# Návrh číslicových systémů (INC)

Jiří Matoušek, Otto Fučík, Tomáš Martínek

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

## Paměti a sběrnice

# Obsah

- Paměti
  - Konstrukce
  - Příklady užití
- Sběrnice
- Adresový dekodér

# Paměti

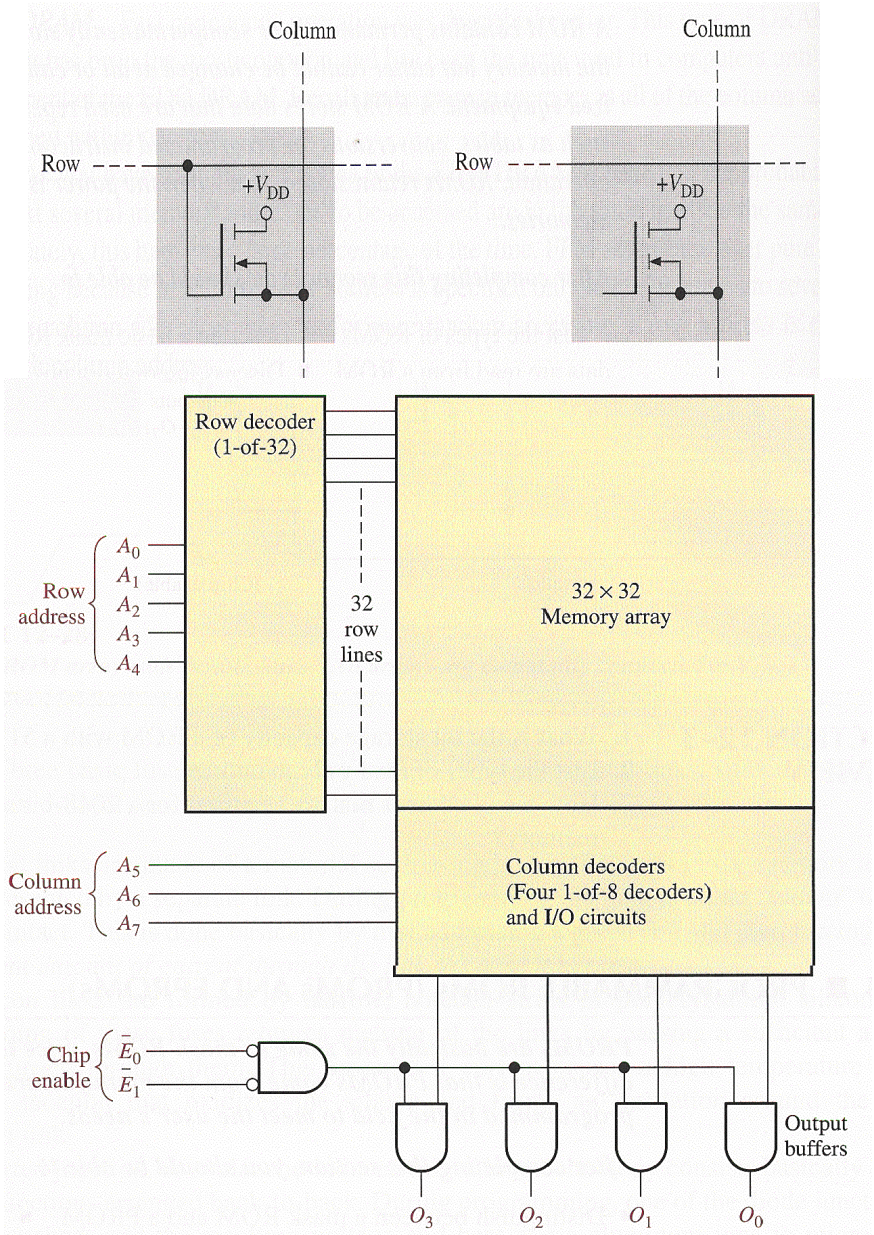
- Pouze pro čtení (Read-Only Memory - ROM)
  - Programované maskou ve výrobě (Mask-Programmable ROM)
  - Jednorázově programovatelné uživatelem (Programmable ROM)
- Pro omezený počet zápisů a neomezené čtení
  - Elektricky mazatelné a programovatelné
  - Existuje řada verzí - EPROM, EEPROM, FLASH (NAND, NOR)…
- Pro neomezený zápis a čtení (Read-Write Memory RWM)
  - Se sekvenčním přístupem
    - Zásobník (LIFO), Fronta (FIFO)
  - S náhodným přístupem (Random Access Memory – RAM)
    - Dynamická RAM (Dynamic RAM -DRAM) s asynchronním či synchronním (SDR, DDR…) přístupem
    - Statická RAM (Static RAM - SRAM) s asynchronním či synchronním přístupem
  - Adresovatelná obsahem (Content-Addressable Memory - CAM)

# Obsah

- Paměti
  - Konstrukce
  - Příklady užití
- Sběrnice
- Adresový dekodér

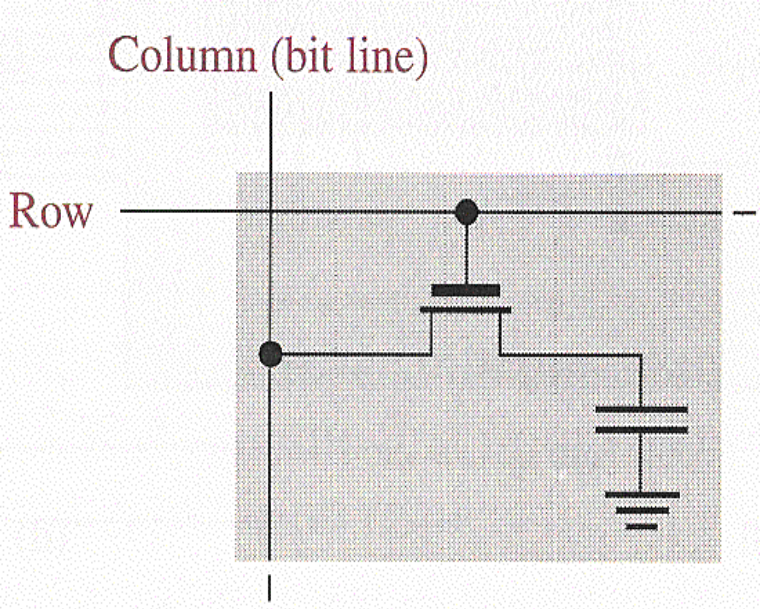
# ROM (Read-Only Memory)

- Příklad implementace paměťových buněk
  - Tranzistorem MOS se zapojenou (log.1)/nezapojenou (log.0) řídicí elektrodou
  - Spoj k elektrodě se např. „přepálí" při programování ve výrobě
- Příklad organizace paměti ROM
  - 32x32 buněk
  - Dekodér adresy sloupce (Column)
  - Dekodér adresy řádku (Row)
  - Chip Select – povolení funkce



