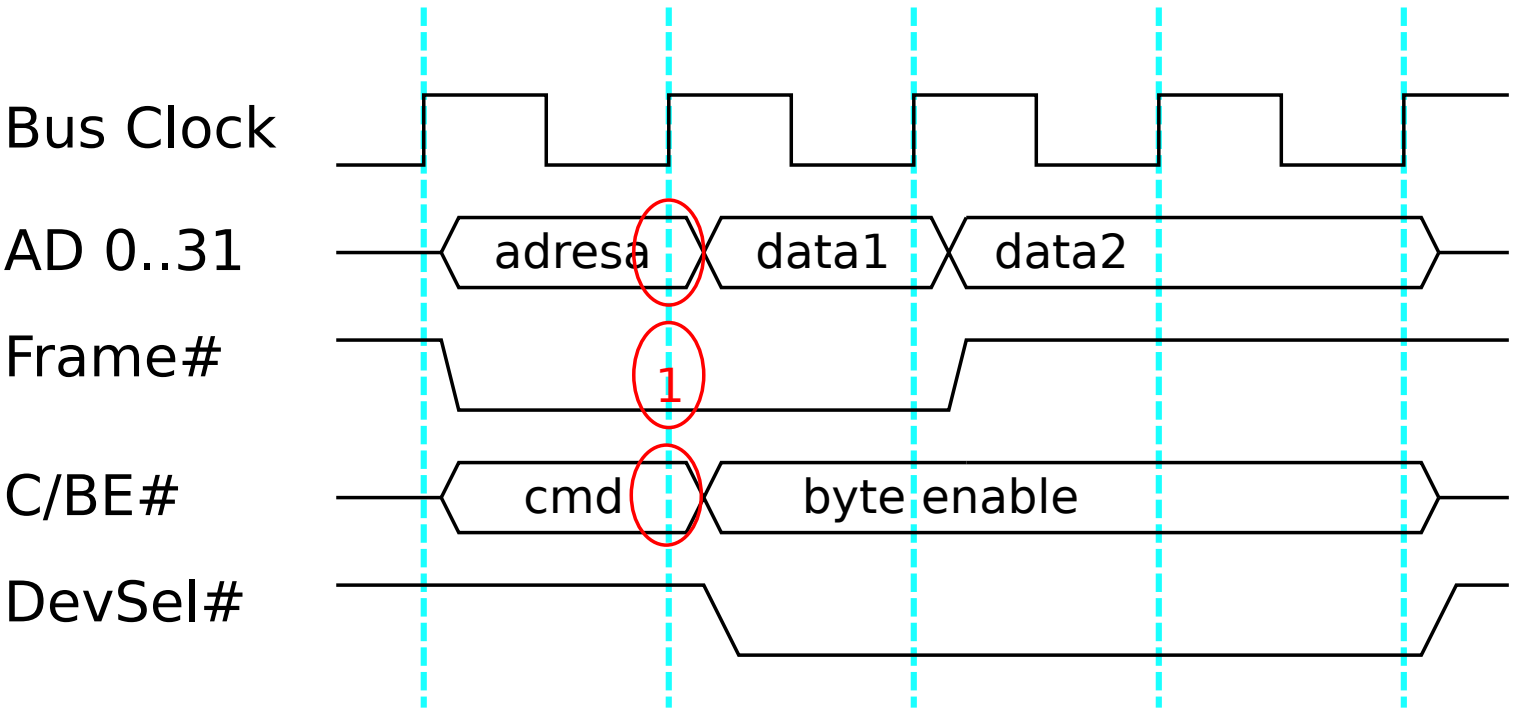
C/BE jsou 4 bity příkazu

CTRL jsou bity řídicí průběh přenosu



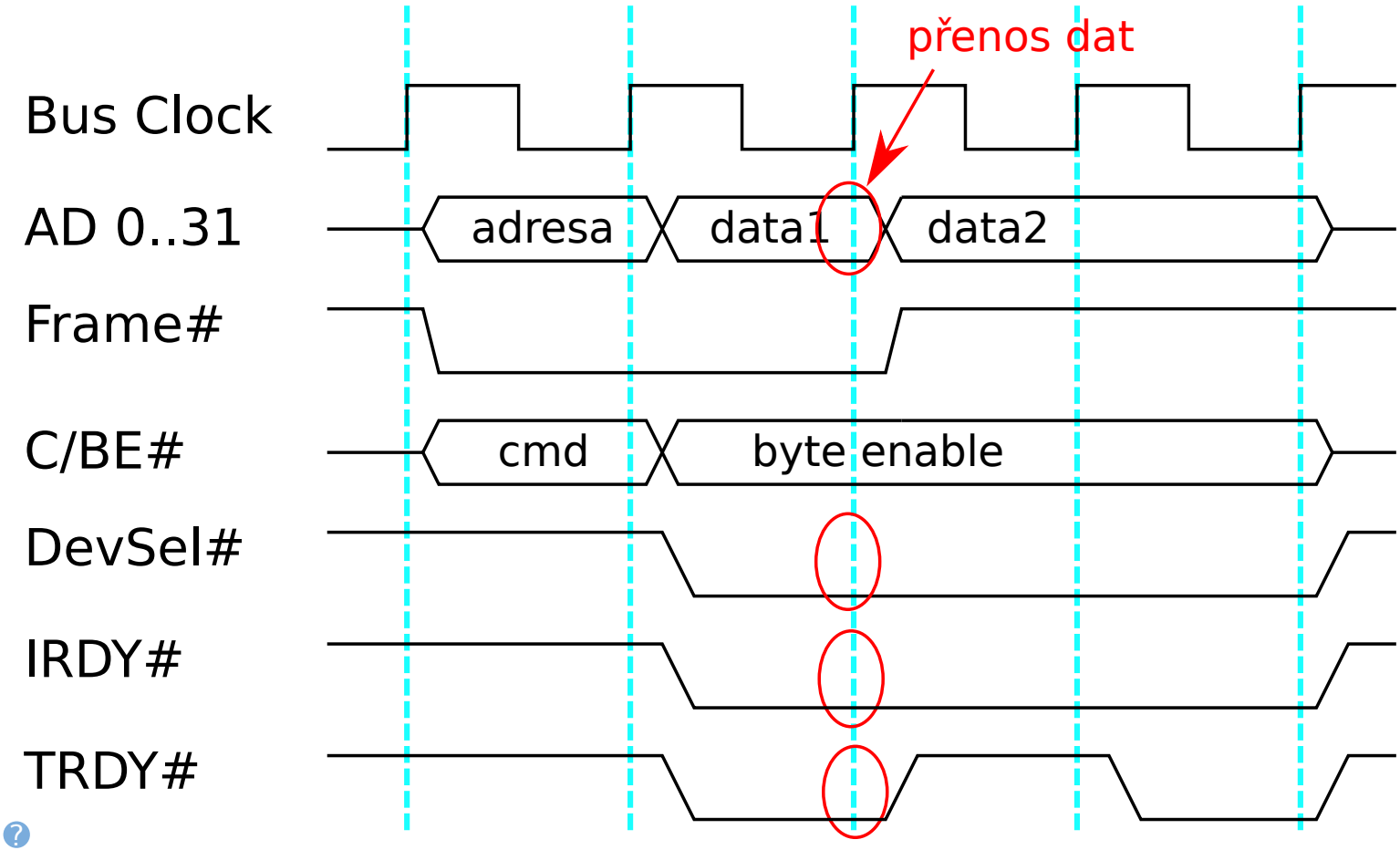
PCI přenos dat – write

- Přenos začíná iniciátor tím, že požádá arbitra o přidělení sběrnice
  - Pokud o sběrnici požádá více zařízení současně, arbitr musí jejich žádosti seřadit a povolit najednou pouze jeden přenos
- Iniciátor začne vysílání nastavením adresy cílové periférie na AD sběrnici a nastavením příznaku FRAME – rámec přenosu a bitů cmd



