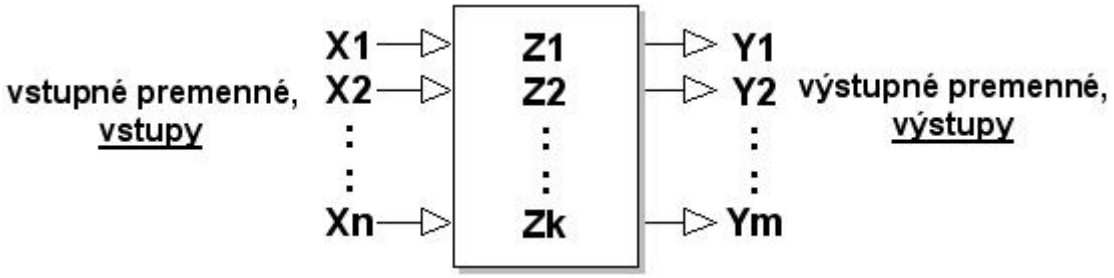
1. CHARAKTERISTIKA DIGITÁLNEHO SYSTÉMU

A. Charakteristika digitálneho systému (A1, A2, A3, A4)
B. Procesory
C. Systém ako kompozícia subsystémov
D. Metodika návrhu systémov

A. Charakteristika digitálneho systému

V tejto kapitole zopakujeme základné pojmy a doplníme niektoré pojmy a charakteristiky digitálnych (číslicových) systémov.

Digitálny systém (číslicový systém) je dynamický systém (vo všeobecnosti) so vstupnými, v čase premennými veličinami a výstupnými premennými veličinami ako aj **sprostredkujúcimi vnútornými** premennými (stavovými) veličinami. Vstupné a stavové veličiny (premenné) sú **kauzálne závislé** od vstupných veličín (premenných) . Systém možno charakterizovať takto:



A1. Vstupné, výstupná a stavové veličiny, ktoré nazývame (jednoducho) **premenné**, nadobúdajú hodnoty z **konečných** množín. Tieto **hodnoty** sa nazývajú aj **údajmi** (dátami) alebo **signálmi**.

V súvislosti s množinami hodnôt hovoríme o **údajových** (dátových) **typoch** priradených premenným. Pri ČS sú teda používajú údajové typy, ktoré pracujú s **konečnou množinou údajov** (napr.: konečná podmnožina množiny celých čísel, boolovské hodnoty, konečná podmnožina racionálnych čísel, konečná množina symbolov, r-rozmerné pole, ktorého prvky sú predtým spomenutých prvky množín, a pod.). Vstupy a výstupy systému nazývame aj primárne (vstupy, výstupy), alebo aj vstupné a výstupné porty (brány)

Množiny hodnôt – domény hodnôt pre jednotlivé premenné Xi, Yj a Zr označíme DXi, DYj, a DZr.

Celkovú situáciu na vstupe, výstupe a vo vnútri systému v danom čase t vyjadrujú hodnoty príslušných premenných v tomto čase. Túto situáciu v čase t vyjadrujeme pomocou **3 vektorov** obsahujúcich konkrétne hodnoty premenných (vstupov, výstupov, resp. stavových premenných) a to:
Vstupný vektor:

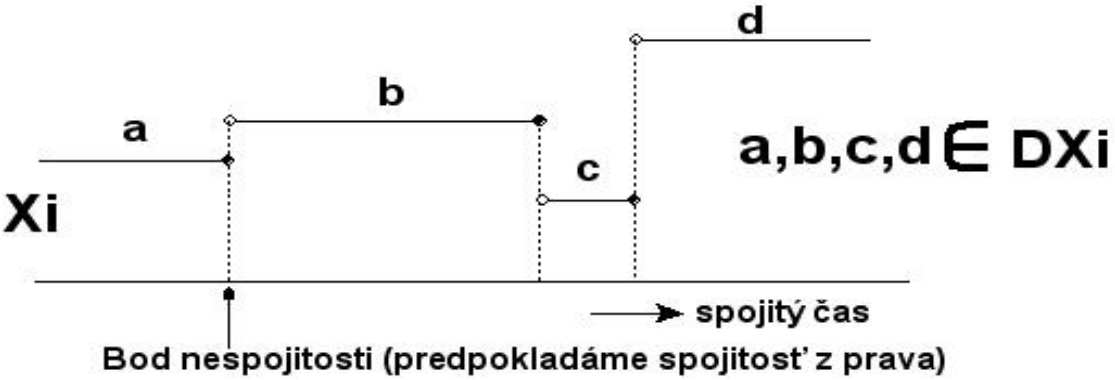
v = (<X1>,..,<Xn>); <Xi> ϵ DXi, i=1,..,n

kde symbol <Xi> značí hodnotu vstupu (v niektorom čase t)
Výstupný vektor:

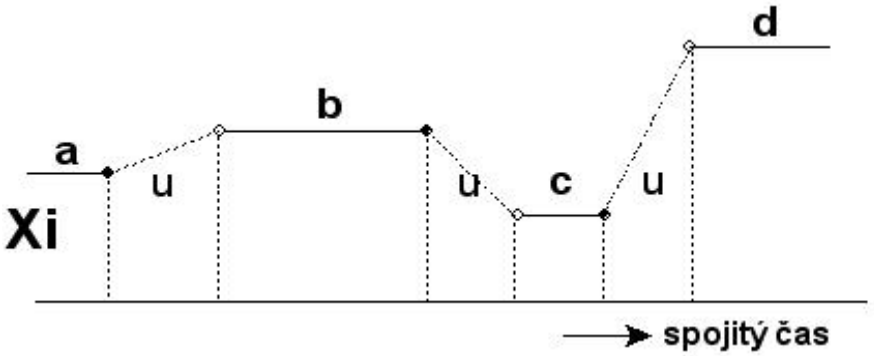
h = (<Y1>,..,<Ym>); <Yj> ϵ DYj, j=1,..,m

Stav: q = (<Z1>,..,<Zk>); <Zr> ϵ DZr, r=1,..,k ,
Stav charakterizuje vnútornú situáciu (v spracovaní informácie) a vyjadruje **pamäť histórie vstupných situácií a výsledkov** procesu pracovania informácie.

Do konečných množín hodnôt DXi, DYj, DZr dodávame jeden symbol, "u", ktorý reprezentuje **nešpecifikovanú hodnou** príslušnej premennej a do každej z troch množín vektorov {v},{h} a {q}; DV, DH a Q pridávame potom vektor (u, u,...,u). Tento vektor označíme symbolom **u** alebo u.
Správanie systému špecifikujeme v čase. Časové priebehy môžu vyzerať napr. takto:

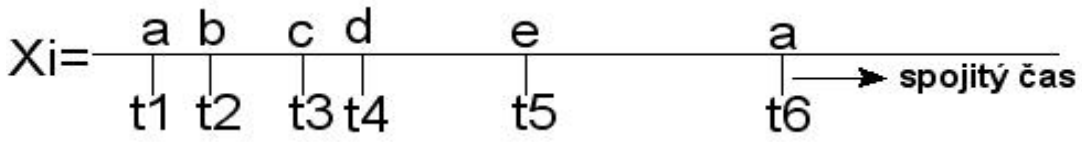


Uvedený priebeh je **istou abstrakciou voči reálnej situácií** pri zmene hodnoty premennej (tu je to „Xi"). Predpokladá **okamžitú zmenu** hodnoty, t.j. **bod nespojitosti** vo funkcií. Premenné, viazané na **fyzikálne veličiny** ako nositeľov ich hodnôt (pozri predmet Logické systémy) nemôžu však zmeniť hodnotu okamžite. Takáto abstrakcia je možná iba v prípade, ak je čas zmeny zanedbateľne krátky.
Realistický časový priebeh bude vyzerať takto:



Počas zmeny (v príslušných časových intervaloch spojitého času - pozri obr.) má premenná **Xi nešpecifikovanú „**hodnotu u".

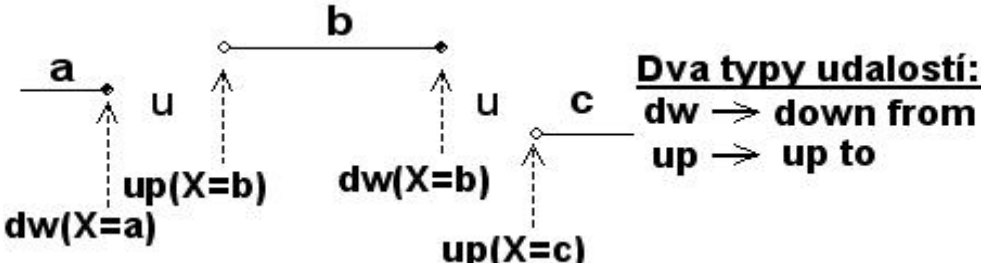
A2. Hodnoty premenných a správanie (funkcia) systému sa špecifikuje v **diskrétnom čase**, ktorý je daný postupnosťou diskrétnych bodov:



t1, **t2**, **t3**, **t4**, **t5**, **t6** je postupnosť bodov = diskrétny čas.
Píšeme: **Xi**(**t1**) = a, **Xi**(**t2**) = b, atď.

