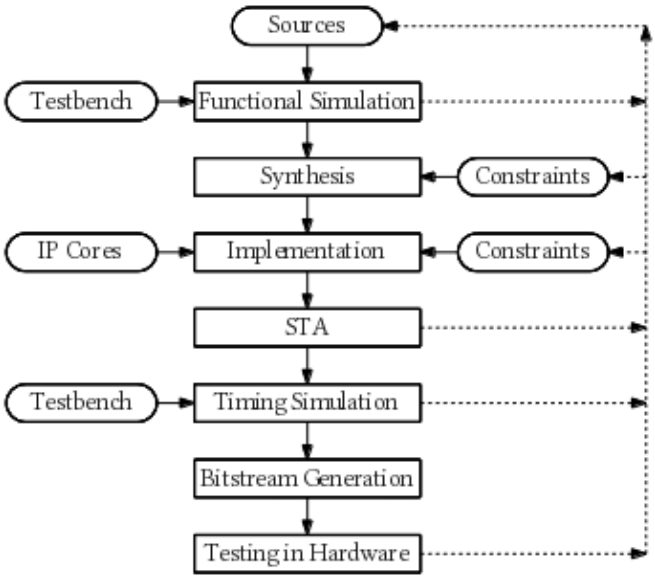
Návrhování pro PLD

Jednoduché PLD – při znalosti vnitřní struktury je možno obvod přímo navrhovat/mapovat.

(PAL), CPLD a FPGA – i při detailní znalosti vnitřní je manuální návrh obtížný (různá omezení, optimalizace) a je nutno použít specializované vývojové nástroje které nám návrh ulehčí/umožní.

Vývojové prostředky pro programovatelnou logiku musí zajistit transformaci vstupního návrhu nebo popisu funkce do výstupního formátu použitelného pro konfiguraci nebo naprogramování konkrétní součástky

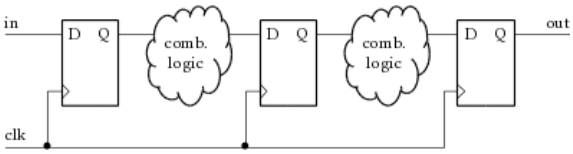
Popis log.obvodu - výraz (logická funkce) , tabulka, Karnoughova mapa (K-mapa), log.obvod (schéma), (stavový) diagram, HDL jazyk (Hardware description language – VHDL, Verilog, system verilog) , HLL (high level language – C/C++, matlab, ..)



Fce návrhového systému:

- Návrh (často možno kombinovat různé typy popisu)

(různé formy popisu se převádějí na RTL – register transfer level(logic)). Už na úrovni návrhu je často nutné/vhodné přizpůsobit návrh možnostem cílového obvodu (např. použití speciálních funkcních bloků – buď vložením a nebo „správným“ vysokoúrovňovým popisem, dále je zde také omezení vstupů/výtupů a mapování pinů (napěťové úrovně, I/O banky, spec.typy I/O – global clock, DDR,… . Omezení propojovací matice - globální rychlé spoje, lokální rychlé spoje, běžné spoje.)



- Verifikace (funkční simulace – testbench – testovací vektory a vyhodnocení výsledků)

- Syntéza (převádí RTL popis na funkční bloky cílové technologie), přicemž bere do úvahy různá omezení (constrains) – typy I/O, přiřazení pinů, omezení zdrojů, povolená časová zpoždění atp)

- Implementace (mapování sytentizovaného popisu na zařízení, přiřazení bloků, propojovacích matic atp – bere v úvahu návrhové omezení – generuje NetList (konkretni popis co vede kam) a SDF (standard delay format) – definuje zpoždění prvků). IP cores = předpřipravené (kofigurovatelné, parametrizovatelné) návrhové bloky „třetích stran“ jenž můžeme zahrnout do návrhu)

- STA (static time analysis) – statická časová analýza – ověřuje splnění časových constrainů (timing constrains) obvodu – frekv.hodinových signálů (rychlost šíření), časovnání I/O, vztahy mezi hodinovými doménami, omezení vstupních časových signálů (náběh, hold)… - ověřuje že navržené synchronní elementy splňují dané constrainy. Toto analýzou je také určena max.možná frekv.hodinových signálů.

- Simulace (časová) a testování – dynamické analýza(simulace) log.obvodu – pro plně synchronní návrh je správnost zaručena STA (za předpokladu správnosti návrhu), ale můžeme se podívat na zpoždění signálů v jednotlivých částech návrhu (např. kde je třeba optimalizovat) atp.