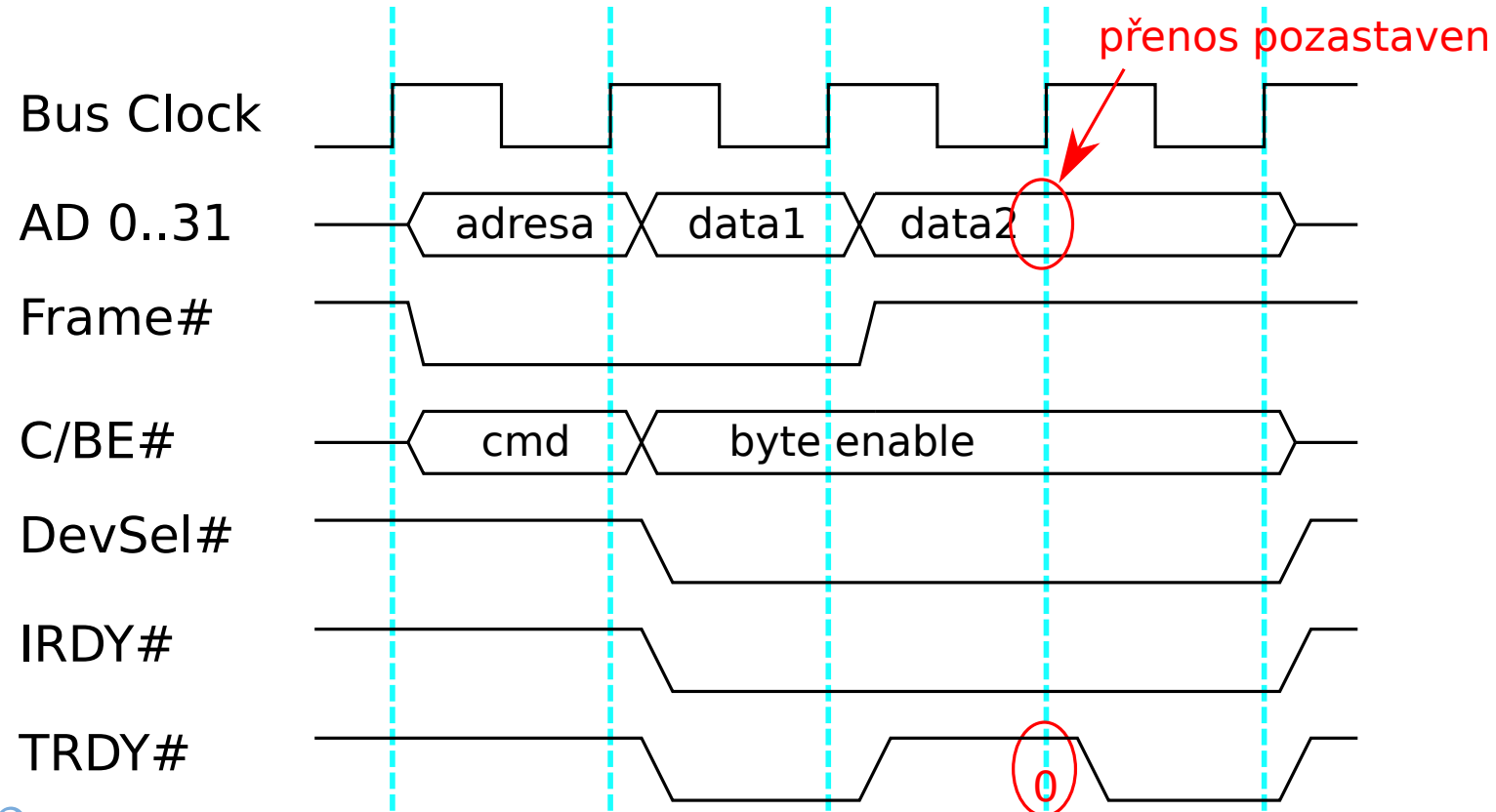
PCI přenos dat – write

- Periférie, která rozpozná svoji adresu nastaví DevSel na 1
- Pokud je cílová periferie (Target) připravena přijmout data, nastaví TRDY na 1.
- Pokud je iniciátor připraven vyslat data, nastaví IRDY na 1