Body diskrétneho času (**d-body**) sú v digitálnych zariadeniach definované pomocou tzv. **časovacích udalostí**, zmien určitých premenných systému, alebo špeciálnych premenných (nositeľov hodín).

Udalosti budeme vo všeobecnosti viazať obyčajne na **zmeny** hodnôt premenných. Zavádzame tieto 2 **typy udalostí** označované up a dw (ilustrujeme na príklade):



Uvedené typy udalosti značíme takto up(X,v) resp. dw(X,v), alebo (pozri obr.) aj takto up(X=v), resp. dw(X=v), aby sme explicitne ukázali, že X má **hodnotu** v.

Ak zavedieme **funkciu** „t“ , ktorá každej udalosti e priradí **bod času** te, v ktorom sa objaví, t.j. t(e) = te, potom je zrejme, že pri zmene hodnoty z "a" na "b" platí:

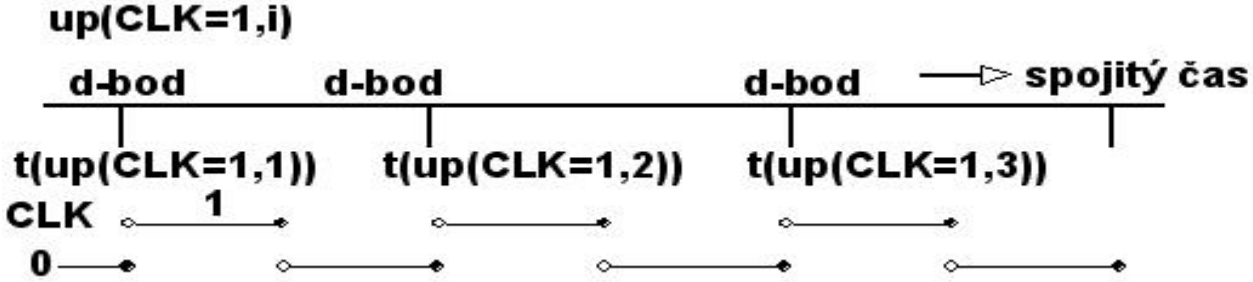
$$t(dw(X, a)) =< t(up(X, b))$$

Pretože **ten istý typ udalosti** (napr. up(X, v)) sa môže v čase opakovať **viac razy**, do zápisu udalostí pridávame index i = 1,2,3,.. a píšeme napr.

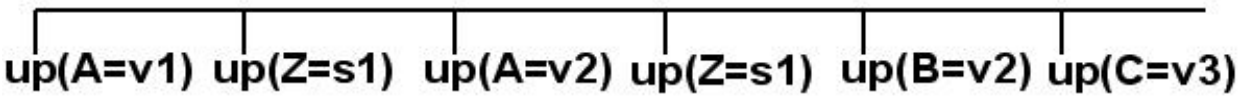
up(X, v, i), alebo up(X=v, i)

Zrejme platí:
Ak i < j, potom t(up(X=v, i)) < t(upX(X=v, j))

Časovacie udalosti sú udalosti, ktoré definujú diskrétne body (d-body) času. Typickým je tento príklad: V zariadení je špecializovaná (hodinová) premenná CLK, pričom časovacie udalosti sú hodinové signály (napr. udalosti):



Systémy, ktoré používajú práve opísanú **metódu definície** diskrétneho času (teda postupnosti d-bodov), sa nazývajú **synchrónne**. Definícia môže byť však ľubovoľná, napr. nech A,B,C sú vstupy a Z je stavová premenná d-body môžu byť definované takto:



Takáto definícia je typická pre **asynchrónne** systémy.

A3. Špecifikácia digitálnych systémov, najmä tých rozsiahlejších je zložitá. Konečné stavové stroje (automaty) ako samostatné entity sa nedajú použiť na špecifikáciu funkcie (správania) systému ako celku, ale sú vhodné iba pri špecifikácií **prvkov** systému.

V podstate je potrebné vyjadriť **závislosť** výstupných slov na stavoch a vstupných slovách.

Vstupné slovo systému **je konečná postupnosť** (reťazec) **vstupných vektorov**, ktoré systém indikuje v diskrétnom čase, na konečnej postupnosti d-bodov.

Vstupné slovo: v1 v2 v3 ... vp má určitú **dĺžku** p
v1,...,vp ϵ DV sú vstupné vektory indikované v d-boch t1, ..., tp

Výstupné slovo definujeme podobne.

h1 h2 h3 ... hp, hi ϵ DH

Vstupné a výstupné slová s rovnakou dĺžkou p, ktoré systém resp. jeho okolie indikuje v bodoch t,..., tp a ktoré sebe zodpovedajú, t.j. výstupné slovo je **reakciou** (**ozvenou**) systému na dané vstupné slovo, možno "**spliesť**" do jednej postupnosti:

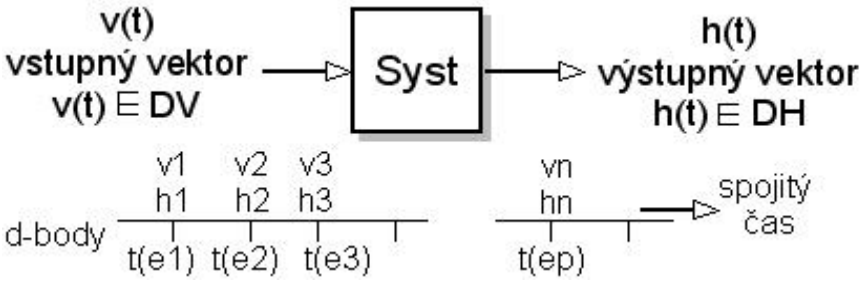
(v1;h1) (v2;h2) (v3;h3) ... (vp;hp)

Toto slovo vyjadruje **kompletnú** "komunikáciu" systému s **okolím**, nazývame ho **komunikačné slovo**.

Do komunikačného slova môžme **pridať časovacie udalosti** e1,..., ep, ktoré definujú d-body t1,..., tp, t.j. platí: t(e1)=t1,..., t(ep)=tp. Dostaneme tzv. **časované** komunikačné slovo (nazývame ho jednoducho tiež iba komunikačným slovom). Typy časovacích udalostí sú z niektorej množiny TE, ktorú je potrebné definovsť, teda e1,…, ep ϵ TE.

(v1;e1;h1) (v2;e2;h2) (v3;e3;h3) ... (vp;ep;hp)

Interpretácia takéhoto zápisu je zrejmá. Je takáto:



Pri špecifikácií správania systému možno použiť tzv. **komunikačnú množinu** W, ktorá obsahuje **všetky** (časované) komunikačné slová, ktoré sa pri spolupráci systému s okolím **môžu vyskytnúť**.

W = { w | w je komunikačné slovo, ktoré vznikne „prepletením“ vstupného slova s príslušným reťazcom výstupných vektorov}

Pri systéme, ktorý je (potenciálne) opísateľný **konečným stavovým strojom** (**FSM**)

M = (DV, Q, DH, p, v), kde

(DV a DH sú množiny vstupných vektorov, Q je množina stavov, "p" je prechodová funkcia, "v" je výstupná funkcia), platí (FA tu ynačí log. Kvantifikátor „For All“):

Pre každý možný začiatočný stav qi ϵ IQ, kde IQ je množina možných začiatočných stavov

W = { w | FA w, FA **proj**(w,DV) ϵ DV* : **proj** (w,DH) = vs (qi, proj(w,DV)) }

kde DV***** je množina **všetkých vstupných slov**, "**vs**" je **rozšírená** výstupná funkcia stavového stroja M a pre w = (v1;h1) (v2;h2) (v3;h3) ... (vp;hp) je:

proj(w,DV) = v1 v2 v3 ... vp (vstupné slovo je projekciou slova w na vstuné vektory)

proj(w,DH) = h1 h2 h3 ... hp (príslušné výstupné slovo generované FSM „M“)

Teda výstupné slovo z komunikačného slova w je práve to slovo, ktoré M vyprodukuje ako **ozvenu** na dané vstupné slovo zo začiatočného stavu q. Teda v každom slove w vo W je so vstupným slovom spletené práve takáto možná ozvena. Treba poznamenať, že s daným vstupným slovom môže byť spletené viacero výstupných slov, pretože ozveny **závisia od začiatočného stavu** qi, ktorý môže byť (podľa definície W) ľubovoľný stav z IQ, Pri opisovaní množiny **W** (výrazom, pozri ďalej), treba túto skutočnosť vyznačiť.

