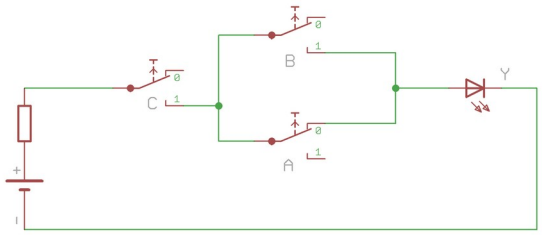
Úvod (opakování)

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

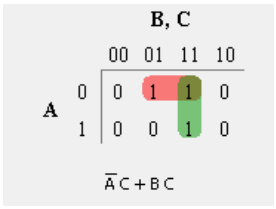
Kombinační logické (digitální) obvody

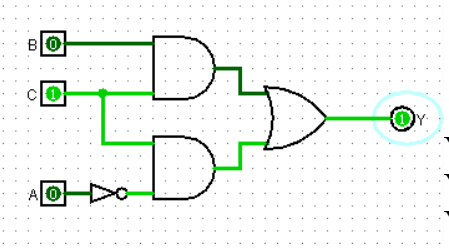
- 0/1
- výstup je fcí vstupu, tj. bez paměti (vs. sekvenční obvody – s pamětí)
- závislost vstupu a výstupu lze popsat
- - výraz (logická funkce)
- - tabulka
- - Karnoughova mapa
- - log.obvod
- - (schéma, diagram, program (Verilog, VHDL))

Příklad1 (vše popisuje jednu fci) – je vzájemně převoditelné

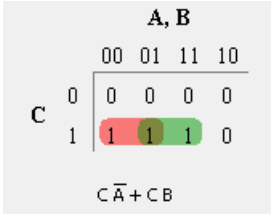


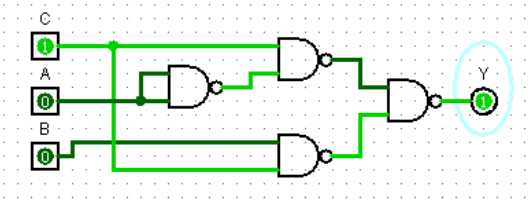
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

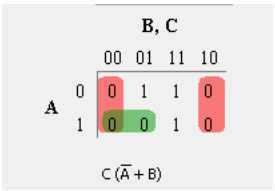


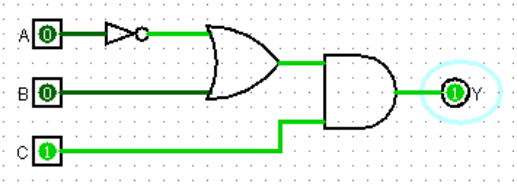


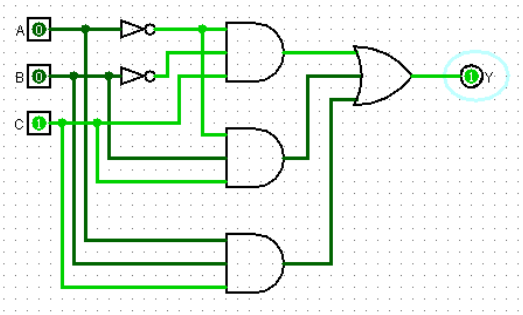
Y = B.C + ~A.C
Y = ~((~(B.C)) . (~(~A.C)))
Y = ~A.~B.C+~A.B.C+A.B.C
Y = C.(B+~A)











Booleovská algebra

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

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

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

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

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

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

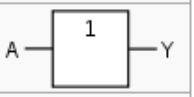
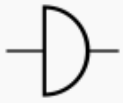
komplementarita: x + ~x = 1, x.~x = 0

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

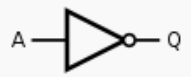
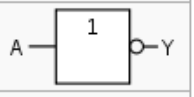
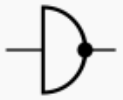
- asociativita: x + (y+z) = (x + y) + z, x.(y.z) = (x.y).z
- absorpce: x + (*x.y) = x, x.(x+y) = x*
- agresivita nuly,jedničky: x.0 = 0, x+1 = 1
- idempotence: x+x = x, x.x = x
- absorce negace: x+(~x.y) = x+y, x.(~x+y) = x.y
- dvojitá negace: ~(~x) = x
- komplementarita 0/1: ~0=1, ~1=0
- DeMorganovy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

Základní logické funkce (hradla)

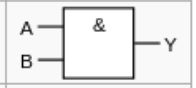
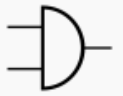
Opakovač, Repeater

Funkce	$\mathbf{Y = A}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		$X(A)$ | Y 0 | 0 1 | 1
IEC		
DIN		

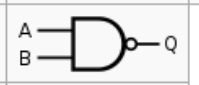
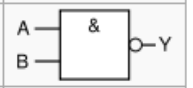
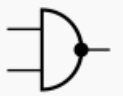
NOT, Invertor

Funkce	$\mathbf{Y = \overline{A}}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		$X(A)$ | Y 0 | 1 1 | 0
IEC		
DIN		

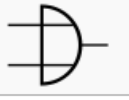
AND, log.soucin, konjunkce

Funkce	$\mathbf{Y = A \cdot B}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		$X_1(A)$ | $X_2(B)$ | Y 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
IEC		
DIN		

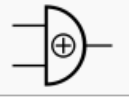
NAND, negovaný log.součin

Funkce	$\mathbf{Y = \overline{A \cdot B} = \overline{A} + \overline{B}}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		$X_1(A)$ | $X_2(B)$ | Y 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
IEC		
DIN		

OR, log.součet, disjukce

Funkce	$\mathbf{Y = A + B}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		$X_1(A)$ | $X_2(B)$ | Y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
IEC		
DIN		

XOR, exkluzivní log.součet

Funkce	$\mathbf{Y = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		$X_1(A)$ | $X_2(B)$ | Y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
IEC		
DIN		

Poznámky k návrhu

schéma, rovnice - plně definované (bez nedefinovaných stavů), dosazením hodnot možno vytvořit tabulku a mapu. (tj. problém pro zpětnou analýzu či redesign – nemusí být optimální)

tabulka, karnaughova mapa – může obsahovat nedefinované hodnoty (prázdné údaje) – ty můžeme dodefinovat abychom dosáhli výslednou co nejjednodušší implementaci (alternativně tím zjistíme že úloha není plně definovaná). Z řádků tabulky lze přímo psát rovnici ve tvaru součet součinů, ale není optimalizovaná.

zjednoduššování, optimalizace – úpravou rovnice (booleovská algebra), minimalizací v karnaoughově mapě

návrhová pravidla mají přesah i do SW – kontrola úplnosti/korektnosti zadání, zjednodušování či návrh podmínkových pravidel (IF ...)