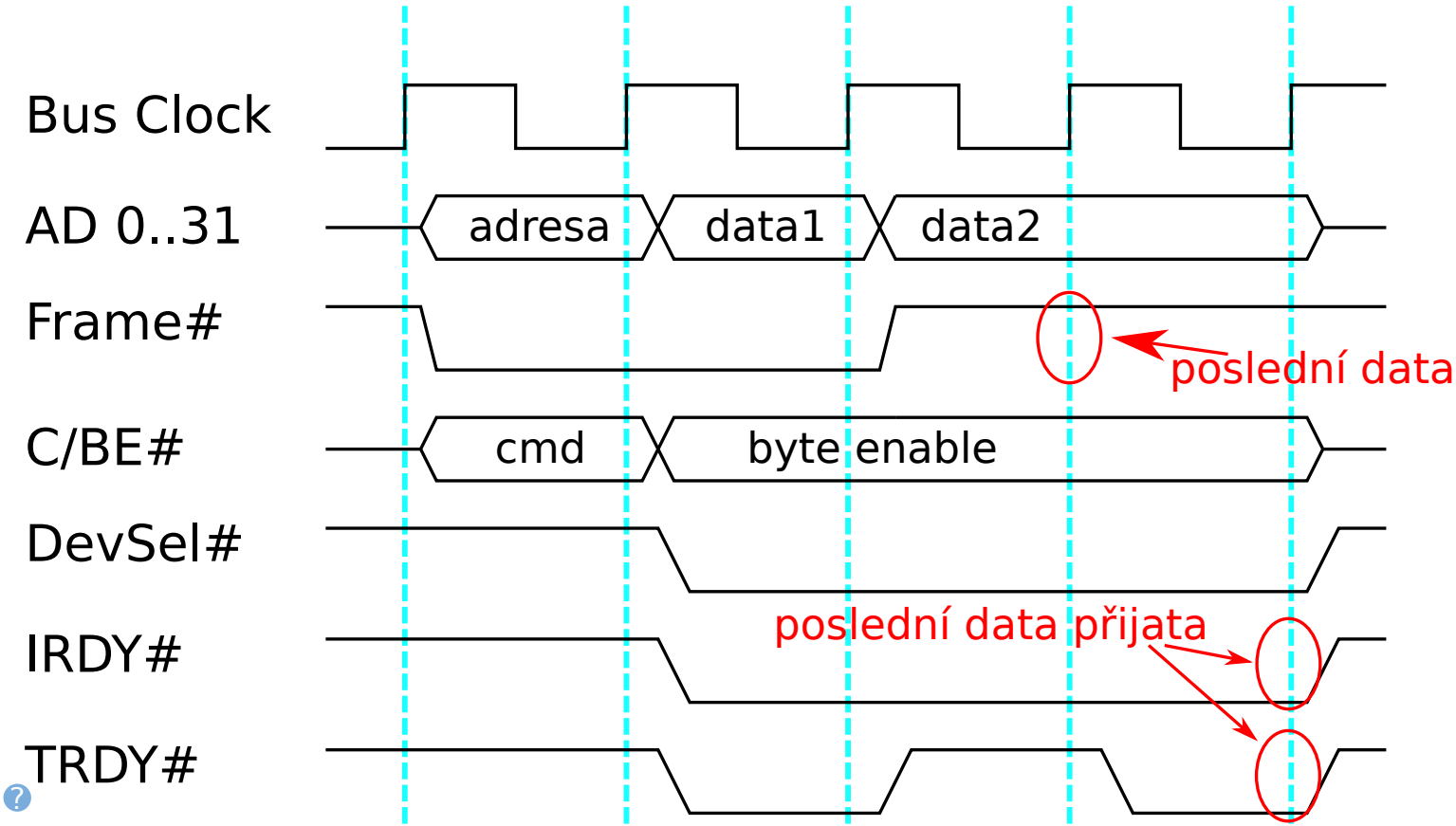
PCI přenos dat – write

- Pokud by cílová periférie nebyla připravena, tak shodí příznak TRDY na 0
- Pokud by iniciátor nebyl připraven dát data na sběrnici, shodí IRDY na 0
- Pokud je TRDY nebo IRDY na 0, pak je přenos dat pozastaven-