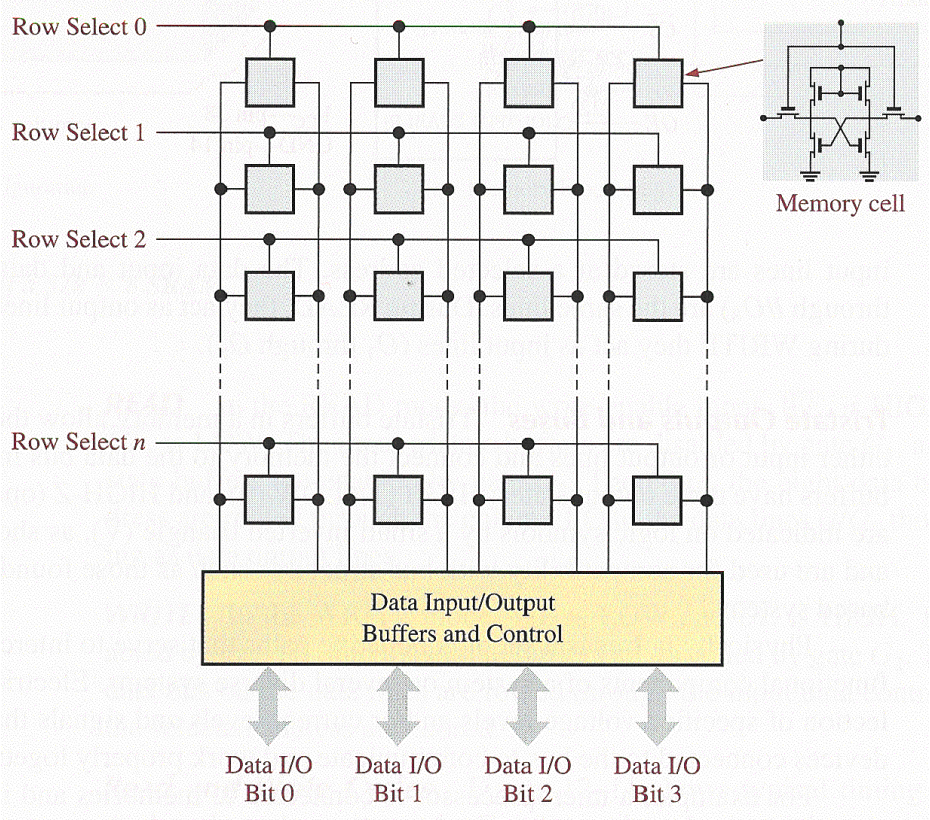
# DRAM (Dynamic RAM)

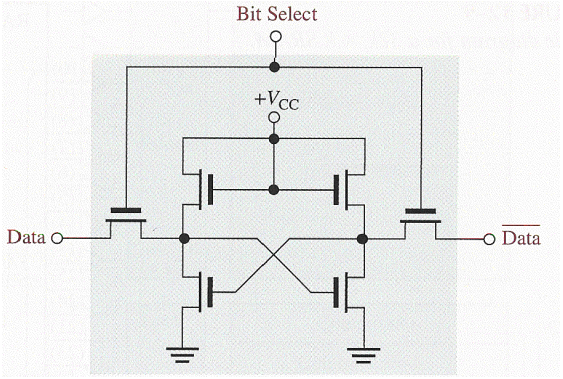
- Dynamická paměť pro zápis i čtení s náhodným přístupem
- Implementace paměťových buněk
  - Hodnota se uchovává na parazitní kapacitě jako náboj
  - Kondenzátor se časem vybíjí vlivem parazitních svodů
  - Hodnota náboje se musí tzv. osvěžovat (anglicky „refreshing")
- Velké kapacity, jednoduché, levné, pomalé



# SRAM (Static RAM)

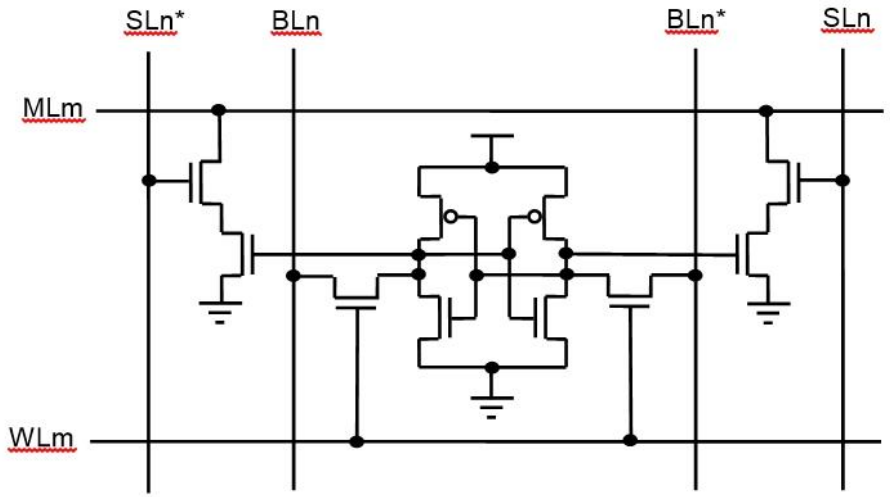
- Statická paměť pro zápis i čtení s náhodným přístupem
- Implementace paměťových buněk
  - V podstatě se jedná o hladinový RS klopný obvod sestavený ze dvou invertorů, který se překlápí na základě komplementárních signálů DATA při aktivním výběrovém signálu BIT SELECT
- Rychlé, složité, drahé, malé kapacity, velký příkon



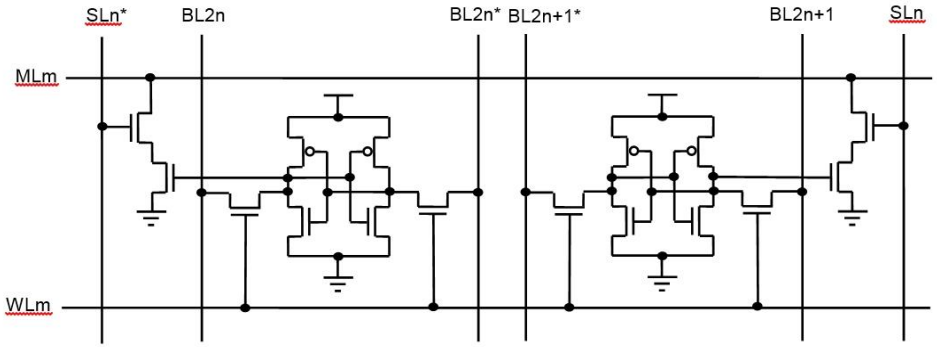



# CAM (Content-Addressable Memory)

- Paměť adresovaná obsahem (výsledkem čtení je adresa)
  - TCAM (Ternary CAM) umí pracovat s hodnotami 0, 1, X
- Implementace paměťových buněk
  - CAM – SRAM paměťová buňka doplněná o dvojici "komparátorů"
  - TCAM – dvojice SRAM paměťových buněk doplněná o dvojici "komparátorů"
- Velmi rychlé, složité a drahé, velmi malé kapacity, velmi vysoký příkon