Příklad 1 (pokračování) – minimalizace, nedefinované stavy

Zadání – jsou-li vsechny spínace A,B,C seplé(=1) má dioda svítít. Je li B a C rozepnuté (=0) nemá svítít. Je li B a C seplé má svítit. Je li A zaplé a B rozeplé nemá svítít. Je li A a C zaplé a B rozeplé nemá svítit.

Hloupý programátor / návrhář	Běžný programátor	Všímavý programátor	Programátor znající minimalizaci

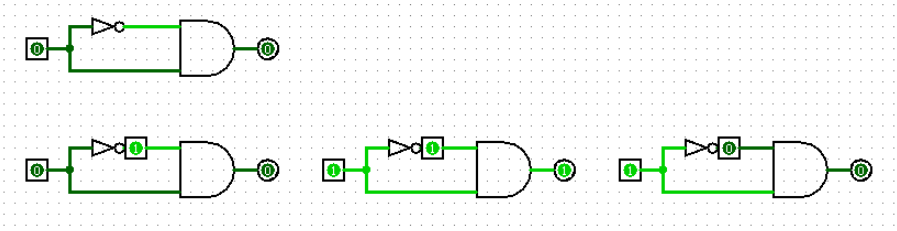
Boolean y;
if ((a==1)&(b==1)&(c==1)) y=1;
if ((b==0)&(c==0)) y=0;
if ((b==1)&(c==1)) y=1;
if ((a==1)&(b==0)) y=0;
if ((a==1)&(b==0)&(c==1)) y=0;
	Boolean y=0;
	if ((a==1)&(b==1)&(c==1)) y=1;
	if ((b==1)&(c==1)) y=1;
		Boolean y=0;
		if ((b==1)&(c==1)) y=1;
			Boolean y=b;

a	b	c	y
0	0	0	0
0	0	1	x
0	1	0	x
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	x
1	1	1	1

a \ b, c	00	01	11	10
0	0	x	1	x
1	0	0	1	x

Příklad 2 – Hazard

Reálné obvody – konečná doba šíření signálu, zpoždění na hradlech



Příklad: Y = ~A&A = 0

Ale: (reálné) hradlo negace má (také) zpoždění a při změne vstupu A $0 \rightarrow 1$ (viz prostřední obr kdy signál ještě neprošel hradlem) bude výstupní hodnota $0 \rightarrow 1 \rightarrow 0$. Tj. Ustálený stav OK, ale přechodně je tam nesprávná hodnota (=hazard)

Sekvenční logické obvody

Ke kombinační části se přidává paměťová, tj. Výstup je fcí vstupů + vnitřních proměných (minulost). tj. stejné vstupy nemusejí vyvolávat stejné vystupy, ale jsou závislé na vnitřním stavu (paměti)

Sekvenční obvody dělíme na

- **asynchronní** – změna vstupů se ihned promítne do stavu sekvenčního obvodu – (+) rychlost (-) složitost návrhu (hazardy)

- **synchronní** – zde je zaveden řídicí synchronizační signál (hodinový signál, hodiny). Změna vstupní proměnné se promítne do stavu sekvenčního obvodu až při příchodu hodinového signálu.

Dle reakce na hodinový signál můžeme dále rozlišit

- - obvod řízený hladinou hodinového pulsu (0, 1)

- - obvod řízený hranou hodinového pulsu (vzestupnou 0->1, sestupnou 1->0, oběma)

(konec opakování)

PLD (Programmable logic device)

Programovatelný log.obvod – typ integrovaného obvodu pro implementaci logického(digitálního) obvodu který může být konfigurován uživatelem k realizaci různých designů.

Výhody – flexibilní změna designu, redukce času vývoje, vysoká integrace, s nížení rizika fatálních chyb návrhu, výšší znovupoužitelnost a spolehlivost, zmenšení obvodu, nízké vstupní náklady

Logické obvody - rozdělení

- Standardní logika (=hradla, klopné obvody, čítače, multiplexery, …) (1-10^2 gates)

- ASIC (Applicatio Specivic IC):

- - Gate array (Hradlová pole) (10^2-10^5 gates))

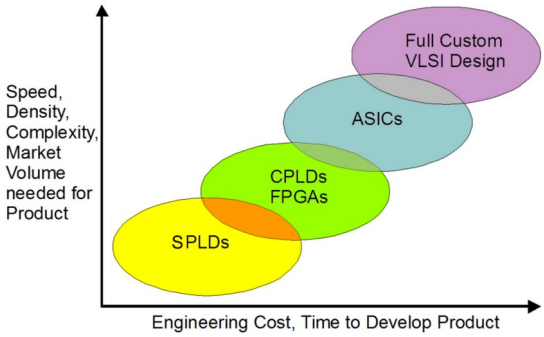
  prefabrikované IC, modifikace jen propojovací vrstvy

- - Cell-based IC (10^4-10^6 gates)

  prefabrikované bloky umisťované dle požadavků

- - Custom IC (10^5-10^7)

  uživatelsky definované všechny vrstvy



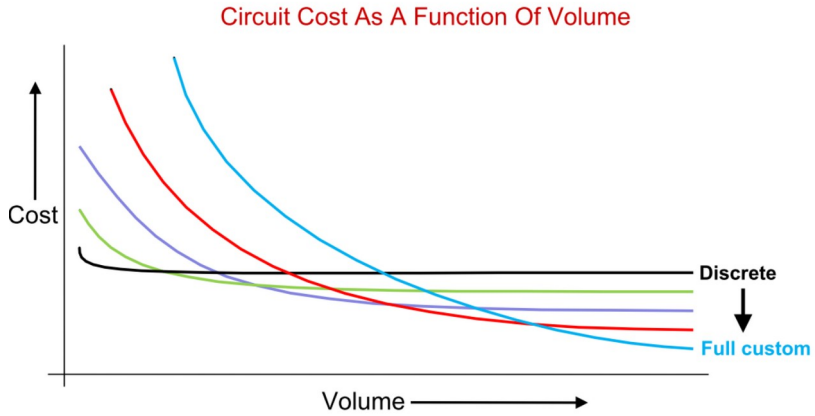
- PLD - Programovatelná logika (patří do ASIC)

- - SPLD (simple programmable logic device) – PROM, PLA, PAL, GAL, .. (10-10^3 gates)