Komunikačnú množinu obyčajne zostavujeme s časovanými komunikačnými slovami. V Kapitole 2 sa k nej vrátime a ukážeme spôsob jej formálneho zápisu a aplikáciu.

Pri špecifikácií správania komplexných systémov sa používajú **rozličné formálne a jazykové prostriedky**. V Kapitole 2 uvedieme formálny špecifikačný model (**Model of Computation**), ktorý je dostatočne všeobecný a vhodný najmä pri opise správania na **systémovej úrovni** návrhu, vychádza v svojej podstate z opisu systému ako "čiernej skrinky" vo forme kompozície konečných stavových strojov - automatov, pričom jednotlivé stavové stroje opisujem tzv. **agentmi** a kompozíciu automatov pomocou **procesov** (programov). Agenty vstupujú do procesov ako ich prvky, ktoré zaraďujú na vykonávanie v systéme (sekvenčným alebo súbežným, konkurenčným spôsobom). Na tomto modeli je založený náš **návrhový a špecifikačný jazyk** HSSL. Tento model použijeme tu pri behavioristickej špecifikácií digitálneho zariadenia jednak na systémovej úrovni a aj pri prechode (vývoji východiskovej špecifikácie) smerom dole k špecifikácií na RT úrovni.

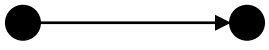
A4. Systém možno opísať nielen jeho správaním, ale aj jeho **štruktúrou** (pozri predmet Logické systémy). Štruktúru možno vo všeobecnosti formálne opísať pomocou dvojice (**E**,**C).**

E je **súbor prvkov** - **subsystémov** s explicitne daným, špecifikovaným správaním vzhľadom na **rozhranie** jednotlivých prvkov s ich kompletným okolím v danom systéme (teda cez primárne vstupy a výstupy (primárne porty) a vnútorné porty spojujúce ich s inými prvkami štruktúry).

a C je opis (často vo forme grafu) **množiny väzieb**. Ilustrujeme to na príklade

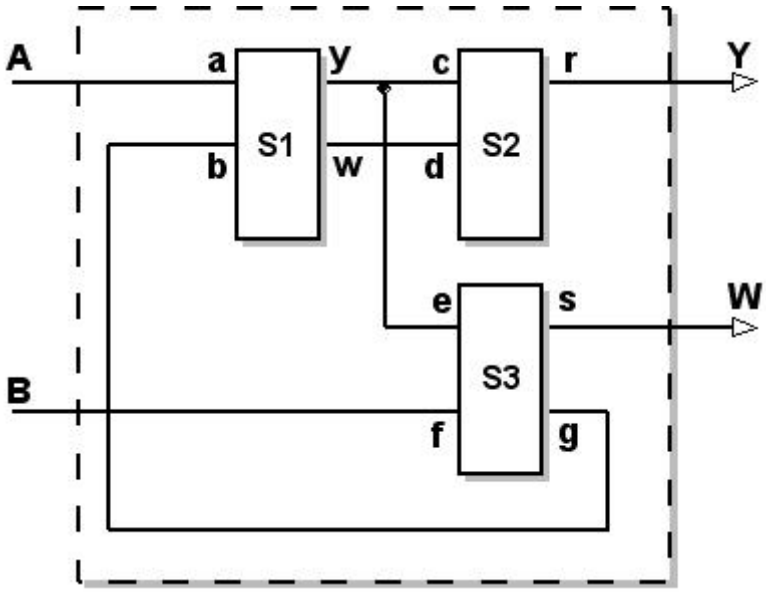
C je množina dvojíc (**i**, **j**), ktorá vyjadruje (formálne opisuje) reláciu **prepojenia** portov a to pripojenie portu „**i**“ na východiskový – signálovo zdrojový port „**j**“.

port „j“ zdroj signálu. **i**



Smer toku informácie pri spojenej dvojici portov (**i**, **j**) je, zrejme z portu **j** do portu **i** .

PRÍKLAD (štruktúra). Systém opísaný štruktúrou opísanou všeobecne dvojicou (**E**,**C**). **E** je súbor **prvkov** – **subsystémov** s explicitne daným správaním vzhľadom na ich rozhranie s ich okolím v danom systéme.



E = { S1,S2,S3 } špecifikácia správania Si je daná, známa

C = {(a,A),(f,B),(c,y),(e,y),(d,w),(Y,r),(W,s),(b,q)}

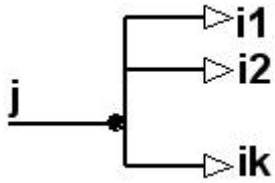
(**i**,**j**) značí implementované prepojenie (t.j., hrana grafu) "z **j** do **i**", kde

"**j**" je vstup systému alebo výstup prvku a

"**i**" je vstup prvku alebo výstup systému.

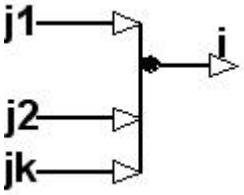
(1) typy prepojení (**i1**, **j**),...,(**ik**, **j**) sú **povolené**, ide o **rozvetvenie** vstupu zdrojového portu

j do portov **i1**, **i2**, ..., **ik**



(Angl. sa nazýva Fan-out) Počet "k" je obyčajne limitovaný

(2) typy prepojení (i, j1),...,(i, jk), t.j vstup do uzla **nie sú** spravidla povolené



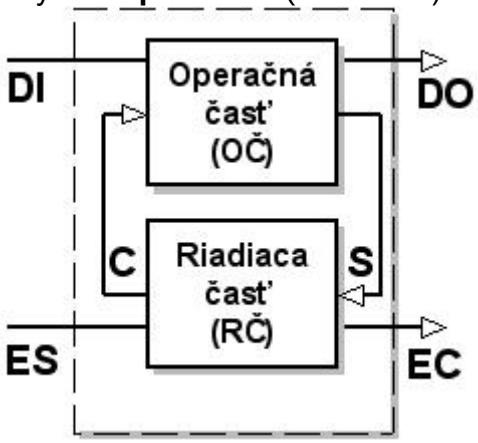
Anglicky sa takéto prepojenie nazýva **Fan-in.** Toto je možné iba výnimočne, ak je dovolené (pri danej elektronickej implementácií) toto známe **montážne prepojenie**, musí byť definovaná určitá **funkcia f(j1**, **j2**, ..., **jk**) v uzle, ktorá určuje hodnotu výstupu v „**i**“ uzle v závislosti od hodnôt jeho vstupov **j1**, **j2**, ..., **jk**. Takáto funkcia **f** sa nazýva **montážna funkcia**. Vyplýva zo spôsobu implementácie uzla v elektronickej

technologickej realizačnej báze. Pri dvojhodnotových signáloch to často to býva napr. logický súčet.

B. Procesory

Procesor je digitálny systém (zariadenie), ktorý prijíma **inštrukcie** a dáta z okolia (prostredníctvom určitých vstupov) a spracuje prijaté a/alebo vlastné vnútorne uchovávané dáta predpísaným spôsobom. Pri spracovaní, ktoré závisí od **inštrukcií**, sa používa určitý algoritmus. Rezultujúce dáta a/alebo riadiace signály sa vracajú do okolia (prostredníctvom vhodných výstupov). Ide dnes o najčastejšie aplikovanú koncepciu digitálneho systému.

Pri implementácií spracovania informácie sa predpokladá, že procesor má riadiacu časť (riadiacu jednotku) a vo všeobecnosti sa používa **procedurálny spôsob** implementácie spracovania dát a signálov, t.j. riadiaca časť riadi určité procesy spracovania. Takýto digitálny systém má v najjednoduchšom prípade štruktúru naznačenú na obrázku, ktorú nazývame **primárna** (**základná**) architektúra.