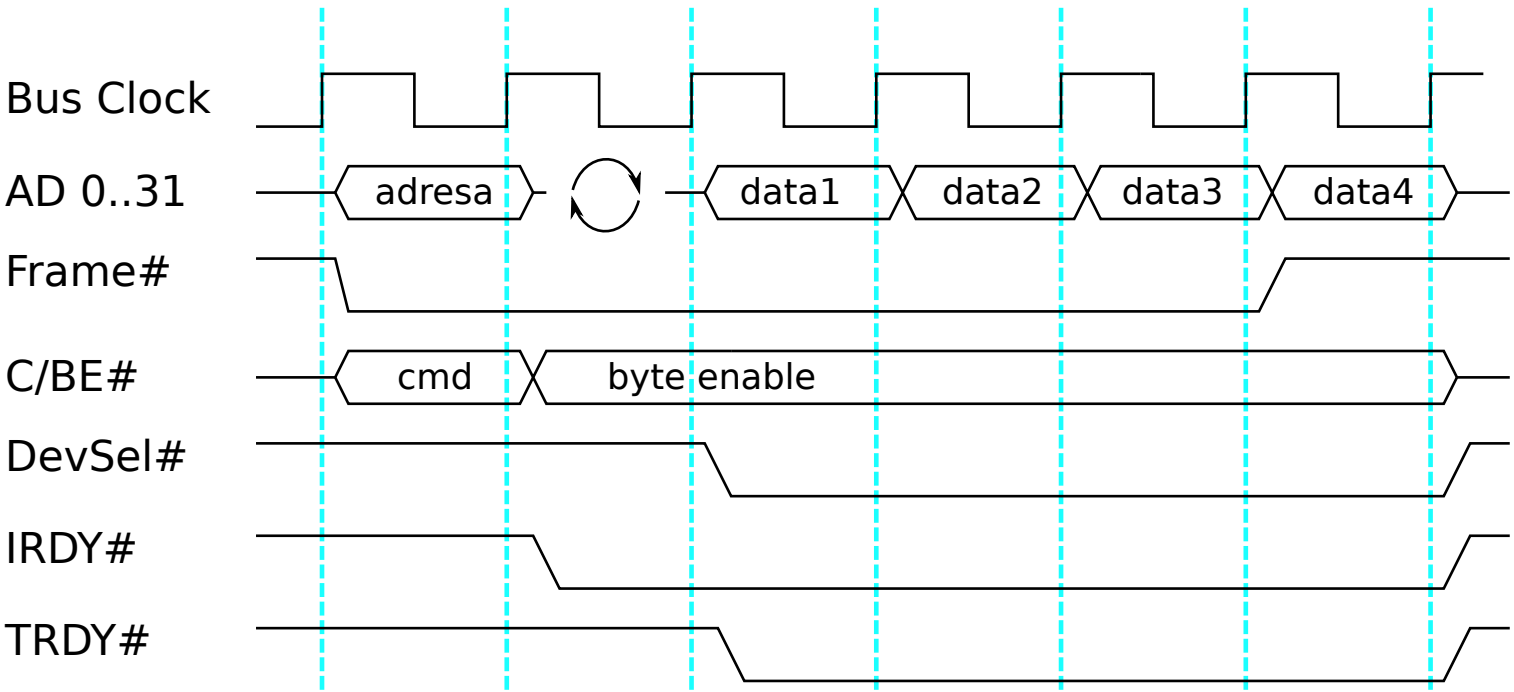
PCI přenos dat – write

- Vyslání posledních dat je indikováno shozením příznaku FRAME na 0.
- V našem případě byl přenos dat pozastaven, takže k přenosu posledních dat dojde až v dalším hodinovém cyklu.
- Po ukončení přenosu se signály IRDY, TRDY a DEVSEL vrátí na 0 a sběrnice se uvolní pro další přenos.



PCI přenos dat – read

- Iniciátor žádá o data z cílové periférie.
- Přenos dat je obdobný, ale nelze začít na následujícím hodinovém cyklu, protože se iniciátor musí od AD sběrnice odpojit a cílové zařízení se musí ke sběrnici připojit.