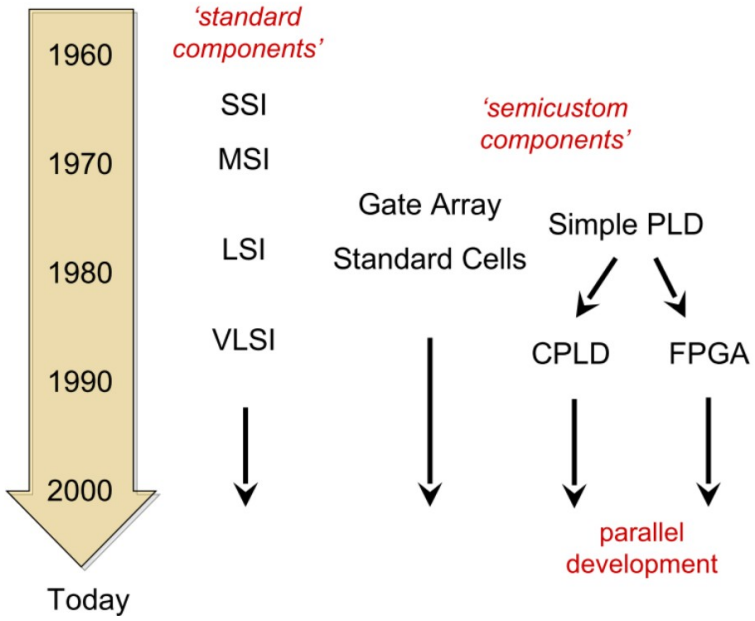
- - CPLD (complex PLD) (10^2-10^4 gates)

- - FPGA (Field programmable gate array) – (10^3-10^6 gates)

Počet „ekvivateltních hradel“ - míra ohodnocení složitosti PLD (kolik dvouvstupovýh NAND hradel může daný obvod nahradit). Nelze brát absolutně, je závislé na typu PLD a aplikaci



PLD – vývoj

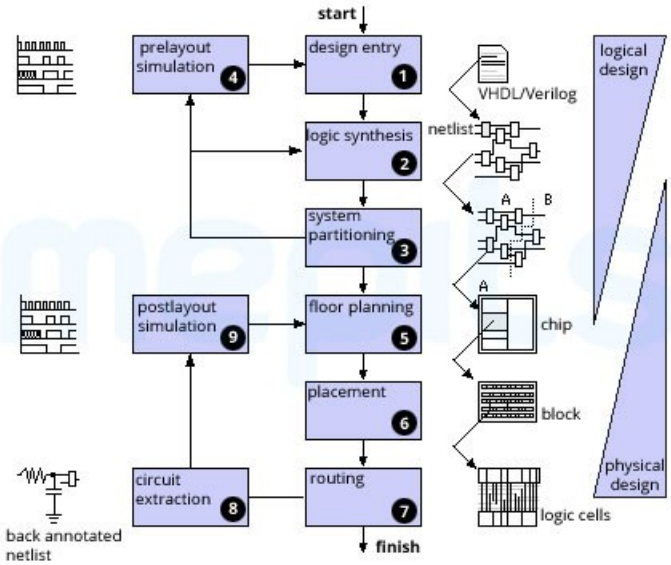


SPLD – pevné vnitřní propojení → determinitické zpoždění (propagation delay), velikost <1000 ekv.hradel

CPLD – více SPLD na čipu s programatelnou (interní) propojovací maticí

FPGA – pole (2D) logických bloků, s programovatelnou propojovacá maticí, spec.funčkní bloky

ASIC design



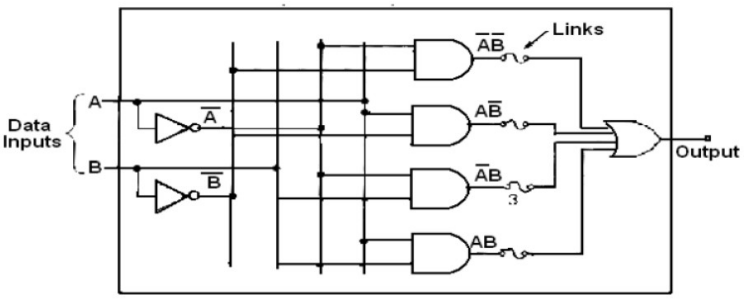
- design entry: HDL,text, scheme, C, ...
- synthesis: map design to cells, create netlist
- partitioning: large design into smaller part
- pre-simulation: logic (functional) simul
- floorplanning: Chip struct, blocks
- placement: cells in block
- routing: block interconnection
- extraction: routed chip to electrical scheme
- post-simulation: check final layout

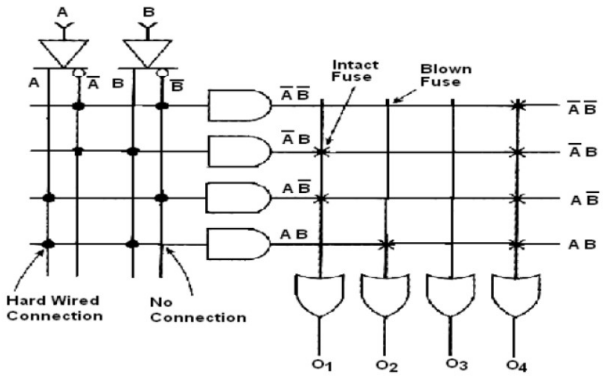
T.Mainzer, ACS2

SPLD

Myšlenka (S)PLD je založena na dvouúrovňové log. fci AND – OR.

Vlevo zapojnení, vpravo zjednodušený nákres.





Dle typu propojovací matice (fixní vs. programovatelná) lze rozlišit tyto typy obvodů:

Typ	AND matice	OR matice
ROM	Fixed (=adress decoder) (=full)	Programmable (data)
PLA	Programmable	Programmable
PAL	Programmable	Fixed

ROM

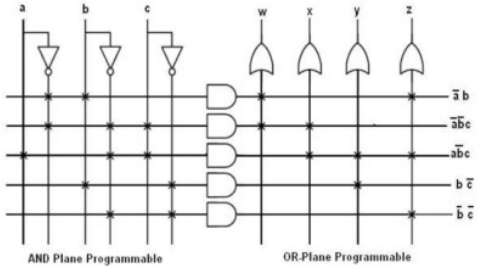
ROM (resp. obecněji jakákoliv pamět) může sloužit jako náhrada kombinačního obvodu. Vstupy = adresa paměti (adresová sběrnice), výstupy = data (datová sběrnice)

Paměťový dekóder dekóduje všechny vstupní termy (všechny AND kombinace vstupů).