CAM

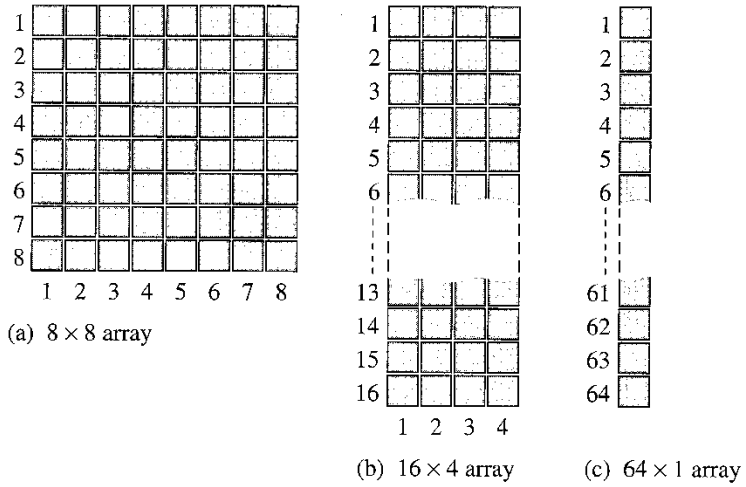


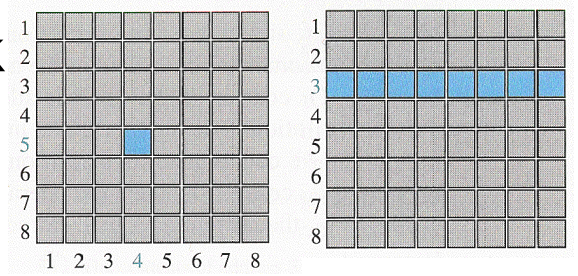

TCAM

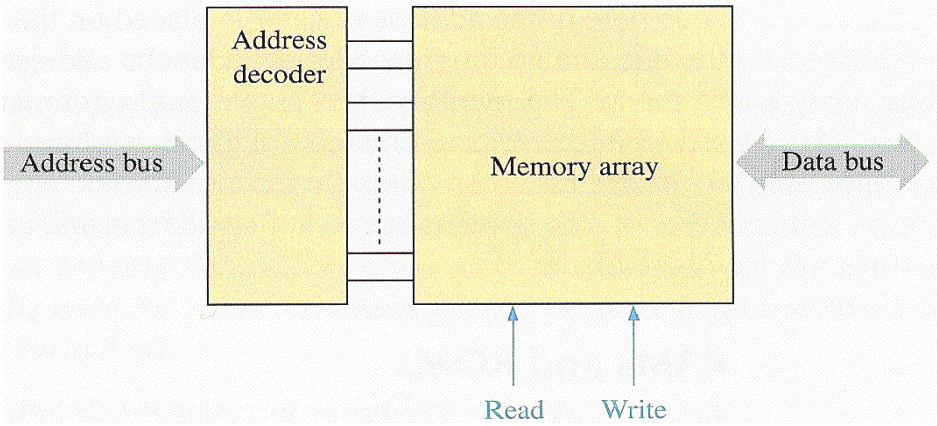
zdroj: https://en.wikipedia.org/wiki/Content-addressable_memory

# Organizace pamětí

- Paměť o kapacitě 64 bitů
  - Matice (pole) 8x8 (a)
  - Matice (pole) 16x4 (b)
  - Matice (pole) 64x1 (c)
- Adresa
  - Umístění jednotky dat v paměťové matici
- Příklad adresování paměťových buněk
  - Adresa bitu - řádek 5, sloupec 4
  - Adresa řádku 5
- Organizace paměti
  - Adresová a datová sběrnice
  - Řídicí signály pro zápis a čtení
  - Paměťová matice





# Parametry pamětí

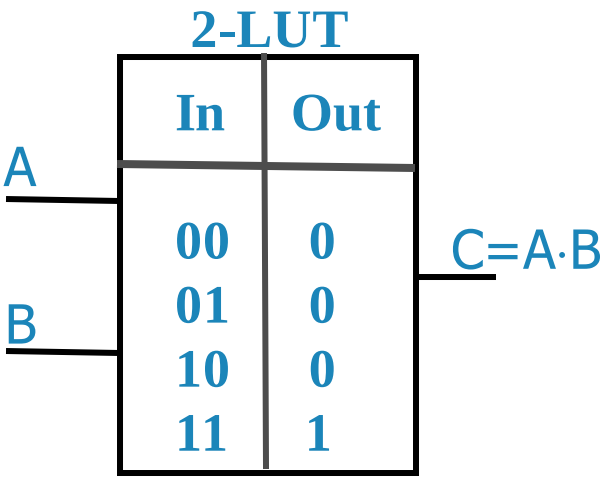
- Vlastnosti
  - Plocha - hustota integrace (počet bitů na $\mu m^2$)
  - Kapacita (počet bitů, organizace)
  - Doba přístupu (access time) - doba četní a zápisu
  - Doba cyklu – (cycle time) – doba mezi jednotlivými přístupu do paměti
  - Propustnost – počet čtení/zápisů za sekundu (závisí na době cyklu)
  - Příkon atd.

# Obsah

- Paměti
  - Konstrukce
  - Příklady užití
- Sběrnice
- Adresový dekodér

# Generátory logických funkcí

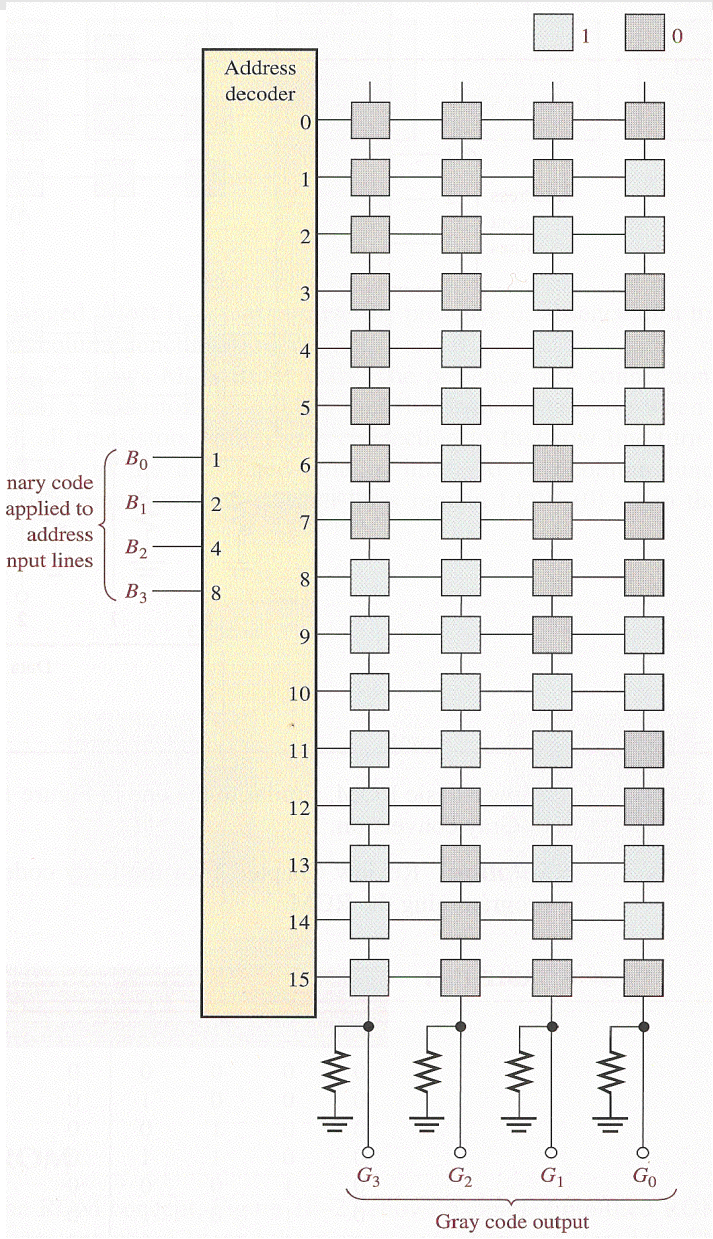
- Paměti lze použít jako generátory logických funkcí
  - Tzv. vyhledávací tabulka (Look-Up Table - LUT) implementuje libovolnou log. funkci N proměnných, kde N je počet adresovacích vstupů paměti
- Příklad
  - Funkce AND