PCI shrnutí

Nevýhody PCI sběrnice:

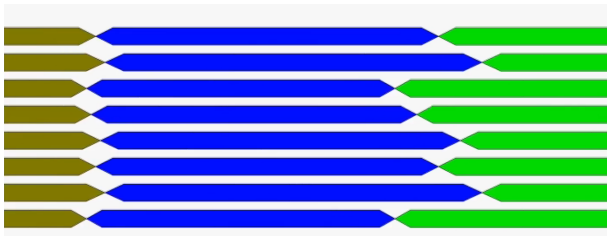
- Half–duplex nelze současně posílat data oběma směry, data se přesouvají pouze jedním směrem
- Více zařízení na sběrnici – pomalá periférie brzdí rychlé periférie, zvyšuje latency všech ostatních periférií
- PCI sběrnice umožňuje pouze hodiny s 33 MHz, nebo 66 MHz
  - To odpovídá 132MB/s nebo 264 MB/s pro 32 bitovou variantu
  - To odpovídá 264MB/s nebo 528 MB/s pro 64 bitovou variantu
- PCI eXtended (PCI-X) sběrnice umožňuje hodiny až 133 MHz později maximálně 533 MHz
  - To odpovídá přenosovým rychlostem 532MB/s až maximálně 4266 MB/s pro 64 bitovou variantu
  - PCI-X verze 2.0 s rychlostmi nad 133MHz nebyly moc rozšířené
- Konektor pro 32 bitovou verzi má 62 pinů – tj. 124 signálů, pro 64 bitovou verzi je to dokonce 188 signálů

PCI Expres – PCIe

Zásadní nevýhoda paralelního přenosu je v potřebné kvalitě vodičů:

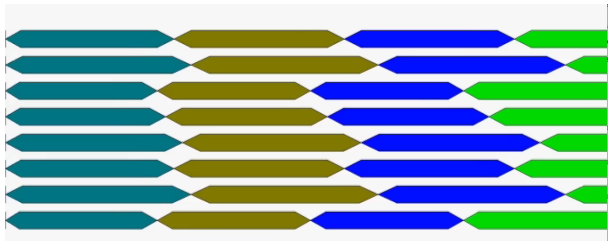
- i malé nepřesnosti v délce vodiče, kvalitě spojů vedou k rozdílným rychlostem šíření elektrického signálu
- pro malé frekvence to není problém a zvyšující se frekvence zkracují délku vysílaných dat.
- Podívejte se na animaci na https://cw.fel.cvut.cz/wiki/courses/b35apo/lectures/07/start

frekvence f



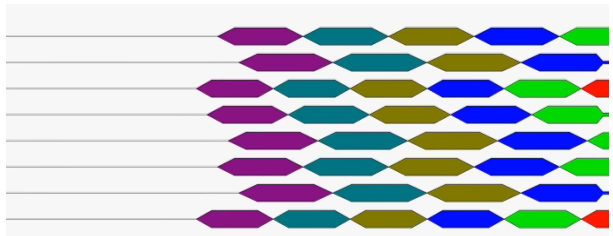
přenos bez problémů

frekvence $2 \cdot f$



přenos na hraně spolehlivosti

frekvence $4 \cdot f$



nelze v jeden okamžik detekovat výstupní data

PCIe rozdíly oproti PCI

- PCIe je tzv. peer-to-peer, tedy signály jsou pouze mezi dvěma zařízeními.
- PCIe je full-duplex tedy současně může probíhat přenos dat oběma směry.
- Pro přenos jedním směrem se používá sériový způsob s dvěma vodiči a napětím mezi těmito vodiči.
  - Tento způsob přenosu je méně náchylný na rušení než jedním vodičem vůči zemi.
- PCIe může obsahovat více linek, ale přenos mezi linkami není synchronizován na úrovni bitů.
- Konektory PCIe mají v nejjednodušší variantě pouze 18 pinů, 36 signálů z čehož 18 je uzemnění a napájení.

PCIe sériový přenos

- PCIe může využívat pro přenos různé rychlosti
- Je nutné, aby přijímací strana mohla detekovat rychlost přenosu.
- Problém je, pokud bajt obsahuje samé 0, nebo samé 1 nedochází ke změně signálu.
- Řešením je, zakódovat bajt (8bitů) do 10bitů tak, aby celkový přenesený počet 0 a 1 byl stejný.

Kvíz: Kolik existuje různých 10-bitových čísel, které mají pět bitů 0 a pět bitů 1?