PLA

Obě matice jsou programovatelné. AND matice nepokrývá všechny vstupní termy.

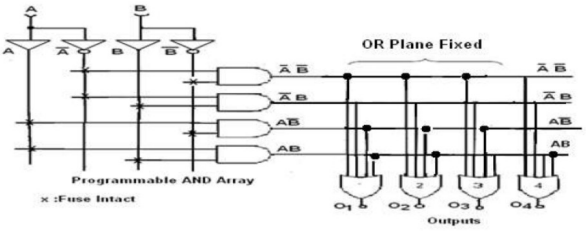
Typické PLA bylo 16x48x8 = 16 vstupů, 48 AND termů, 8 výstupů.



PAL

PAL (na rozdíl od PLA) má programovatelnou jen AND matici, OR je fixní – levnější, ale méně flexibilní.

Nověji býval na výstupu kl.obvod s připadnými dalšími fcemi či zpětnou vazbou do OR pole



GAL

Elektricky (re)programovatelný (vylepšený) PAL.

T.Mainzer, ACS2

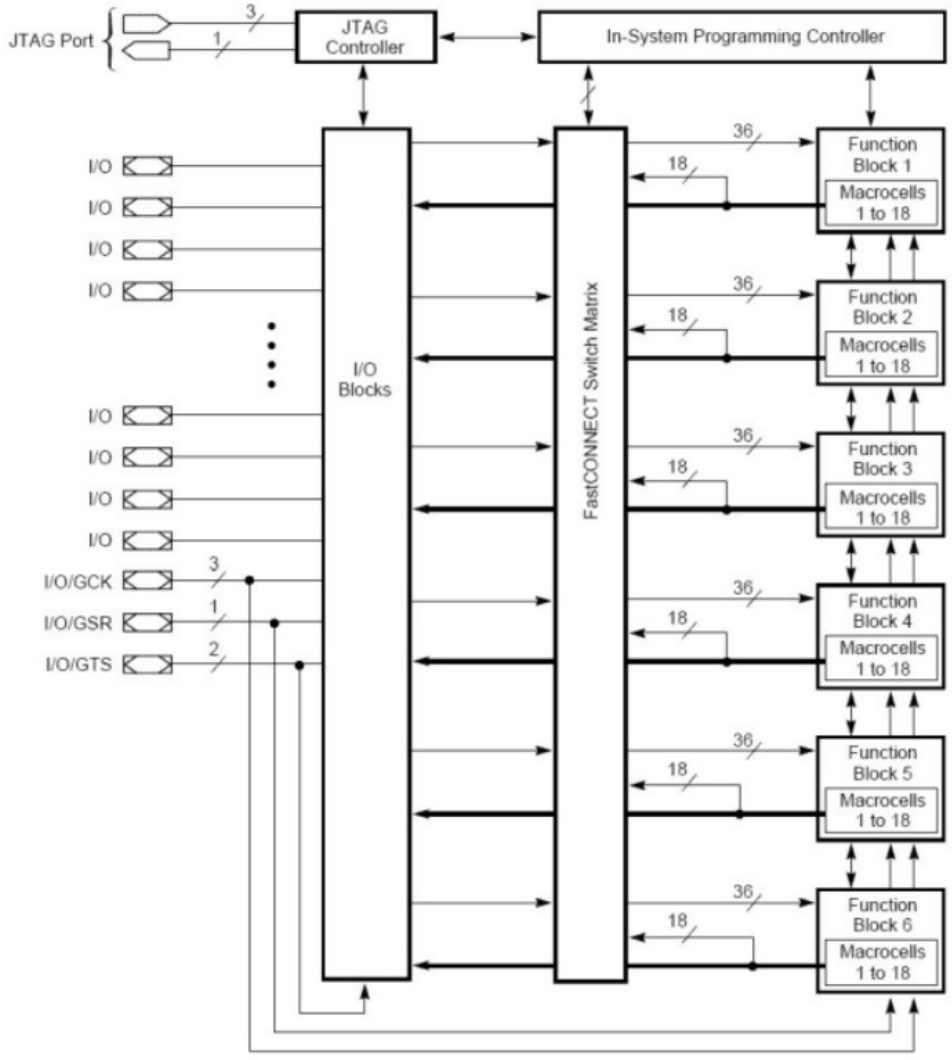
CPLD

CPLD (Complex PLD) se skládá z většího počtu logických nebo funkčních bloků propojených konfigurovatelnou prop.maticí. Pro každý výstup je zde konfigurovatelný I/O blok.

Proč takto? Proč jen nerozšířit myšlenku PLA/PAL na větší šířku?

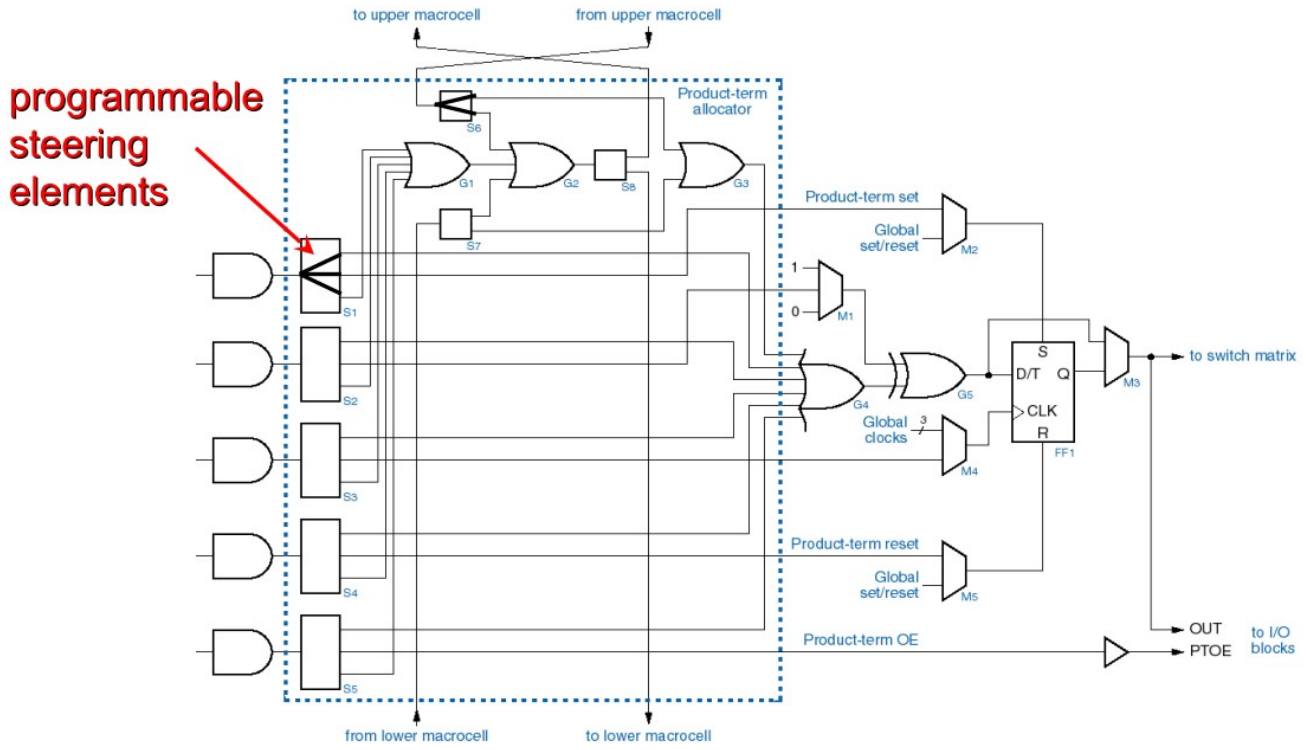
Velký počet vstupů vede na (kvadraticky) větší počet termů – zvětšuje se plocha čipu. Také hradla (AND/OR) jsou pak vícevstupová (výrobní omezení pak zamená nutnost je rozdělit do více úrovní což vede ke zpomalení)