DI, **DO** je dátový vstup, resp. výstup. tento vstup a výstup je vo všeobecnosti "komplexný", t.j. je mu priradený komplexný dátový typ (napr. **DI**: boolovský n-bitový vektor pre vstup dát, resp. **DO**: dvojica boolovských vektorov pre výstup dát a adresu)
C je riadiaci vstup pre OČ; obyčajne dátového typu: r-tica boolovských vektorov, z ktorých mnohé môžu byť triviálne, t.j. jednomiestne vektory.
S je status operačnej časti; spätná väzba z OČ do RČ; údajový typ podobný ako pri C
ES, **EC** je externý status z okolia resp. externý riadiaci výstup pre okolie
(C a S často nazývame **riadiace** resp. **stavovo-informačné** signály)

Pojem inštrukcie je pomerne široký. Rozoznávame dve veľké triedy inštrukcií:
-- signálové, neštruktúrované
-- štruktúrované

Signálové (**neštruktúrované**) **inštrukcie** obsahujú iba informáciu o predpísanej operácií s daným algoritmom, ktorá sa má už s implicitne špecifikovanými, vnútornými, a/alebo zvonku privádzanými dátovými štruktúrami uložených (resp. privádzaných) na fixných miestach (vstupoch). Reprezentuje ich spravidla logický signál

PRÍKLAD: RESET, INTERRUPT, ktoré sú priradené logickým vstupom systému; alebo COMPLEMENT, ktorá predpisuje logickú negáciu boolovského vektora uloženého vo fixnom mieste.

Štruktúrované inštrukcie obsahujú okrem predpísanej operácie aj smerníky na údajové štruktúry, ktoré vstupujú do spracovania a teda ide o univerzálne inštrukcie spracujúce inštrukciou identifikované dáta.

PRÍKLAD: ADD, ktorá predpisuje sčítanie dvoch celých čísel, pričom špecifikuje kde sú tieto dáta, na ktoré treba tentoraz ADD aplikovať. Miesta vo vnútornej a/alebo vonkajšej pamäti, v ktorých sa uchovávajú dáta a kde sa má preniesť výsledok sú spravidla v inštrukcií explicitne voliteľné.

Pri univerzálnych procesoroch (pozri ďalej) existujú rozličné iné triedenia (napr. známe CISC - RISC inštrukcie).

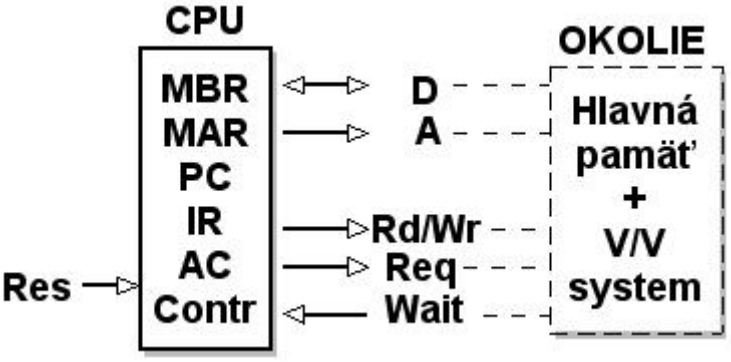
Z hľadiska **spôsobu privádzania inštrukcií** do procesora procesory rozdeľujeme na aktívne a pasívne

Aktívne procesory

Riadiaca jednotka je uspôsobená na automatické vyberanie usporiadanej postupnosti inštrukcií (strojového programu) z vonkajšieho pamäťového média, ktorým je spravidla pamäť typu RAM a prenášanie do procesora. Spravidla, popri neštruktúrovaných pracujú aj so štruktúrovanými inštrukciami a sú schopné lokalizovať (adresovať) a vyberať dáta z vnútorných a vonkajších pamäťových médií alebo naopak ukladať tieto štruktúry vo vonkajších médiách.

PRÍKLAD: Patria sem napríklad známe CPU počítačov typu SIMD; nižšie uvedieme príklad zjednodušeného aktívneho procesora s pamäťovo - orientovanou architektúrou.

Špecifikácia procesora (CPU):

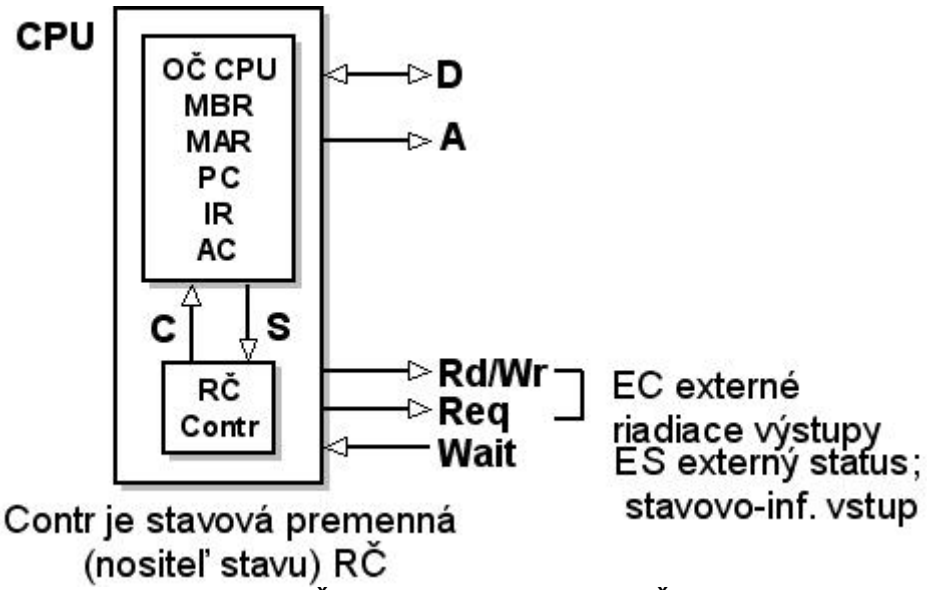


Vstup inštrukcie a údajov z okolia a výstup údaja do okolia:
D...údajový vstup-výstup 32 bit; údajový typ: **údaj** alebo **inštrukcia** (pozri ďalej)
A.. adresovací výstup 30 bit; údajový typ: **adresa** (pozri ďalej)

Riadiace vstupy a výstupy (údajový typ: logický signál):
Rd/Wr.Vyberanie inštrukcie z okolia (pamäti) a čítanie resp. zápis údaja (cez D) z alebo do okolia (t.j. pamäti alebo V/V systému).
Aktívny výstupný signál pre výstup **Rd/Wr** je:
vyberanie inštrukcie alebo čítanie údaja Rd/Wr = 1
zápis údaja Rd/Wr = 0
Req....Procesor požaduje čítanie inštrukcie alebo údaja, resp.
požaduje zápis údaja. Aktívna hodnota výstupného signálu Req = 1

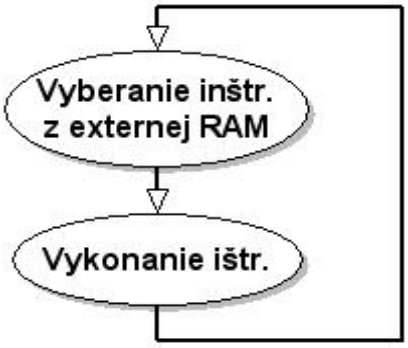
Wait...Okolie informuje o potrebe čakania na čítanú inštrukciu
alebo údaj, resp. na ukončenie cyklu zápisu vyslaného údaja.
Res... Nulovací vstup procesora
Stavové (vnútorné) premenné procesora:
MBR...Vyrovnávací register pre inštrukciu alebo údaj; údajový typ: **inštrukcia** alebo **údaj** (pozri nižšie)
MAR...Vyrovnávací register pre adresu; údajový typ **adresa**
PC....Programové počítadlo (ukazovateľ inštrukcie); údajový typ: adresa, t.j.: celé číslo z intervalu < 0, + 2**30)
IR....Register inštrukcie (inštrukčný register);
údajový typ: inštrukcia, t.j.:
záznam (record)
Opkod: symbol z množiny { LD,ST,ADD,BRN }
Adresa: celé číslo z intervalu < 0, + 2**30)
operácia + s modulom 2**30
AC...Register Akumulátor (pre operand a výsledok operácie);
údajový typ: **údaj**, t.j.celé číslo z množiny < - 2**31, + 2**31); + v doplnkovom . kóde
Contr...stavová premenná vyjadrujúca stav riadenia (pozri neskôr)

Hrubá štruktúra procesora (CPU



C...riadiace vstupy OČ, pomocou ktorých RČ riadi vykonávanie celého **inštrukčného cyklu** (pozri obr. dole)**:**

1. Vyberanie inštrukcie z RAM v okolí a
2. Vykonanie inštrukcie v globálnej OČ, teda OČ CPU a v okolí



S...stavovo-informačné vstupy RJ, ktorými OČ CPU informuje RČ o svojom stave pri vykonávaní inštrukcie
ES...externý stavovo-informačný vstup Wait z okolia
EC...externý riadiaci výstup (Rd/Wr a Req) pre okolie

Inštrukcie nášho CPU sú štyri a to:

Súčet (ADDition): formát = ADD, adresa;
vykoná sa súčet obsahu AC a obsahu externej pamäte RAM na mieste s adresou "adresa" a výsledok sa objaví ako nová hodnota (obsah) AC. Teda

AC := AC + M[adresa].