A $2^5 \cdot 2^5 = 64$

B $5! \cdot 5! = 14400$

C $\binom{10}{5} = 252$

D $5! + 5! = 240$

PCIe kódování 8b/10b

- 8bitů, neboli 256 různých hodnot zakódujeme do 10bitového čísla, které má minimálně čtyři 0 a minimálně čtyři 1
  - Takových 10-bitových čísel je již $\binom{10}{5} + 2 \cdot \binom{10}{6} = 672$
  - Vybereme si takové, kde se hodně mění 1.
- V kódovacích tabulkách je volnost vybrat si zda kód bude mít více 1, nebo více 0.
- Ve skutečnosti se kód skládá ze dvou podkódů 5b/6b a 3b/4b.

tabulka kódu 3b/4b

Input		RD = −1	RD = +1
Code	HGF	f g h j	
D.x.0	000	1011	0100
D.x.1	001	1001	
D.x.2	010	0101	
D.x.A3 D.x.B3	011	1100 0011	
D.x.4	100	1101	0010
D.x.5	101	1010	
D.x.6	110	0110	
D.x.P7 D.x.A7	111	1110 0111	0001 1000

PCIe ver. 1.x a 2.x

Ver 1.x

- Frekvence přenosu je 2.5 GT/s (počet transferů za s)
- Pro jeden bajt 8bitů je potřeba 10 transferů
- Maximální přenosová kapacita je tedy 250 MB/s $= (2500 \cdot \frac{8}{10})$ Mb/s $= (2500 \cdot \frac{1}{10})$ MB/s
- PCIe umožňuje pro jednu periferie mít 16 nezávislých linek, které se podělí o přenášená data – nezávisle paralelně
  - Maximální přenosová kapacita je $(16 \cdot 250)$ MB/s $= 4$ GB/s

Ver 2.x

- Frekvence přenosu je 2.5 GT/s (počet transferů za s)
- Pro jeden bajt 8bitů je potřeba 10 transferů
- Maximální přenosová kapacita jedné linky (x1) je 500 MB/s
- Maximální přenosová kapacita pro 16 linek (x16) je $(16 \cdot 500)$ MB/s $= 8$ GB/s

PCIe ver. 3.x, 4.x a 5.x

Kódování 8b/10b je zbytečně neefektivní, zvolilo se kódování 128b/130b s obdobnými parametry.

- Ver 3.x
  - Frekvence přenosu je 8 GT/s (počet transferů za s)
  - Maximální přenosová kapacita je tedy skoro 985 MB/s $= (8000 \cdot \frac{128}{130})$ Mb/s $= (8000 \cdot \frac{16}{130})$ MB/s
  - Maximální přenosová kapacita pro x16 je $(16 \cdot 985)$ MB/s $= 15.75$ GB/s
- Ver 4.x
  - Frekvence přenosu je 16 GT/s (počet transferů za s)
  - Maximální přenosová kapacita je tedy skoro 1.97 GB/s $= (16000 \cdot \frac{16}{130})$ MB/s
  - Maximální přenosová kapacita pro x16 je $(16 \cdot 1.97)$ GB/s $= 31.5$ GB/s
- Ver 5.x
  - Frekvence přenosu je 32 GT/s (počet transferů za s)
  - Maximální přenosová kapacita je tedy skoro 3.94 GB/s $= (32000 \cdot \frac{16}{130})$ MB/s
  - Maximální přenosová kapacita pro x16 je $(16 \cdot 3.94)$ GB/s $= 63$ GB/s