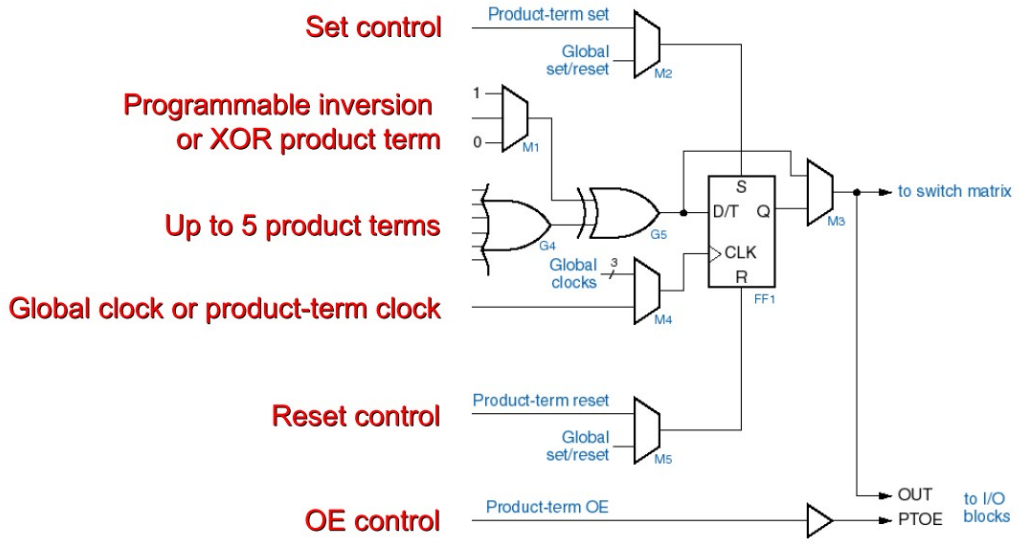
Příklad interní struktury CPLD XC95108.



Makrocell (18 makrocells na funkční blok):

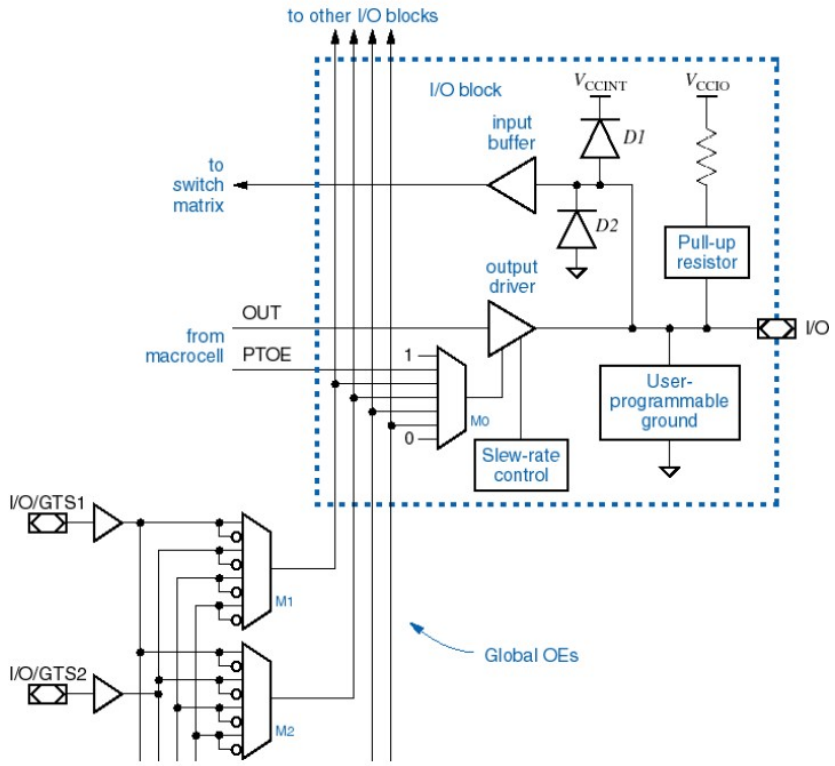
(AND-OR termy, vazby z vyšší/nižší macrocell, možnost negace výstupu, D-klopný obvod + Set/Reset, ...)