# Převodníky

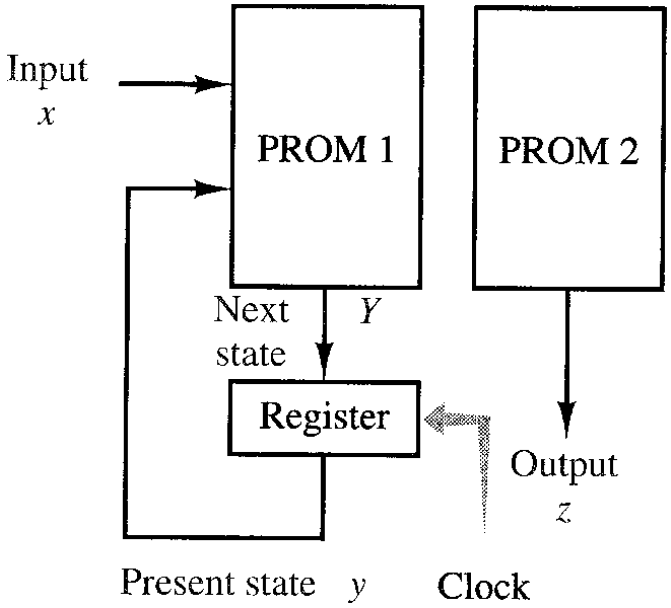
- Příklad použití ROM paměti
  - Převodník binárního kódu na Grayův kód
  - Pravdivostní tabulka je implementována v paměťové matici
  - Adresa – vstupní proměnné
  - Data – výstupy

| Binary | | | | Gray | | | |
|---|---|---|---|---|---|---|---|
| $B_3$ | $B_2$ | $B_1$ | $B_0$ | $G_3$ | $G_2$ | $G_1$ | $G_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |



# Řadiče

- Implementace kombinační logické sítě
  - PROM1 – přechodová funkce
  - PROM2 – výstupní funkce
    - Obsah obou pamětí lze programovat, a tím též měnit činnost automatu
    - Zapojení obvodu zůstává stejné (výhoda)
    - Obě paměti lze sloučit do jedné
  - Registr – paměť stavu

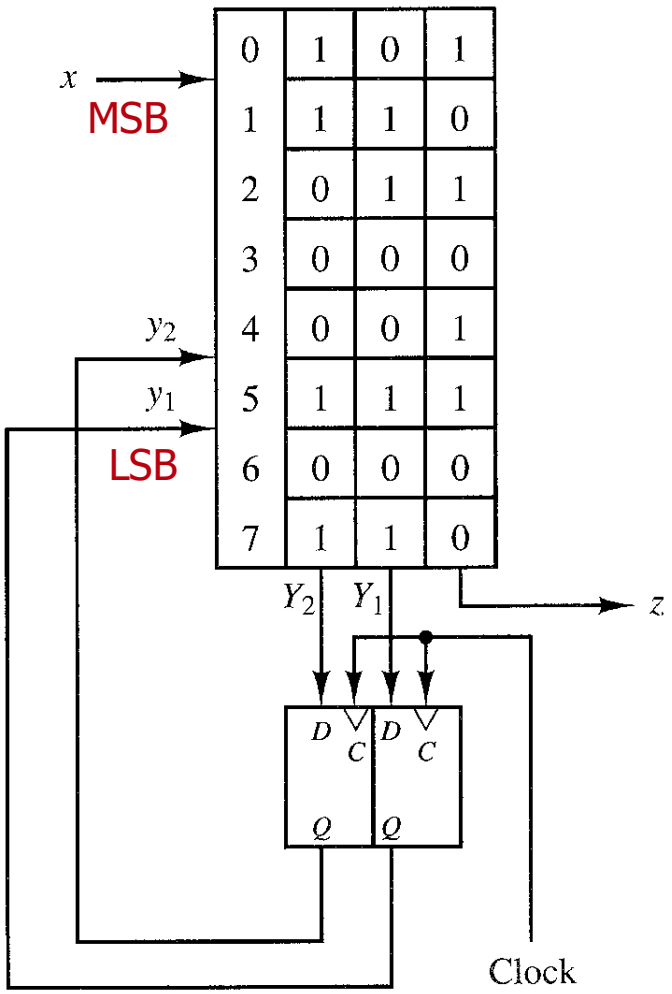



- Popis
  - Vstupy X jsou přivedeny na část adresových vodičů paměti PROM1
  - Na datových vodičích paměti PROM1, které jsou přivedeny na registr stavu, se čte kód následujícího stavu (next state)
  - Výstup registru stavu (present state - současný stav) je přiveden na zbytek adresových vodičů paměti PROM1
  - Adresové vodiče PROM2 jsou přivedeny
    - Pro Mooreův automat na výstup registru
    - Pro Mealyho automat na výstup registru a vstupy X
  - Na datových výstupech PROM2 se čtou výstupy Z

- Popis
  - Tabulka přechodů

| $y_2y_1$ | $x$ = 0 | $x$ = 1 |
|---|---|---|
| 00 | 10/1 | 00/1 |
| 01 | 11/0 | 11/1 |
| 10 | 01/1 | 00/0 |
| 11 | 00/0 | 11/0 |

$Y_2Y_1/z$

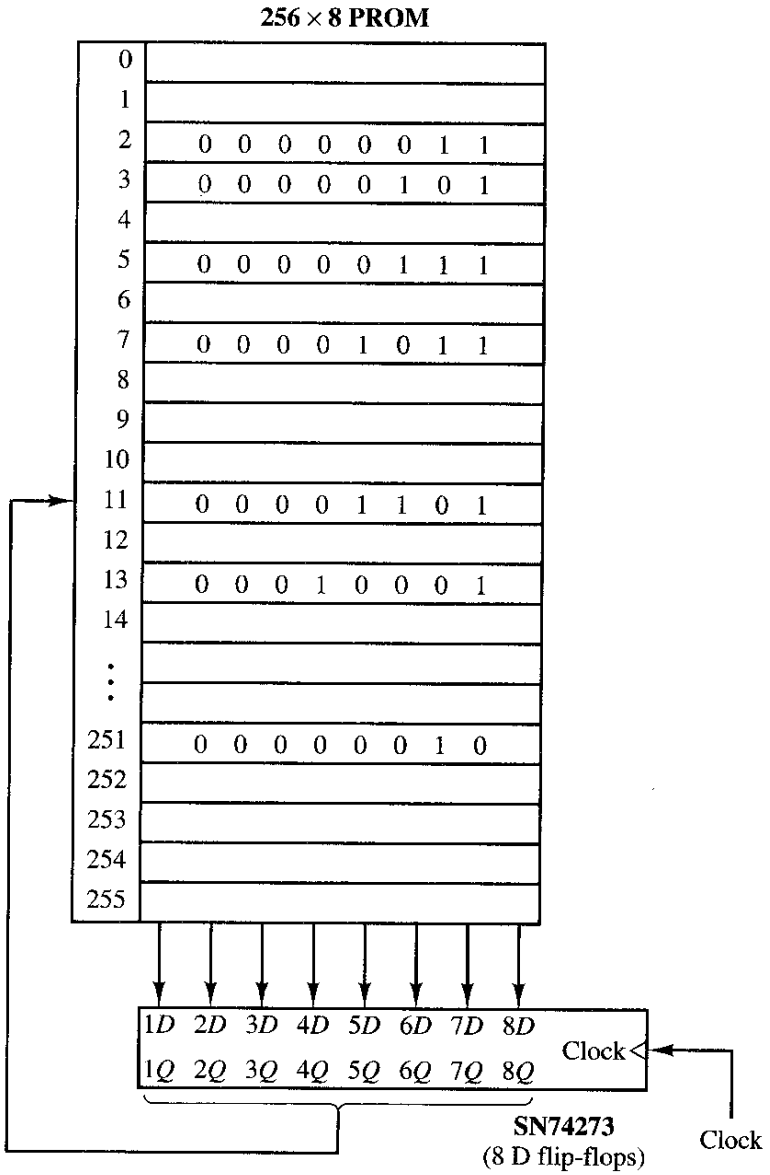



  - Přechodová a výstupní funkce = obsah paměti PROM

| $x$ | $y_2$ | $y_1$ | $Y_2$ | $Y_1$ | $z$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |

# Generátory posloupností

- Generátor posloupnosti
  - Na adresách odpovídajícím jednotlivým prvočíslům jsou uloženy adresy následujících prvočísel
  - Automat startuje od adresy 2
  - S každým taktem hodin se na výstupu registru generuje rostoucí posloupnost prvočísel 2,3,5,7,...251,2,... (8 bitů)
  - Pouhou změnou obsahu paměti lze vytvořit (přeprogramovat) automat pro generování libovolné posloupnosti s max. počtem $2^n$ prvků, kde n je počet bitů paměti a registru



# Obsah

- Paměti
  - Konstrukce
  - Příklady užití
- Sběrnice
- Adresový dekodér

# Základní určení (synchronních) sběrnic

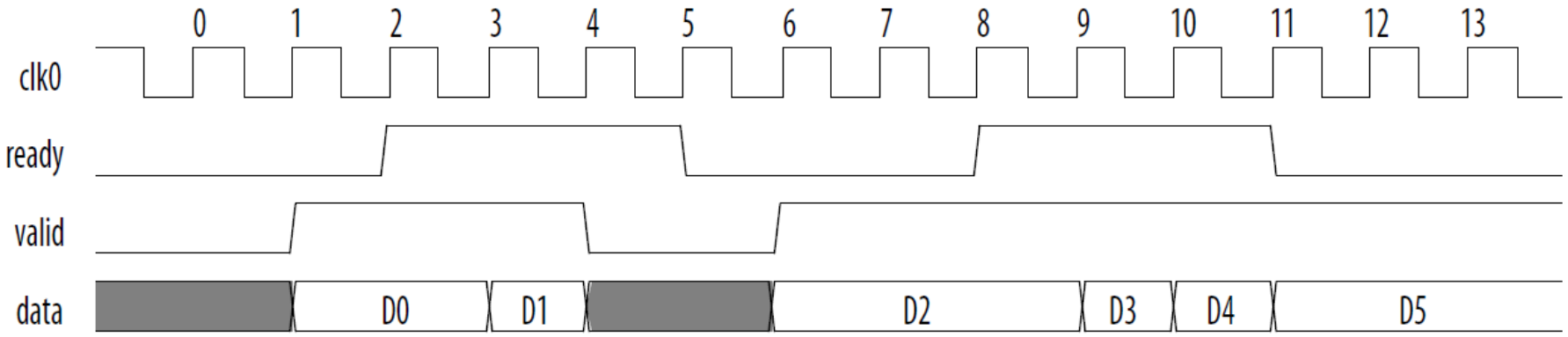
- Pro proudové zpracování
  - Hodinový a resetovací signál
  - Vodiče pro přenos dat/metadat
  - Řídicí vodiče (handshake protokol)
- Pro paměťové operace
  - Hodinový a resetovací signál
  - Vodiče pro zapisovaná a/nebo vyčtená data a metadata
  - Vodiče pro zápisovou a/nebo čtecí adresu
  - Řídicí vodiče (volba zápisové/čtecí operace)

# Standardizované sběrnice pro FPGA

| | AMD (Xilinx) | Intel |
|---|---|---|
| **Proudové zpracování** | AXI4-Stream | Avalon Streaming |
| **Paměťové operace** | AXI4<br>AXI4-Lite | Avalon Memory-Mapped |

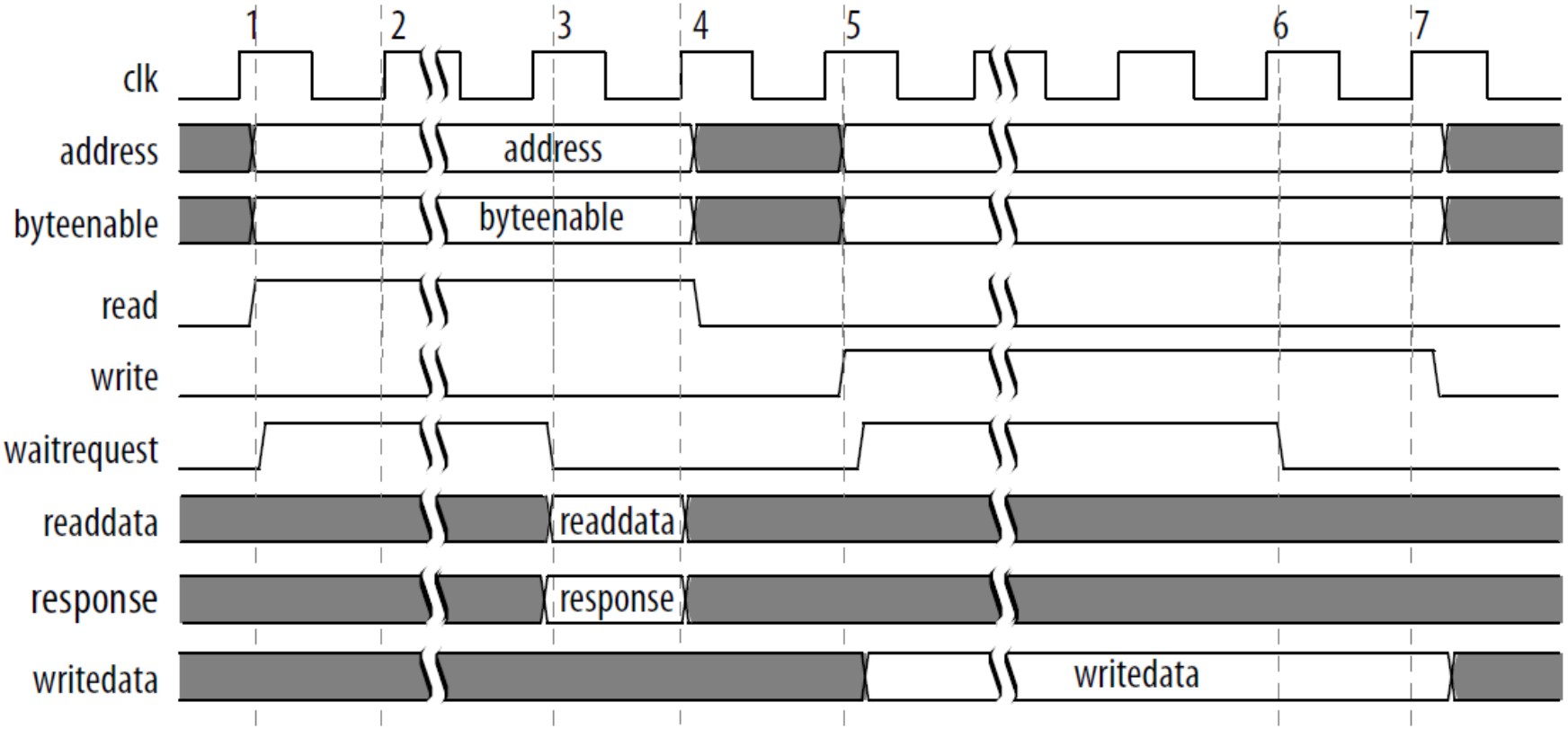
- Sběrnice AXI vycházejí z rodiny sběrnic AMBA od ARM
- AXI Reference Guide od AMD (Xilinx):
  - https://www.xilinx.com/support/documentation/axi_ref_guide/latest/ug761_axi_reference_guide.pdf
- Avalon Interface Specifications od Intel:
  - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