Čítanie údaja (LD => LOad Data): formát = LD, adresa;
prenesie sa údaj **z externej pamäte RAM** z miesta s adresou "adresa" alebo z portu vo V/V systému s adresou "adresa" na stavovú premennú AC (zapíše sa do registra AC).

Zápis (uloženie) údaja (ST => STore data): formát = ST, adresa;
prenesie sa údaj z AC **do externej pamäte RAM** na miesto s adresou "adresa" alebo do portu V/V systému s adresou "adresa".

Skok (rozvetvenie) podľa znamienka hodnoty **AC** (BRanch if Negative => BRN):

formát = BRN, adresa;

ak je hodnota AC záporná, potom sa nastaví PC na hodnotu "adresa", v opačnom prípade sa PC nastaví na PC + 1.

Vidno, že ide o inštrukcie, ktoré sme nazvali **štruktúrované.**

Poznámka: Ako je známe z klasifikácie architektúr, aktívne procesory so štruktúrovanými inštrukciami sa rozdeľujú aj na dve triedy:

1. S **komplexnou množinou inštrukcií**; tradične sa označuje skratkou CISC (Complex Instruction Set Computing); napríklad procesory radu 86 firmy Intel (v systémoch PC)

2. S **redukovanou množinou inštrukcií**, označované skratkou RISC (Reduced Instruction Set Computing); napr. procesory firmy SUN alebo firmy Digital (ALFA).

Pasívne procesory

Dostávajú inštrukcie zvonku, nie sú uspôsobené na samostatné vyberanie programu z vonkajšieho média; môžu byť však uspôsobené na **adresovanie dátových štruktúr vo vonkajších médiách** a ich prenos do procesora ako aj na ukladanie dátových štruktúr do týchto médií.

PRÍKLAD: Aritmetický ko-procesor

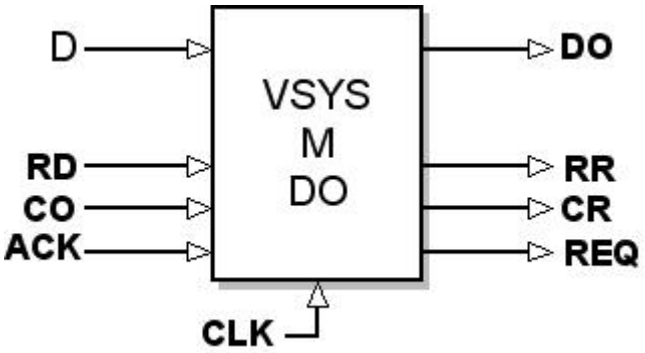
Z **hľadiska aplikácie** rozdeľujeme procesory na **univerzálne**, a **špeciálne**, **aplikačno-špecifické**

Univerzálne procesory tvoria základnú súčasť počítačov vo funkcií **CPU** sú uspôsobené na spracovanie podľa ľubovoľného algoritmu. Takéto systémy sa dnes produkujú ako štandardné integrované obvody (IO) typu VLSI, prípadne SoC, alebo ako séria takýchto IO.

Aplikačno-špecifické (A-S) procesory sú uspôsobené univerzálne procesory na **špeciálne funkcie**. Sú vyrábané ako štandardné IO vo veľkých sériách, alebo ponúkané ako IP súčiastky - pozri ďalej (napr. procesory na spracovanie signálov, grafické a aritmetické ko-procesory, DMA procesory, a pod.) alebo **na zákazku** ako v menších množstvách ako AS IO (napr. špeciálne pasívne procesory pre tzv. vnorené aplikácie)

A-S procesory tvoria **rozsiahlu triedu** do ktorej zaraďujeme aj množstvo rozmanitých pasívnych procesorov s jednoduchými inštrukciami. Pre ilustráciu uvedieme (školský) príklad A-S procesora, ktorý budeme používať v prednáškach na ilustráciu návrhu). Ide o pasívny procesor

PRÍKLAD: **A-S** procesor VSYS



Procesor spracováva 16-bitové údaje (celé čísla v doplnkovom kóde), ktoré sa privádzajú cez dátový vstup **D**. Má dve neštruktúrované inštrukcie, ktoré sa nastavujú zvonku prostredníctvom logických vstupov **RD** (čítanie a prenos poľa údajov do procesora) a **CO** (výpočet súčtu údajov)

RD=1 je aktívny signál, ktorý naštartuje vykonanie inštrukcie **R**EA**D**IN, prenesenie 16 čísel d1,..,d16 cez D do vnútornej RAM pamäti, reprezentovanej premennou M typu pole 16 čísel a nastavenie signalizačného výstupu RR=1 (**R**eadin **R**eady) informujúceho okolie o skončení operácie.

CO=1 je aktívny signál, ktorý inicializuje vykonanie inštrukcie **CO**MPUTE, výpočet aritmetického súčtu 16 čísel čítaných z vnútornej RAM (poľa M), vyslanie výsledku na výstup DO (zároveň stavovej premennej) a nastavenie signálu CR=1 (**C**ompute **R**eady) informujúceho o skončení operácie.

Premenné **REQ** (REQuest) a **ACK** (ACKnowledge) sa používajú pri asynchrónnom alebo semi-synchrónnom prenose poľa čísel (zo vzájomným potvrdením) z okolia do procesora cez vstup **D** počas vykonávania inštrukcie READIN.

C. Systém ako kompozícia subsystémov

Medzi digitálne systémy patria **veľmi zložité a rozsiahle systémy**, ktorých špecifikácia správania a návrh štruktúrnej implementácie sa nedajú zvládnuť ako u jedného koherentného celku (čiernej schránky). Pri takýchto systémoch je nevyhnutné vychádzať z určitej štruktúry, HW platformy, ktorej prvky predstavujú

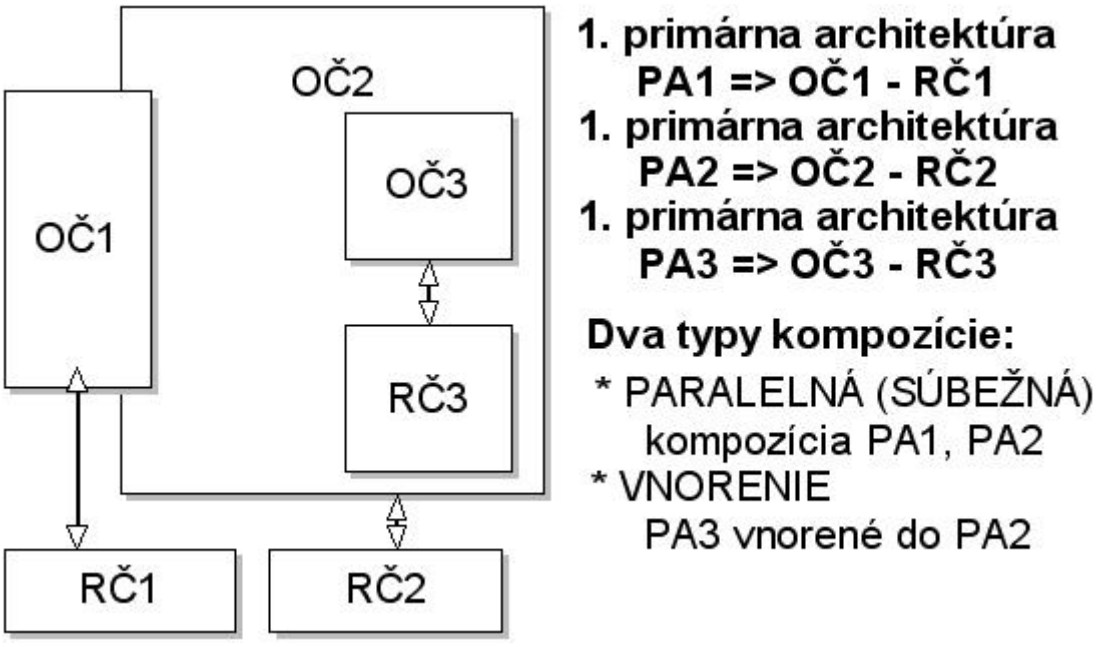
-- **hotové** subsystémy (knižničné moduly, subsystémy, ktoré boli vypracované v minulosti a znova sa použijú, "**reuse**", pozri ďalej **IP** (virtuálne) súčiastky
-- prvky, subsystémy, ktoré sa budú **navrhovať a implementovať pre daný prípad**, t.j. možno ich špecifikovať a implementovať ako koherentné systémy.