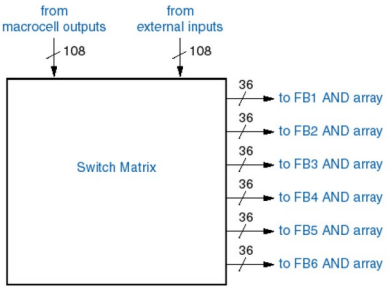




T.Mainzer, ACS2

I/O block (u každého pinu) – Vstup, výstup, Z-stav (řízený signál OE), volitelný pull-up





Switch matrix

Obceně cokoliv - od omezené množiny multiplexerů (menší rychlejší, omezená množina spojů, obtížnější a omezující návrh) k plnému crossbaru (snadný návrh, veliké, pomalější)

CPLD – závěr

- desítky až (malé) stovky klopných obvodů (makrocells), desítky I/O, rychlosti v radech 10ns

- naprogramovaní obvykle non-volatile technologií (na rozdíl od FPGA)

- pin locking (při návrhu lze napevno nastavit I/O což může omezovat výslednou rychlost nebo mapování prostředků. Bez pevného nastavení je možné že (i malá) změna návrhu povede k jinému alokování I/O)

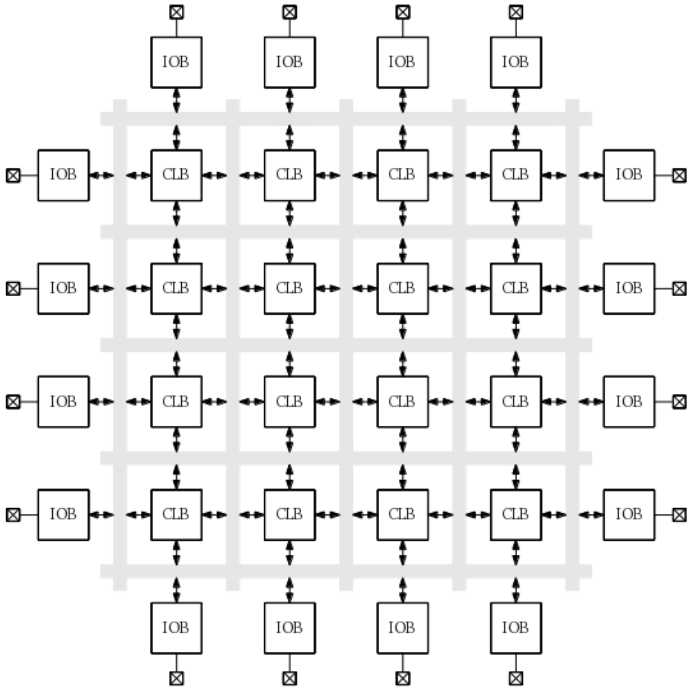
- větší obvody jsou (více než lineárně) dražší

- pokud návrh se nevejde do jednoho IC lze jej jen velmi obtížně rozdělit (vnitřní propojovací možnosti jsou větší než vnější)

- výrobci: Xilinx (XC9500, Coolrunner), Altera (MAX), Lattice, Cypress, Atmel

FPGA

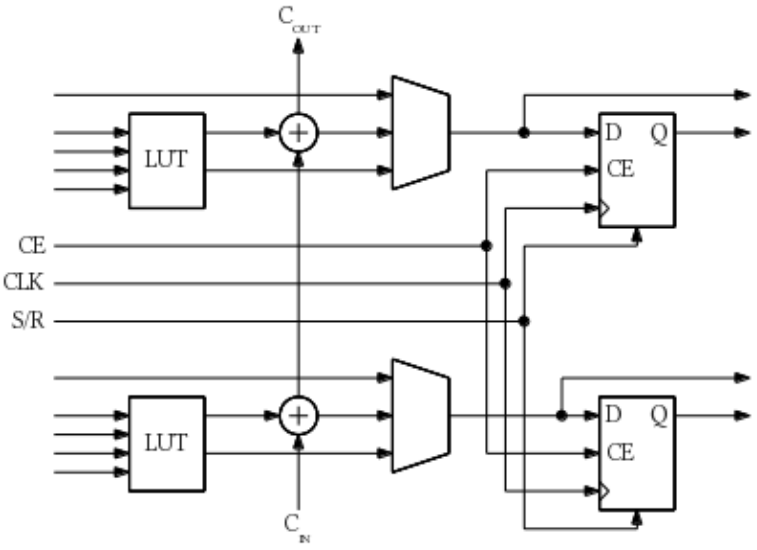
Programovatelná hradlová pole jsou tvořena velkým množstvím menších konfigurovatelných bloků obecné logiky, které jsou navzájem propojitelné propojovací maticí. Komunikaci s okolním prostředím zajišťují vstupně-výstupní bloky. Kromě toho mohou FPGA ještě obsahovat další speciální funkční bloky - typicky statických/dynamických pamětí RAM, nebo správy hodinových signálů, speciální vstupně-výstupní bloky, ale třeba i procesorové jádro.



Základní log.bloky

Základním stav,.prvkem je realtivně jednoduchý blok log.fce (CLB = configurable log.block (Xilinx), LAB = logic array block (Altera))

Typicky se skládá z:



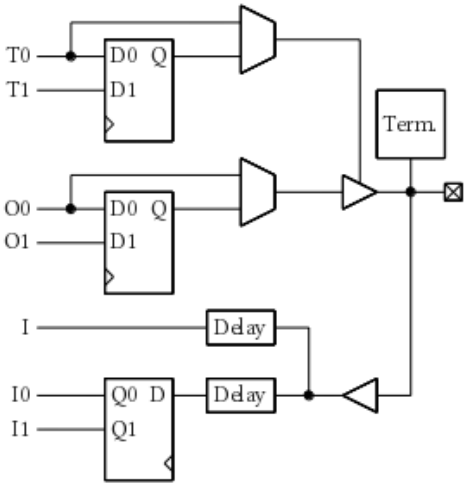
- LUT – programovatelné logická tabulka (realizovaná jako look-up table) - Realizuje tedy binární fci několika (typicky čtyř, pěti nebo šesti) vstupů. Kromě této základní funkce umožňují LUT obvykle realizaci i jiných typů funkčních bloků jako jsou malé paměti RAM a ROM, nebo posuvné registry

- signály pro (rychlé) šíření přenosu mezi CLB

- mutiplexery pro výber fce a spojování více LUT do větších celků (toto není na obr. nakresleno)

- registry pro relizaci sekv.log.fcí.

I/O block



Vstupně/výstupní blok (IOB) zprostředkovává spojení FPGA s dalšími obvody. Každý pin má IOB blok.

Vstupně-výstupní blok obsahuje vstupní a výstupní registry, budiče a přijímače, zpožďovací linky, obvody impedančního přizpůsobení a ochranné obvody. Vstupní i výstupní registry jsou u všech dnešních FPGA realizovány pomocí kombinace dvou registrů tak, aby umožňovaly vstup a výstup DDR (double data rate) signálů. Samozřejmě mohou být nakonfigurovány do běžného režimu SDR. Dále jsou IOB bloky často vybaveny podporou pro vstup/výstup diferenciálních signálů.

Výstup signálu je realizován (volitelným) výstupním registrem s třístavovým budičem. Výstup budiče je přímo připojen na pin FPGA.