# Avalon Streaming




zdroj: Avalon Interface Specifications

# Avalon Memory-Mapped



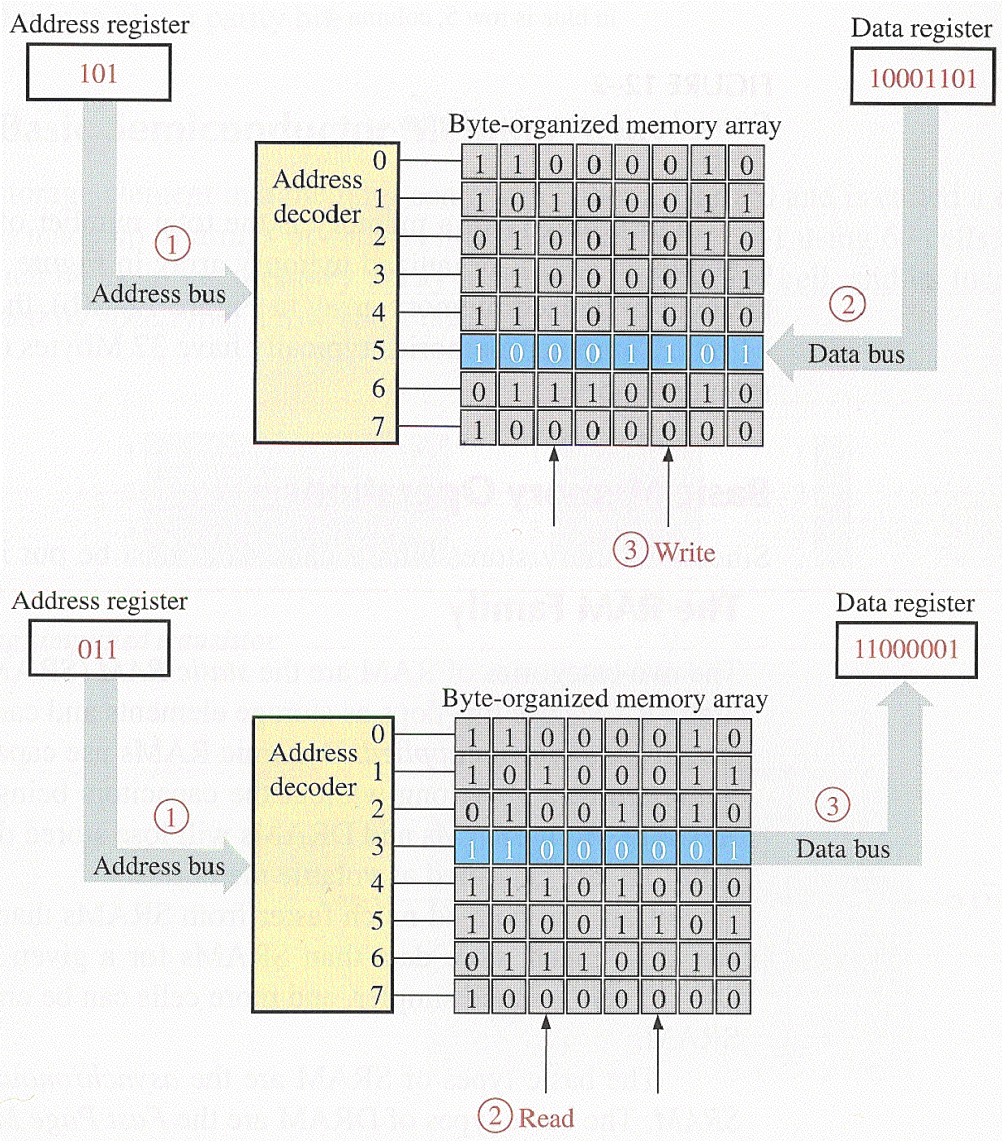

zdroj: Avalon Interface Specifications

# Obsah

- Paměti
  - Konstrukce
  - Příklady užití
- Sběrnice
- Adresový dekodér

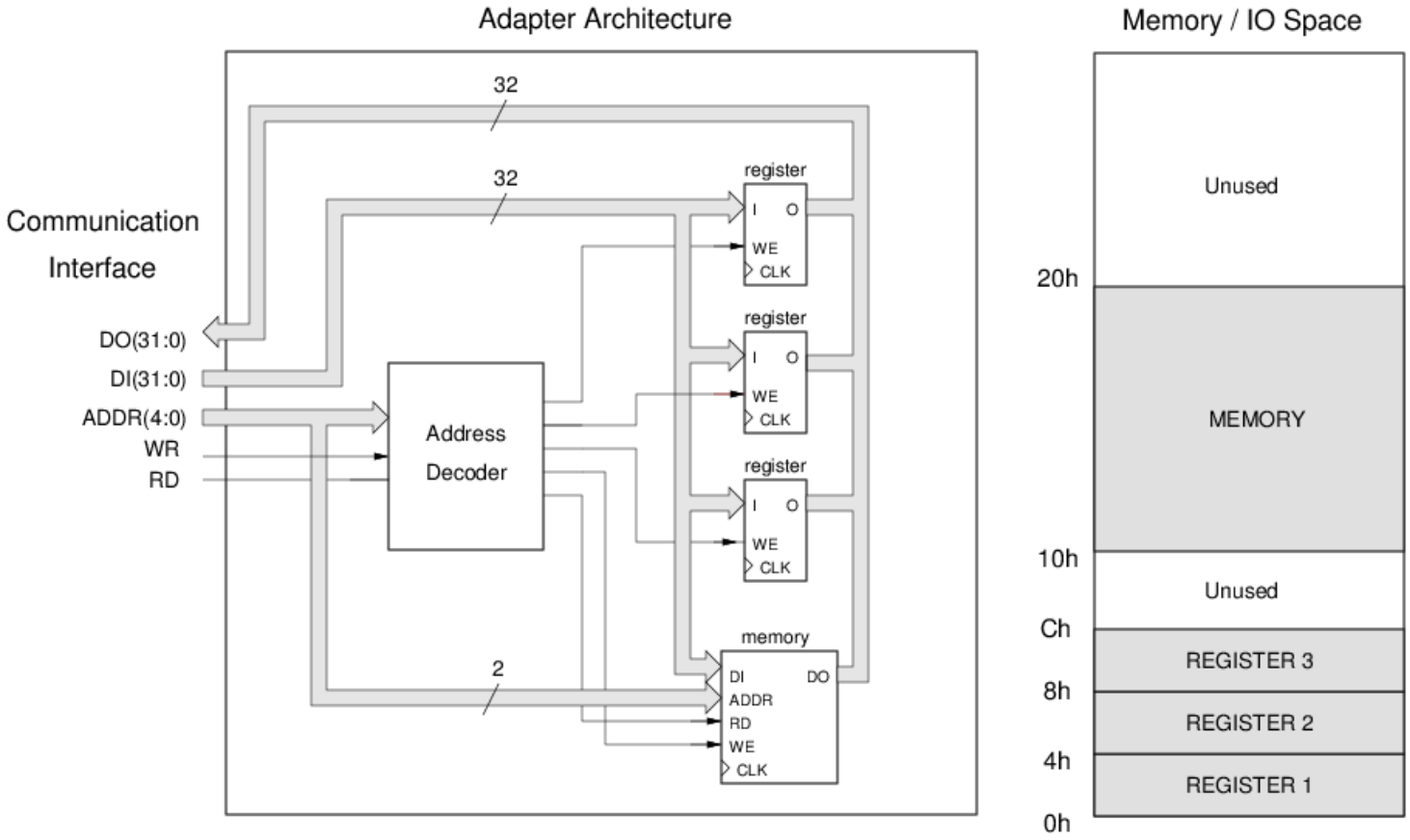
# Zápis a čtení pamětí

- Zápis dat do paměti
  - Adresový registr si pamatuje adresu, na kterou se má psát
  - V datovém registru jsou uložena data, která se mají zapsat do paměti
  - Signálem WRITE (zapiš) se data uloží do paměti
- Čtení dat z paměti
  - Adresový registr si pamatuje adresu, ze které se má číst
  - Signálem READ (čti) se data přepíší z paměti do datového registru