Komplexné a rozsiahle HW systémy musíme teda chápať ako **kompozíciu** jednoduchších vzájomne komunikujúcich **subsystémov**. Návrh takejto kompozície, ktoré v podstate obsahujú rôzne procesory, pamäťové systémy (napr. SRAM, DRAM, ROM) a prepojovacie subsystémy (napr. zbernice a iné prepojovacie siete), patrí do úrovne návrhu **HW systému**. V súčasnosti sa ráta s umiestnením takýchto veľkých HW systémov na jednom čipe s desiatkami miliónov tranzistorov (systém na čipe **SoC**, pozri ďalej).

V tomto predmete sa budeme venovať najmä **metodike a systematickému prístupu k návrhu HW procesorov** a to aplikačno-špecifických. Takáto znalosť v oblasti návrhu HW/SW štruktúr je aktuálna aj u malých a stredných firiem pôsobiacich v oblasti počítačového inžinierstva pri tvobe špeciálnych, vnorenýchHW/SW zariadení.

V ďalšom uvedieme základnú koncepciu kompozície (rozkladu) systému. Budeme predpokladať, že systém tvorí viacero komunikujúcich **primárnych architektúr** (PA). Vysvetlíme to na (abstraktnom) príklade.

Kompozícia primárnych architektúr

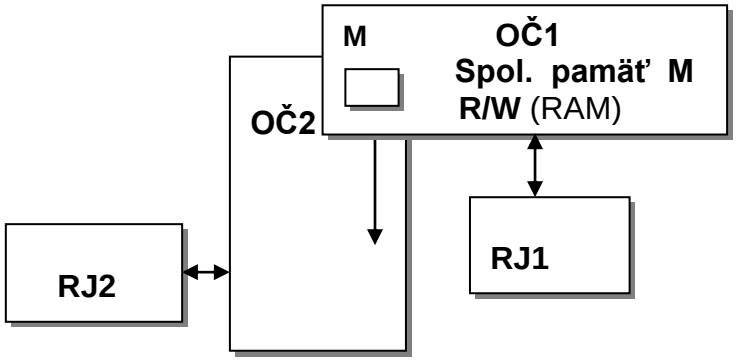


PRÍKLAD: Jeden z jednoduchších a dostatočne známych príkladov kompozície sú procesory Intel systému X 86, ktoré už pri svojich prvých typoch využívali kompozíciu dvoch PA: EU - Execution Unit a BIU - Bus Interface Unit v konkurenčnom postavení s prekrytím OČ v spoločných pamäťových prostriedkoch, napr. pamäť frontu bytov inštrukcií

KOMUNIKÁCIA medzi procesormi v kompozícií primárnych architektúr

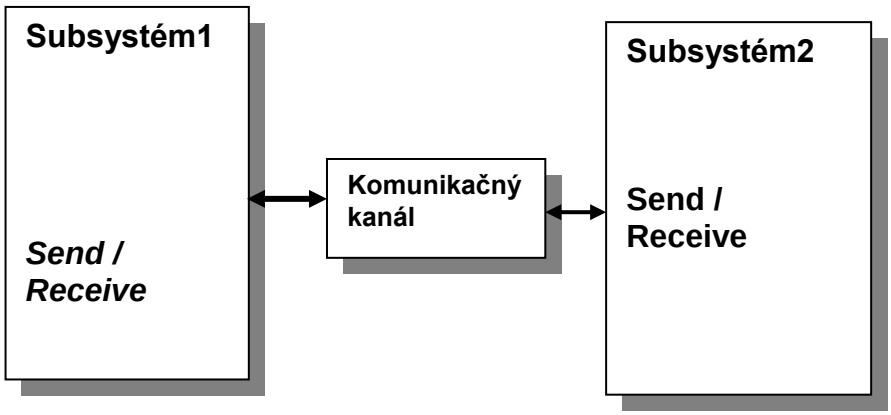
Všimneme si ešte spôsoby komunikácie dát v kompozícií primárnych architektúr (procesorov) medzi jednotlivymi primárnymi procesormi. Existuje mnoho spôsobov medzi dvojbodovými a sieťovými prepojeniami. Tu uvedieme iba dva všeobecné princípy pri výmene dát a to Výmena cez spločnú pamäť (shared memory) a výmena odovzdávaním správ (message passing)

Výmenu dát cez spoločnú pamäť znázorňuje obrázok.



OČ oboch primárnych architektúr sa prekrývajú a v prekrytí je pamätový subsystém **M**, ktorého riadenie je v dosahu oboch riadiacich jednotiek RJ1 a RJ2. Tieto jednotky riadia čítanie, resp zápis údaja (konvenciou - protokolom stanoveného balíka údajov) z príslušnej strany. Údaj, resp. balík údajov je potom k dispozícií príslušnej OČ, ktorá s nim pracuje.

Výmena dát systémom **odovzdávaním správ** je naznačená na ďalšom obrázku.

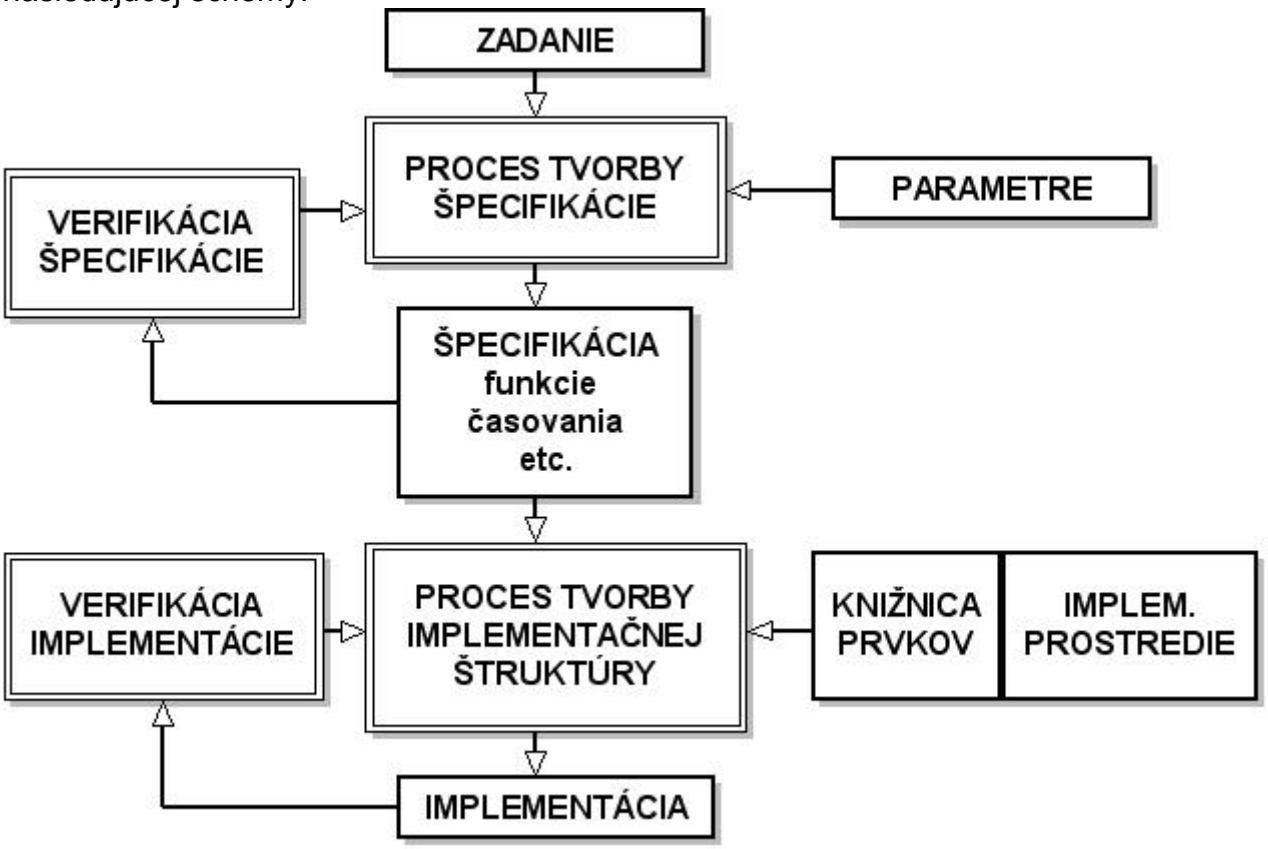


Dva subsystémy (procesory – primárne architektúry) si vymieňajú údaje pomocou osobitného komunikačného kanála. Inštrukciou “send” vo vysielacom subsystéme (vysielači) sa pripraví želaný údaj (prípadne balík údajov) a vyšle sa do kanála splu so správou o jeho vyslaní. Prijímací subsystém (prijímač) si údaj (balík) s kanála preberie a posle spravu o jeho prebratí.