Vstupní externí signál je přiveden na konfigurovatelný přijímač, resp. zde mohou být různé přijímače pro jednotlivé podporované I/O standardy následované multiplexerem. Výstup přijímače může být přes konfigurovatelnou zpožďovací linku přiveden přímo na propojovací matici FPGA nebo na DDR/SDR vstupní registr a z něho teprve na propojovací matici. Pro vstup diferenciálních signálů obsahuje přijímač ještě druhý vstup připojený do sousedního I/O bloku. Programovatelné zpožďovací linky umožňují realizaci časového posunu vstupního signálu. Tím je možné například korigovat vzájemný fázový posun signálů, případně posun signálu vůči hodinám.

Ochranné obvody (nejsou nakresleny) - chrání logiku IOB před poškozením z vnějšího světa . Obvody **impedančního přizpůsobení** umožňují přizpůsobit vnitřní impedanci budiče v FPGA impedanci externího vedení nebo zakončit externí vedení správně přizpůsobenou impedancí. Novější FPGA podporují dynamické řízení zakončovací impedance, což je důležité například pro připojení různých variant DDR a QDR rozhraní nebo třeba PCIe.

Propojovací „matice“

Globální propojení - umožňují vzájemné propojení libovolných funkčních bloků FPGA. Jsou realizovány jako několik vrstev různě organizovaných spojů s propojovacími maticemi pro jednotlivé funkční bloky. Často bývají spoje organizovány jako sada horizontálních a vertikálních vodičů různých délek. Programovatelné propojovací matice umožňují připojení vstupů a výstupů funkčních bloků k jednotlivým propojovacím vodičům. Omezená rychlost (omezeno délkou, počtem programovatelných spínačů)

Lokální propojovací prostředky umožňují propojení pouze sousedních funkčních bloků. Jedná se například o propojení řetězců šíření přenosu logických řezů, propojení diferenciálních signálů vstupně-výstupních bloků, ale i krátké rychlé spoje mezi sousedními konfigurovatelnými logickými bloky.

T.Mainzer, ACS2

Lokální spoje jsou mnohem kratší, s méně spínači a nižší kapacitou než spoje globální. Lokální spoje tedy způsobují mnohem nižší zpoždění signálů než globální spoje.

Speciální propojovací prostředky slouží pro spojení mezi vyhrazenými vstup a výstupy funkčních bloků. Jedná se zejména o spoje určené k šíření hodinových signálů a dalších globálních signálů typu reset, tristate apod. Tyto speciální propojovací prostředky jsou optimalizovány tak, aby způsobovaly co nejmenší zpoždění procházejících signálů. Je jich omezené množtví. Mohou být globální i lokální.

Speciální funkční bloky

Dnešní programovatelná hradlová pole obsahují další, často jednoúčelové, funkční bloky usnadňující použití FPGA. Typickými zástupci této kategorie jsou blokové paměti, násobičky a jiné aritmetické obvody, obvody pro správu a generování hodinových signálů, sériové transceivery a další.

Blokové paměti jsou obvykle dvouportové statické paměti RAM o kapacitě jednotek až desítek kilobytů a s konfigurovatelnou šířkou adresových a datových sběrnic. Může být použit jako jednoportová i dvouportová paměť typu RAM i ROM. Datové sběrnice mohou být obvykle nastaveny na šířku 1, 2, 4, 8/9, 16/18, 32/36, nebo 64/72 bitů (bez parity/s paritou). Dvouportové blokové paměti usnadňují realizaci různých typů speciálních pamětí jako jsou FIFO, kruhové buffery, CAM (content associative memory) apod.

Aritmetické obvody jsou reprezentovány obvykle celočíselnými násobičkami, případně složitějšími DSP bloky umožňujícími realizaci náročnějších matematických operací v jednom hodinovém cyklu. Typickým příkladem jednodušších, ale nejčastějších operací realizovaných pomocí DSP bloků je MAC (Multiply-and-Accummulate), což je základní operace číslicové filtrace a téměř všech DSP algoritmů.

Bloky pro správu a generování hodinových signálů jsou založeny na PLL (phase-locked loop — fázový závěs) nebo DLL (delay-locked loop). Složitější hodinové bloky ve větších FPGA často obsahují i několik PLL a DLL. Tyto bloky nabízejí funkce jako je syntéza a dělení hodinových signálů, generování fázově posunutých hodin, kompenzace vnitřních i vnějších zpoždění a fázových posunů a podobně. Vstupy a výstupy těchto bloků jsou obvykle připojený přímo ke speciálním spojům určeným pro šíření hodinových signálů. Na rozdíl od předchozích dvou typů speciálních funkčních bloků nejdou hodinové obvody jednoduše nahradit běžnou logikou FPGA.

Sériové transceivery umožňují poměrně jednoduchou realizaci velmi rychlých sériových rozhraní. Blok transceiveru obvykle obsahuje diferenciální budič a přijímač, serializer a deserializer, kodér a dekodér, extraktor hodinového signálu a další pomocné obvody. Pomocí transceiverů je možné sériově komunikovat s okolím rychlostmi v řádech jednotek až desítek gigabitů za sekundu. Interní rozhraní v FPGA je pak realizováno synchronními paralelními daty s mnohem nižší hodinovou frekvencí. Sériové transceivery jsou nezbytné pro realizaci mnoha standardních sériových rozhraní jako jsou například PCIe, XAUI pro 10Gbps Ethernet a mnohá další.

Z **ostatních funkčních bloků**, které se vyskytují v FPGA si zaslouží zmínku řadiče pamětí a mikroprocesory. Některá hradlová pole obsahují hotový řadič dynamických pamětí, který obvykle

podporuje paměti typu SDR, DDR, DDR2 a DDR3 SDRAM. Hotový blok řadiče uspoří mnoho logických prostředků a často významně zvýší datovou propustnost externí dynamické paměti oproti realizaci řadiče pomocí běžné logiky. Některé (spíše starší) high-end FPGA obsahovaly mikroprocesor PowerPC řady 4xx nebo ARM s instrukční sadou verze 2. Nověji se v FPFA objevují vestavěné vícejádrové procesory ARM Cortex(-A9). Tvoří tak zajímavou alternativu běžným SoC svou programovatelností, ale i FPGA díky mnohem výkonnějšímu CPU než běžné soft-core procesory.