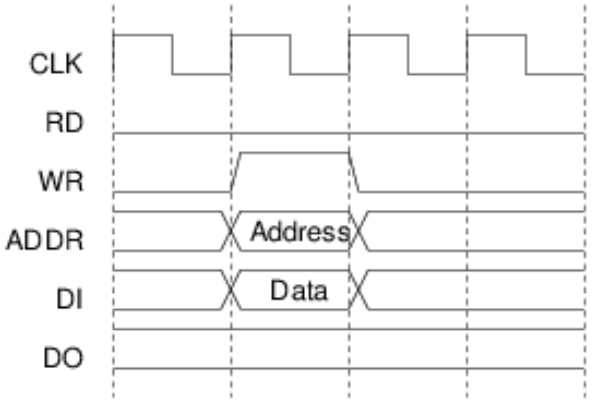


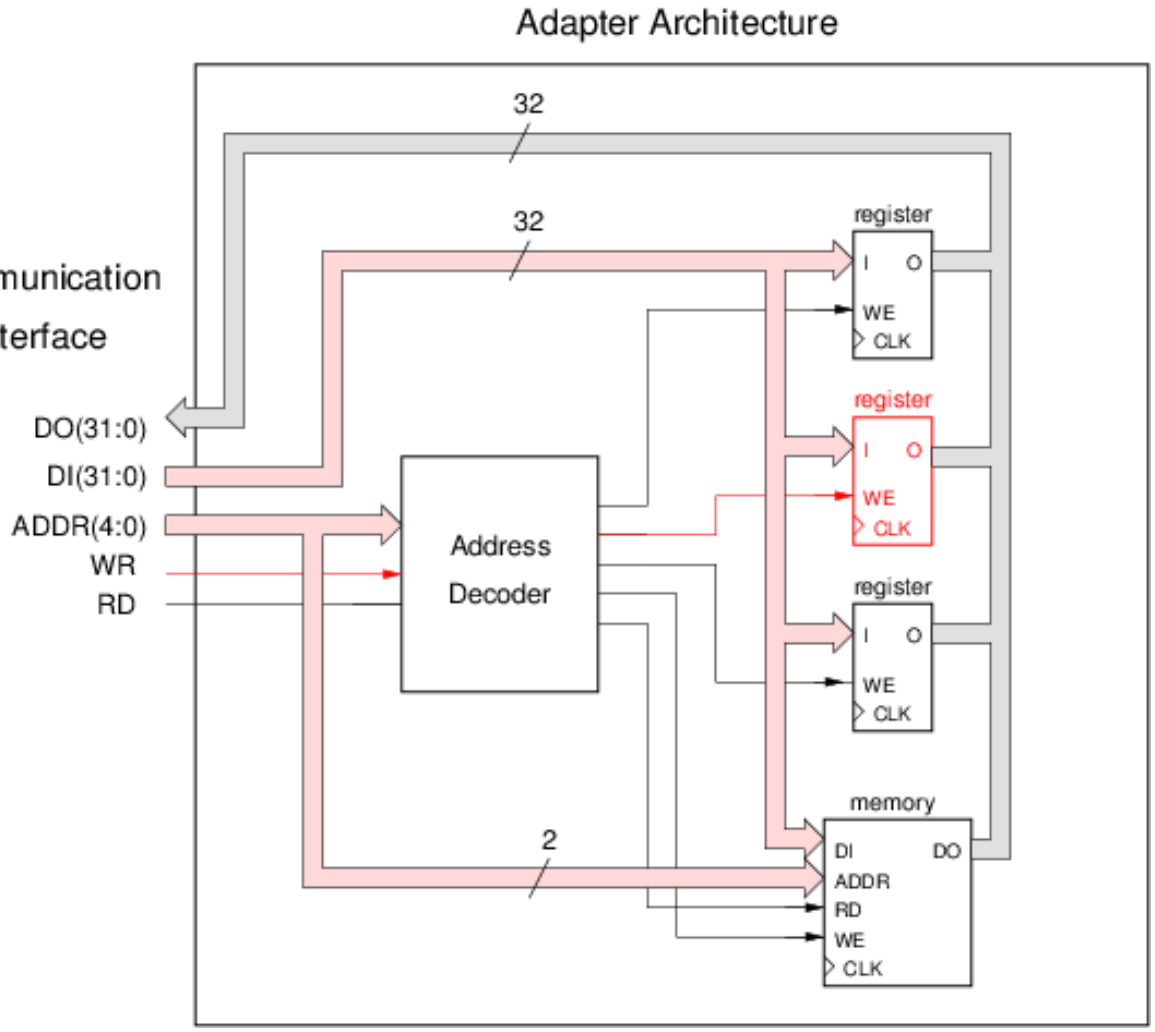
# Adresový dekodér

- Podle adresy na sběrnici detekuje, zda má být jednotka aktivní, a vybírá správný paměťový prvek pro zápisovou či čtecí operaci
  - Řídicí/stavový/datový registr
  - Paměťový blok pro operandy/výsledky