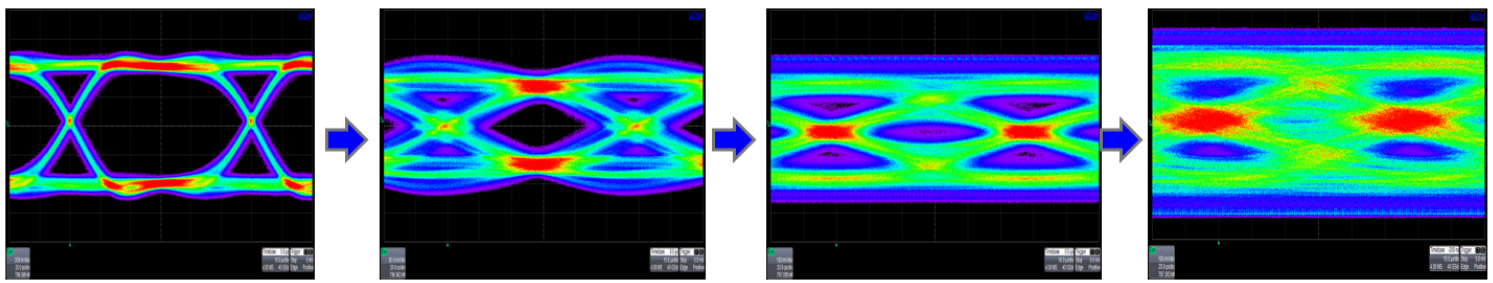
PCIe topologie

Komunikace po PCIe sběrnici se podobá komunikaci po síti (ethernet).

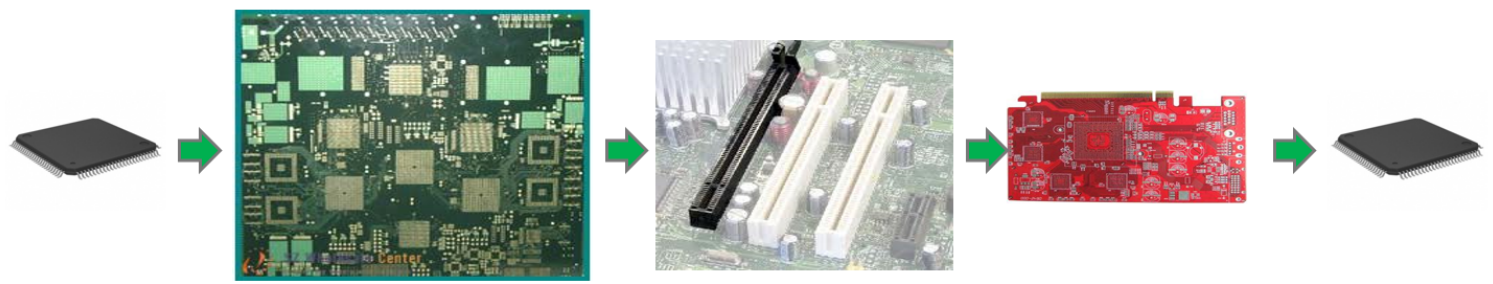
- Komunikace probíhá po paketech
  - POZOR – režie paketů není započítána do maximální přenosové kapacity.
  - Každý paket má synchronizační hlavičku, adresu, data, crc – obdobně jako v protokolu ethernet.
- Využití přepínačů (switch) je podobné jako v síti
  - Přepínače umožňují komunikovat pouze mezi dvěma zařízeními
  - Přepínače umí prioritizovat pakety – výhodné pro snížení latence (použití paketů naproti tomu latenci zvyšuje)
  - Pomocí přepínačů lze zajistit automatickou detekci a konfiguraci připojených zařízení na obdobném principu, jako to dělal PCI sběrnice

Realita signálů sériové sběrnice

Vysokorychlostní komunikace přináší mnoho různých problémů.
Vzhled signálů v závislosti na vzdálenosti



Signal degrades over long transmission path and connectors



Hard disky

- Obdobný vývoj, který je u změny PCI na PCIe lze pozorovat i u disků.
- PATA neboli Parallel ATA je paralelní připojení disku od roku 1984 pro první IBM PC/AT
  - Název ATA znamená vlastně AT Attachment, AT je zkratka Advance Technology.
  - Také označován jako IDE, později Extended IDE (EIDE) Utra ATA (UATA)
  - PATA je 16-bitový paralelní přenos dat mezi CPU a diskem
  - Ve své nejrychlejší variantě uměl přenášet až 133 MB/s
- SATA je sériová verze komunikace s diskem.
  - V minimální variantě potřebuje pouze 7 vodičů, A+,A-, B+, B- a 3x zemění.
    - SATA 1.0: 150 MB/s (PATA:130MB/s)
    - SATA 2.0: 300 MB/s
    - SATA 3.0: 600 MB/s
    - SATA 3.2: about 2 GB/s