- Generování bitstreamu pro programování

- Programování – programování bitstreamu do zařízení

- Verifikace v zařízení – ověření vazby na okolní systém. Specificky pro progr.logiku je možno přímo do návrhu zabudovat logiku (typicky IP jádro) pro sledování (interního) stavu obvodu – typicky přes JTAG rozhraní umožňuje sledovat/nastavovat signály, hodnoty, stavy atp. (ala log.analyzátor)

Synchronní návrh

Obvody FPGA (CPLD) jsou navrženy pro efektivní implementaci plně synchronního návrhu. To znamená, že samotný návrh aplikace by měl dodržovat pravidla synchronního návrhu – viz též popis RTL (registr transfer).

Pro správný (jednoduchý) synchr.návrh je vhodné mít co nejnižší počet hodinových domén (ideálně jednu), zajistit případně správný přechod mezi těmito doménami a synchronizovat vnější vstupy do správných hodinových domén.

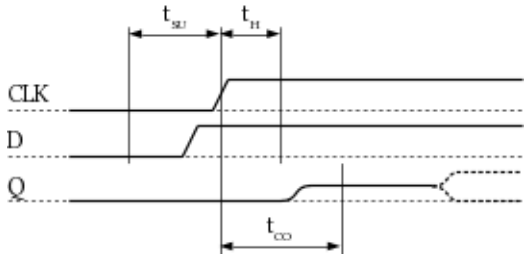
(hodinová doména (clock domain) – část obvodu, jejíž všechny synchronní sekvenční prvky používají jeden společný hodinový signál – typicky klopné obvody, ale (v FPGA) i např. synchronní blokové paměti, aritmetické bloky atp)

Asynchronní signály

Asynchronní signály na vstupu synchronního obvodu mohou obecně způsobit metastabilitu a nekoherenci.

Metastabilita klopného obvodu

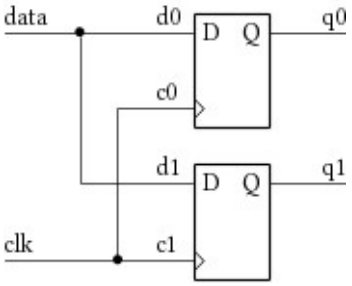
Metastabilita je dočasné uvedení klopného obvodu do nestabilního stavu. Tato dočasná nestabilita se může projevovat například oscilacemi na výstupu nebo častěji nedefinovanou výstupní logickou úrovní mezi logickou nulou a jedničkou. Vzniká při porušení specifikovaných časových parametrů na vstupu klopného obvodu. Pokud vstupní signál klopného obvodu mění úroveň v době kdy se mění signál hodin (v čase vymezeném časovými parametry Tsu (*setup time)* a Th (*hold time))*, existuje nenulová pravděpodobnost přivedení klopného obvodu do metastabilního stavu.



Vstupní signál *D* na obrázku mění úroveň příliš blízko aktivní hrany hodin *CLK* čímž porušuje t_{SU}. Důsledkem je metastabilní stav výstupu *Q* vyjádřený nedefinovanou úrovní mezi logickou nulou a jedničkou. Po době _delší_(!) než t_{CO} (clock to output) se výstup nakonec překlopí do některé definované logické úrovně. Může to být ale jak logická jednička, tak i nula.

Pro zamezení vzniku problémů způsobených metastabilitou je důležité vždy správně ošetřit vstupy asynchronních signálů tak, aby nedošlo k porušení podmínek daných t_{SU} a t_H nebo zajistit aby případná metastabilita neohrozila funkci obvodu

Nekoherence



Pokud je asynchronní signál připojen do vstupů několika synchronních prvků zároveň, velmi často nastává problém s nekoherencí jednotlivých signálových a hodinových cest.

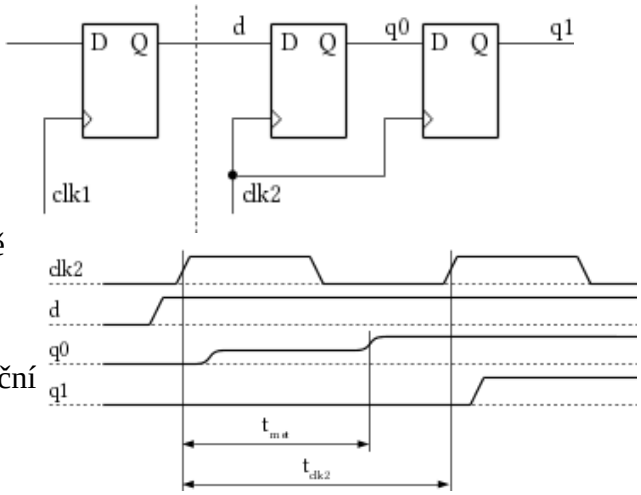
Obecně délka cest (datová i hodinová) ke klopným obvodům se může lišit a tedy i okamžik překlopení klopných obvodů. Tedy kvůli rozdílným zpožděním vstupních a hodinových signálů může dojít k nastavení výstupů (q0 q1) do rozdílných logických úrovní.