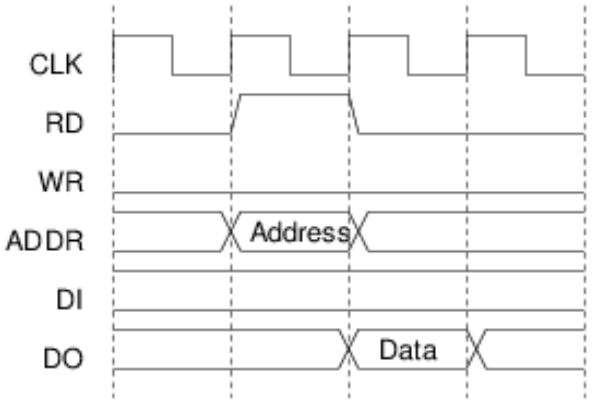


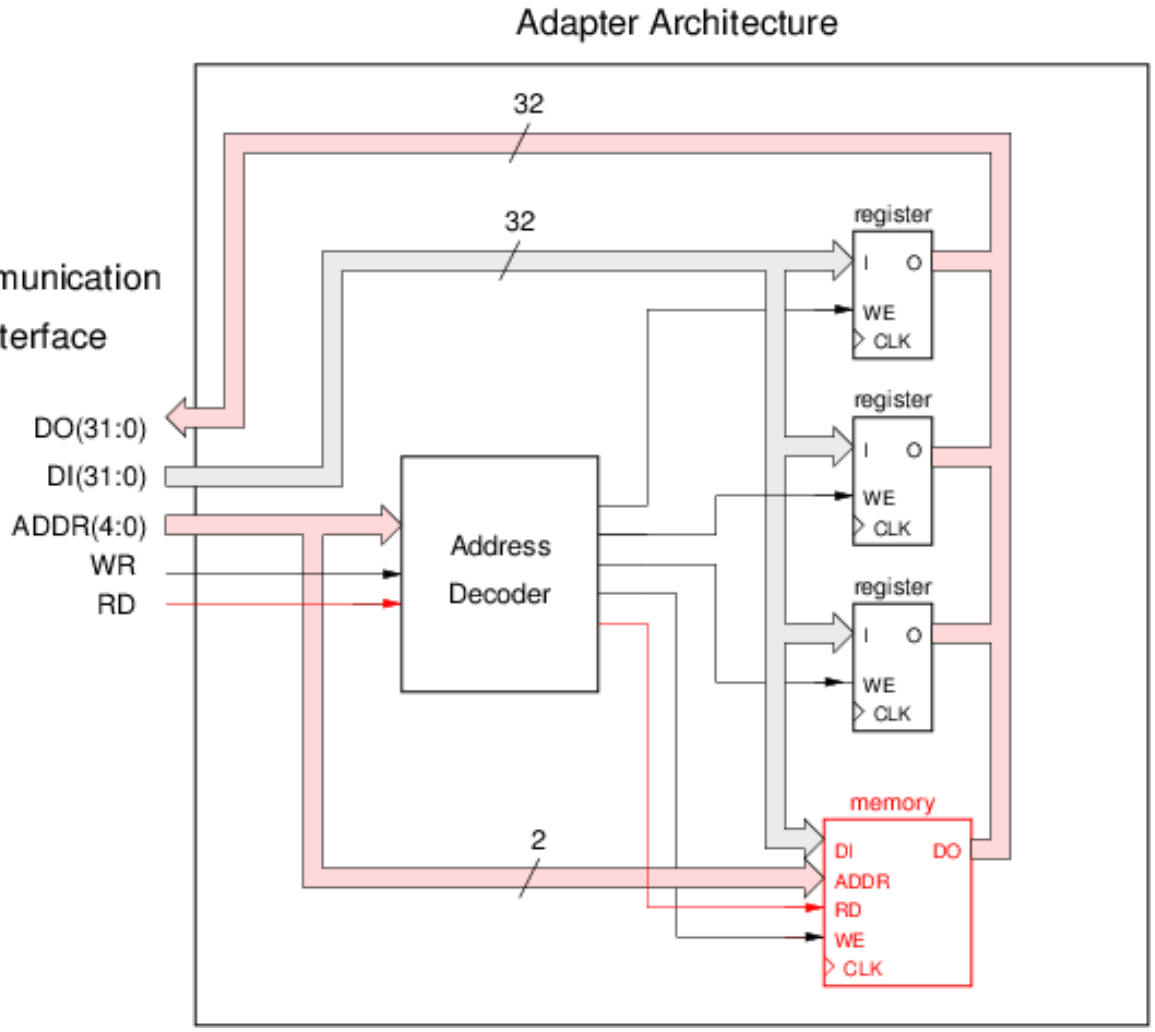
# Zápisová operace



# Čtecí operace



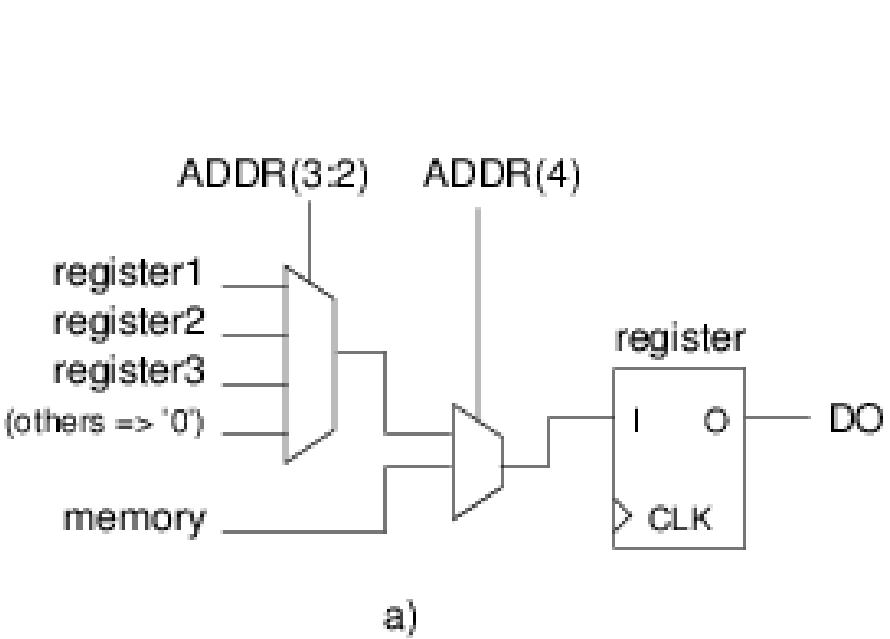



# Výběr prvku + generování WE a RD signálů

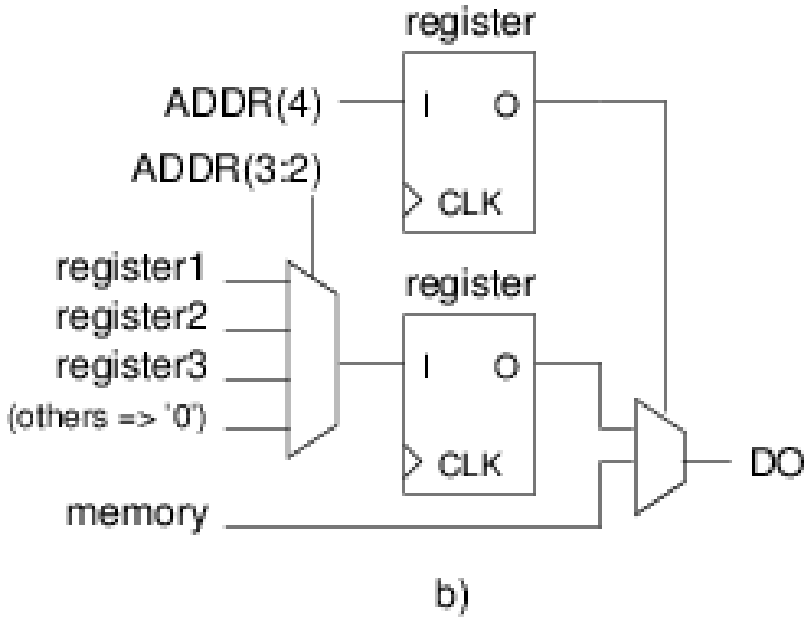
```
process(ADDR)
begin
   cs_reg1 <= '0';   cs_reg2 <= '0';
   cs_reg3 <= '0';   cs_mem  <= '0';
   case (ADDR(4)) is
      when '0'    => case (ADDR(3 downto 2)) is
                        when "00"   => cs_reg1 <= '1';
                        when "01"   => cs_reg2 <= '1';
                        when "10"   => cs_reg3 <= '1';
                        when others =>
                     end case;
      when '1'    => cs_mem <= '1';
      when others =>
   end case;
end process;

reg1_we <= cs_reg1 AND WR;   reg2_we <= cs_reg2 AND WR;
reg3_we <= cs_reg3 AND WR;   mem_we  <= cs_mem  AND WR;
```

# Výběr hodnoty výstupních dat

- Řešení s využitím výstupního multiplexoru
  a) Asynchronní čtení z paměti
  b) Synchronní čtení z paměti
- Alternativně lze implementovat s využitím třístavového budiče



a)



b)

# Aliasing paměťového prostoru