Komunikačný kanál môže byť napr. realizovaný špeciálnym procesorom ako je DMA jednotka v univerzálnych počítačoch. Na druhej strane to môže byť aj vyrovnávacia pamäť (buffer) pričom vlastný prenos úfajov je riadeny RJ oboch systémov. V druhom prípade miožno tento systém výmeny dát zobraziť na prípad so spoločnou pamäťou (ktorou je vyrovnávacia pamäť.

D. Metodika návrhu systémov

V predmete Logické obvody sa venovala pozornosť návrhu digitálnych zariadení na úrovni logických obvodov Na tejto úrovni ako aj na hierarchicky vyššej úrovni abstrakcie, na úrovni registrových prenosov (RTL – Register Transfer Level) sa postupuje podľa nasledujúcej schémy.



Ide o návrh metódou **zhora nadol**, ktorý tvorí základnú paradigmu pri systematickom návrhu. Pri návrhu na RT úrovni ide často o návrh koherentných autonómnych subsystémov (napr. takých, ktoré sú implementované na jednom čipe) Na RT úrovni sú prvkami navrhovanej štruktúry registre, funkčné jednotky (napr. sčítačka, násobička a pod.) a multiplexné obvody pre nastavenie smerovania dát z ich zdrojov do určených cieľových vstupov prvkov (štandardné multiplexory, zbernice a pod). Pri opise špecifikácií ako aj implementácií systémov sa používajú rôzne opisné jazyky, nazývané „návrhové jazyky, napr. VHDL alebo Verilog. Ide o všeobecne aplikovateľné štandardné jazyky, ktoré možno použiť pri opise správania (špecifikácií) aj štruktúrnych implementácií na RT úrovni a na logickej úrovni. V tomto predmete sa budeme venovať práve **návrhu** (syntéze) **štruktúrnej** (**HW**) implementácií na úrovni **registrových prenosov** (na **RT** úrovni). Priamo tým nadviažeme na návrh na úrovni logických obvodov preberaný v predmete zaoberajúcim sa návrho digitálnych systémov na logickej úrovni abstrakcie.

Na RT úroveň návrhu sa dostávame z najvyššej úrovne abstrakcie – z tzv. **systémovej** úrovne. Na tejto úrovni sa navrhne štruktúra zostavená z (prvkov) modulov, ktorých špecifikácie správanie vzhľadom na rozhrania s inými prvkami sú už zostavené a je už rozhodnuté akým spôsobom budú jednotlivé moduly implementované. Pôjde o dva známe spôsoby implementácie digitálnych systémov a to **SW spôsob** – exekúciou určitého aplikačného programu pod niektorým operačným systémom na vnorenom počítači, alebo **HW spôsob** v štruktúre (na RT a logickej úrovni), ktorú treba navrhnúť (urobiť syntézu)

alebo konfigurovať na **pogramovateľnej logike**, alebo často použiť **hotový obvod**, ktorý ako sa ako tzv. virtuálna súčiastka zakúpi od niektorého producenta. Takúto súčiastku (anglicky označovanú aj skratkou IP – **I**ntellectual **P**roperty **C**omponent) možno vložiť a realizovať v systéme na čipe (SoC – System on Chip). Použitie IP súčiastok sa stáva nevyhnutným najmä pri **návrhu rozsiahlejších softwérovo-hárdverových** (**SW/HW**) **systémov**, ktoré sa už teraz začínajú a v blízkej budúcnosti sa budú čoraz viac implementovať na jednom čipe (ako SoC).

S opačným procesom návrh, s **návrhom zdola nahor**, sa stretávame pri **tvorbe knižníc** HW prvkov a modulov ako aj IP súčiastok. Tu sa vytvoria moduly, obvody z jednoduchších prvkov, ktoré sú k dispozícií ako hotové, implementované úž aj na úrovni elektronickej realizačnej báze. V tomto prípade sa teda z primitívnych alebo jednoduchších štruktúrnych prvkov (ydola nahor) tvoria prvky so zložitejším správaním s preverenou korektnosťou funkcie vzhľadom na dané rozhrania (vstupy a výstupy) spojujúce ich s okolím.

Okrem funkčnej (behavioristickej) špecifikácie v diskrétnom čase (v cykloch – teda v tzv. presnosti na cykly) je treba pre knižničné prvky špecifikovať aj **obmedzenia v časovaní zmien** vstupov a výstupov v spojitom čase. Systém takýchto obmedzení nazývame v tomto predmete „**časovacia disciplína**“. Časovacia disciplína teda špecifikuje to, ako majú „vyzerať“ časové priebehy hodnôt premenných rozhrania. **Rozširuje cyklovú presnosť** na presnosť **v reálnom čase**.

Na systémovej úrovni postačuje obvykle presnosť opisu správania modulov (prvkov) štruktúry systému na cykly, na RT úrovni je podstatné aj splnenie časovej disciplíny v reálnom čase.

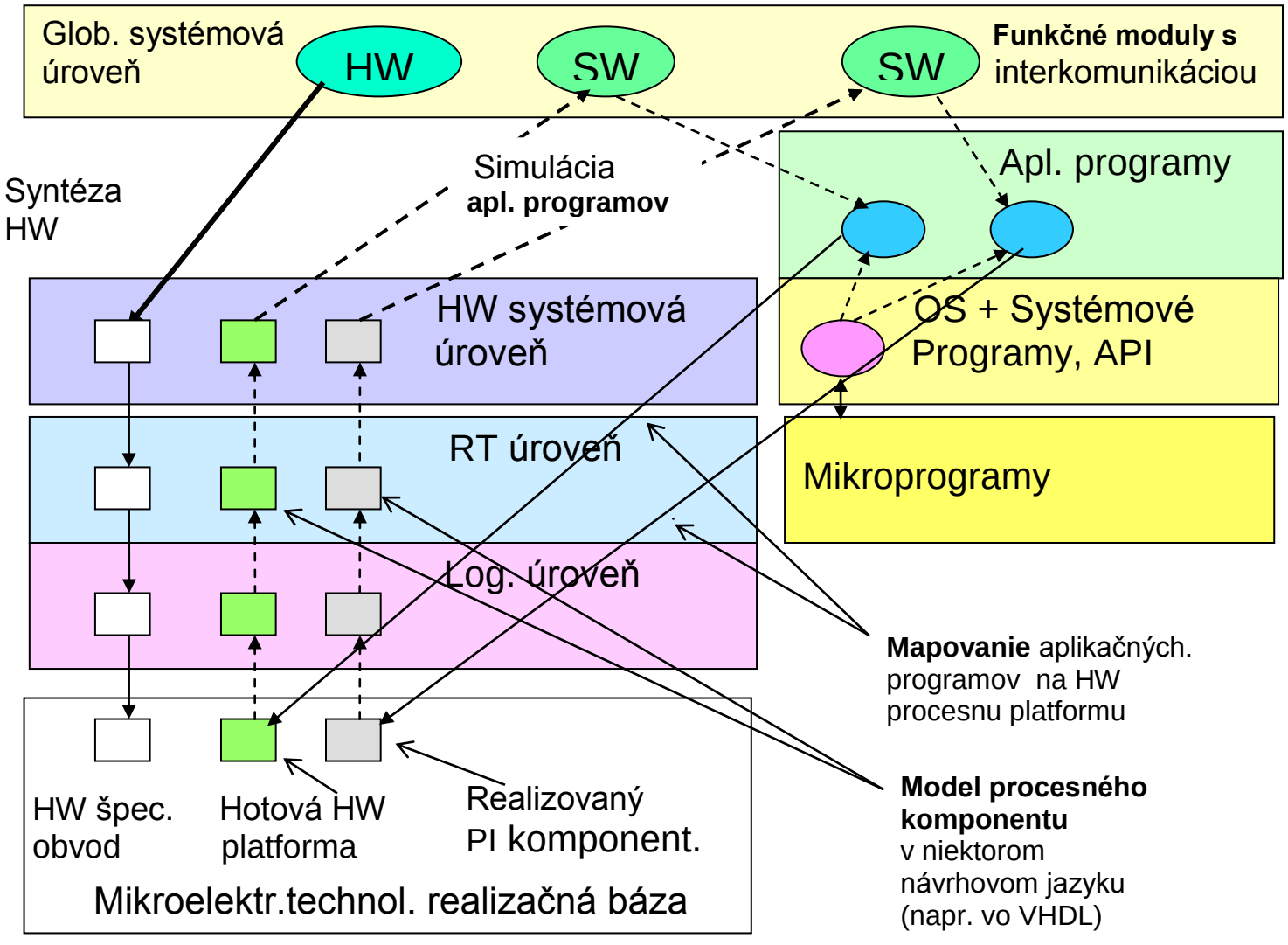
Na záver 1. Kapitoly uvedieme **hierarchickú štruktúru úrovní abstrakcie** pri analýze a návrhu digitálneho systému. Na najnižšej úrovni abstrakcie sa nachádza **elektronická technologická realizačná báza**, ktorá tvorí základ pre tvorbu digitálnych a iných zmiešaných (digitálno - analogových - mechanických) mikroelektronických systémov.