FPGA – závěr+++

- stovky až statisíce LUTS, až 10Mbity RAM, desítky az stovky I/O, rychlosti v radech ns
- specializované periférie (paměti, řadiče DRAM, procesory, MUL+ACC, AES, DSP fce, ..)
- podpora různých typů vstupů/výstupů, včetně diferenciálního a DDR (double data rate)
- naprogramovaní obvykle SRAM technologií – po startu je třeba „zavést program“

  (někteří výrobci i non-volatile - flash nebo antifuse technolgie)

  (program lze zavést z nadřazeného systému a nebo z konfigurační flash)
- realativně dobrá škálovatelnost
- (specifické možnosti: HW&SW codesign, částečná dynamická rekonfigurovatelnost)
- výrobci: Xilinx (Virtex, Spartan), Altera (Stratix, Cyclone), Lattice

PLD - programování

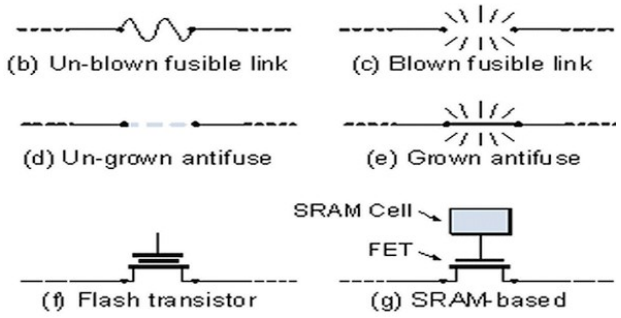
OTP – one time programmable – fuses, antifuses (hi resistance → low resistance), PROM

EPROM - erasable ROM (erase by UV light)

EEPROM – electric erasable ROM (programovací napětí)

FLASH

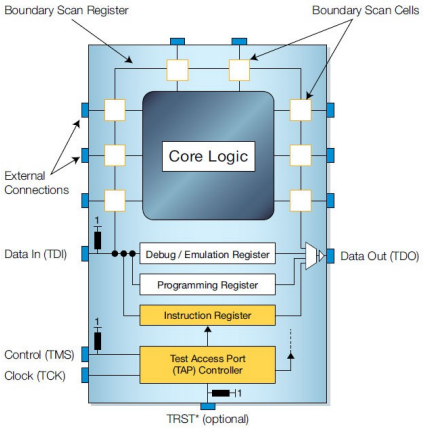
SRAM



volatile / non-volatile

ISP - in-system programming

JTAG (PLD testování)



JTAG (joint test action group) je standard definovaný normou IEE 1491. původně zejména pro testování pcb. Nyní může sloužit pro

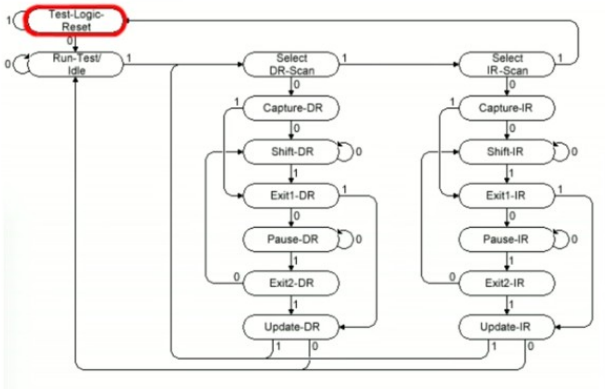
- testování pcb a IC (boundary scan test)

- pro programovaní log.obvodů (in-system programming)

- pro debugování (SW debug and emulation).

Standard definuje signály a jejich význam, nedefinuje fyzické rozhraní (konektor)

Signály rozhraní (TAP = test access port):



TDI (Test Data In)

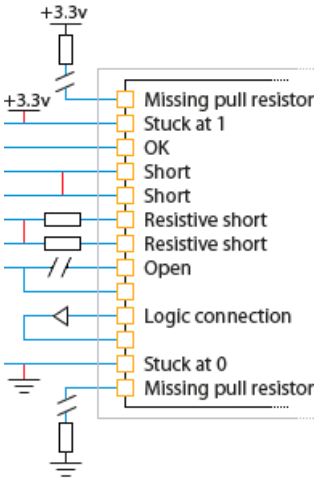
TDO (Test Data Out)

TCK (Test ClocK)

TMS (Test Mode Select)

TRST (Test ReSeT) - volitelný

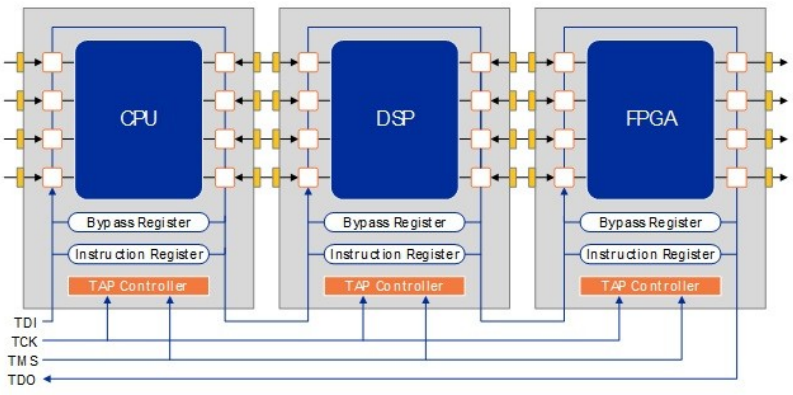
Data jsou přenášena sériově (TCK = hodiny, TDI/TDO = data), signal TMS definuje chovani JTAGu (přechod mezi stavy - viz stavový diagram. Signal Reset je volitelný neboť reset stavu lze dosáhnout pomocí TMS (5*1))



Ve stavu Shift-IR respp. Shift-DR je přes TDI/TDO přenášen reg.IR (instruction reg.) resp. DR (data register) – jenž definují další fci JTAG (pozor – není standarizována, je závislé na výrobci). Standardizovany jsou jen instrukce pro boundary scan, zjištění id, a bypass.

Boundary scan umožnuje zapisovat a číst I/O. BSDL = boundary scan description language je podmnožinou VHDL. BSR = boundary scan register. Boudary scan registry mohou být vnitřní/vnější.

JTAG je koncipován jako daisy-chained takže JTAG interface jednotlivých čipů mohou (ale nemusí) být (sériově) spojeny (tj. Každý obvod na desce nemusí mít vyveden svůj konektor ale všechny je možno testovat pomocí jednoho). Pozn – i jeden IC se může z hlediska JTAGu tvářit jako několik – typické pro SoC)



! JTAG může být bezpečnostní dírou do systému

T.Mainzer, ACS2