Na najvyššej hierarchickej úrovni je **globálna systémová úroveň**, na ktorej sa začína s formuláciou špecifikácie systému a na ktorej treba rozhodnúť, ktoré jej moduly (funkčné subsystémy) budú realizované v HW metódou resp. SW metódou. Teda tu treba **rozhodnúť o rozmiestnení modulov na SW a HW implementovaných a o štruktúre HW platformy.** Treba royhodnúť o tom, ktoré moduly sa budú navrhovavať a implementovať **HW spôsobom** a ktoré – **SW spôsobom**. V druhom prípade treba rozhodnúť aj o tom, aké počítačové **HW platformy** (aktívne procesory s pamäťovým subsystémom a I/O subsystémom), sa navrhnú alebo použijú ako hotové pri SW implementácií týchto modulov. Rozmiestnenie modulov podľa spôsobu implementácie je založené na odhadoch optimalnej výkonnosti, spotreby energie a prípadne aj plochy pre HW platformu. Ide o náročný návrhový proces, ktorý vyžaduje **formálne opisy systému** na systémovej úrovni pomocou osobitných systémových návrhových a špecifikačných jazykov – jazykov SDSL (napr. náš HSSL alebo v praxi často používaný SystemC, HandelC, a pod., ale aj jzykz založené na jazyku Java a UML-2) ako aj rad nevyhnutných, osobitných špeciálnych **SW nástrojov** pre návrh, teda na vývoj (zjemňovanie) špecifikácie navrhovaného systému, na syntézu HW na úrovni RT, na overenie funkčnosti systému a jeho štruktúrnych prvkov, na verifikáciu špecifikácií tvoreného systému pri jeho vývoji a na verifikáciu riešenia na RTL voči špecifikácií, na vyriešenie rozmiestnenia spôsobu implementáciu a na optimalizáciu navrhovaného systému. Popri jednotlivých

funkčných moduloch, významnú úlohu hrá aj optimálny návrh a verifikácia komunikácie dát a signálov medzi modulmi (subsystémami) prepojujúcej moduly realizované rozličnými spôsobmi (HW,SW). Návrhom digitálneho systému na systémovej úrovni sa v tomto predmete dotkneme **iba okrajovo**. V budúcich rokoch bude venovaný tejto problematike pre jej aktuálnosť a význam osobitný predmet.

Medzi hierarchicky najvyššou úrovňou abstrakcie - globálnou systémovou úrovňou a realizačnou technologickou elektronickou bázou je **viacero HW a SW úrovní**, ktoré vidno na **nasledujúcom obrázku**.

Pri **implementácií systémových modulov SW spôsobom** je potrebné pracovať aj so SW systémovou úrovňou. Aplikačné programy, ktorých exekúcia na daných procesoroch HW platformy bežia pod určitým (obvykle špecializovaným) **operačným systémom** (OS) a spolupracujú so špeciálnymi programami pre **aplikačné rozhranie** (API) s navrhovaným komunikačným subsystémom prepojujúcim implementované systémové moduly (HW, SW implementované s danými rozhraniami a protokolmi). Problematike SW spôsobu implementácie funkčných modulov sa nebudeme v danom predmete podrobnejšie venovať.

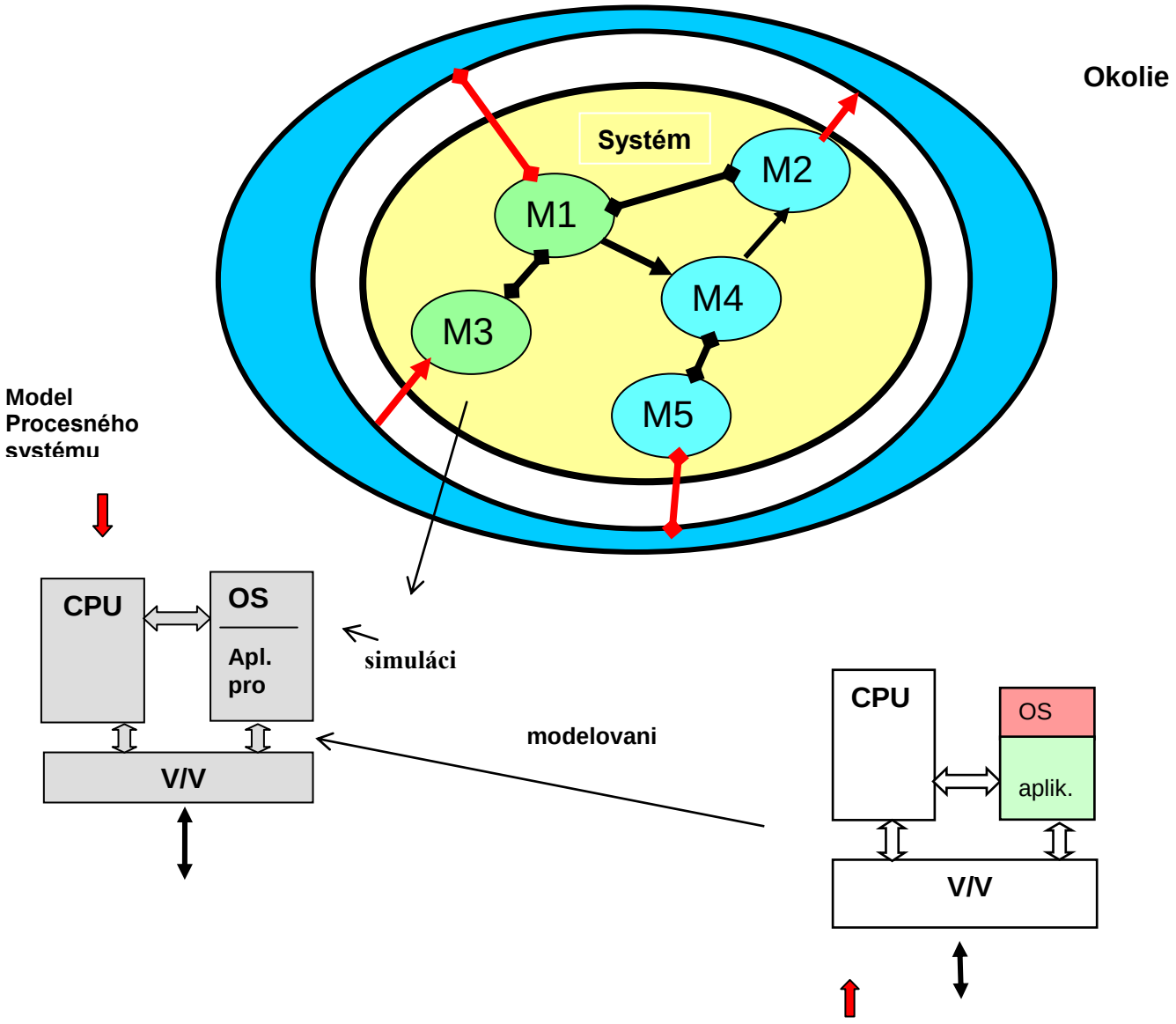


Hlavné úsilie v danom predmete je venované HW **RT úrovni** a prechodu s funkčných modulov systémovej úrovne k implementácií **HW spôsobom na RT úrovni**. Zjemneniu do úrovne návrhu logických obvodov bol venovaný osobitný predmet Logické systémy, na ktorý tu nadväzujeme a nebudeme sa taktiež už podrobnejšie venovať.

Náčrt implementácie modulu M1, ktorý sa implementuje SW spôsobom je na nasledujúcom obrázku

Princíp transferu (mapovania) systémových modulov do HW procesnej platformy

Moduly M1 a M3 a interkomunikácia s ich okolím sa má implementovať **SW spôsobom**



Vloženie OS a SP; mapovanie AP na realizovaný procesný systém pod OS s vyžadovaným rozhraním

Realizovaný procesný systém