Řešením je opět nutnost vždy zajistit správnou synchronizaci asynchronních signálů.

Přechod mezi hodinovými doménami (crossing clock domains)

Správné ošetření asynchronních vstupů je nezbytné pro korektní funkci synchronního logického obvodu. (tj. externích asynchronních signálů i interních signálů v jiné hodinové doméně). Existuje několik různých metod synchronizace asynchronních signálů, přičemž každá je vhodná pro jiný typ signálu.

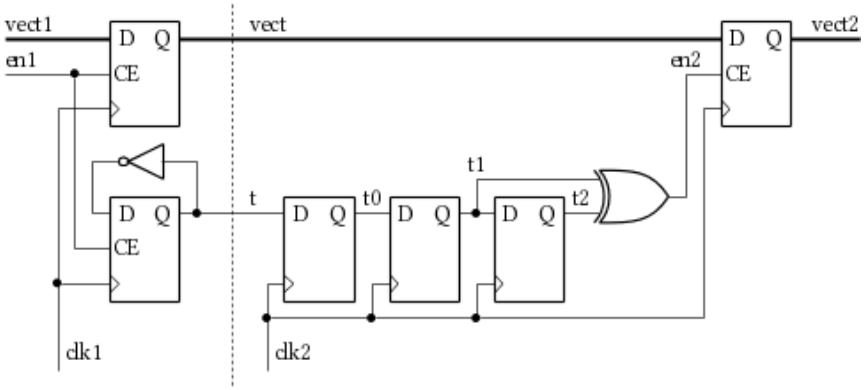
Jednotlivé signály



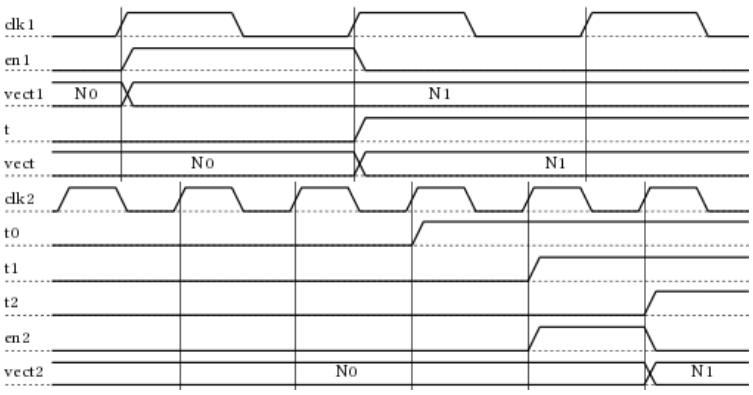
Nejjednodušším případem přechodu mezi hodinovými doménami je synchronizace samostatného signálu (d). Potlačení vlivu metastability na zbytek obvodu se dosáhne několikanásobným registrováním signálu v nové hodinové doméně podle obrázku . Pro synchronizaci musí být použity minimálně dva klopné obvody v hodinové doméně *clk2*. Přestože může být první klopný obvod uveden do metastabilního stavu, výstup druhého klopného obvodu se nachází již pouze ve stabilních stavech. Tento synchronizační obvod je funkční pokud platí, že $T_{clk2} - t_{SU} \geq t_{met.}$

Tato synchronizační metoda je obecně vhodná pouze pro jednotlivé signály. Její použití pro sběrnice je možné pouze ve speciálních případech kdy nové hodnoty v cílové hodinové doméně budou využita až dostatečný počet hodinových cyklů po změně, tak aby nová hodnota vektoru byla stabilní a konzistentní.

Sběrnice (pomalé)



Tato metoda je vhodná pro synchronizaci sběrnic v cílové hodinové doméně s vyšší hodinovou frekvencí než má zdrojová doména nebo pro synchronizaci sběrnic, které mění svou hodnotu pouze občas. Metoda je založena na využití samostatného pomocného signálu, který signalizuje změnu hodnoty (en1-en2) .

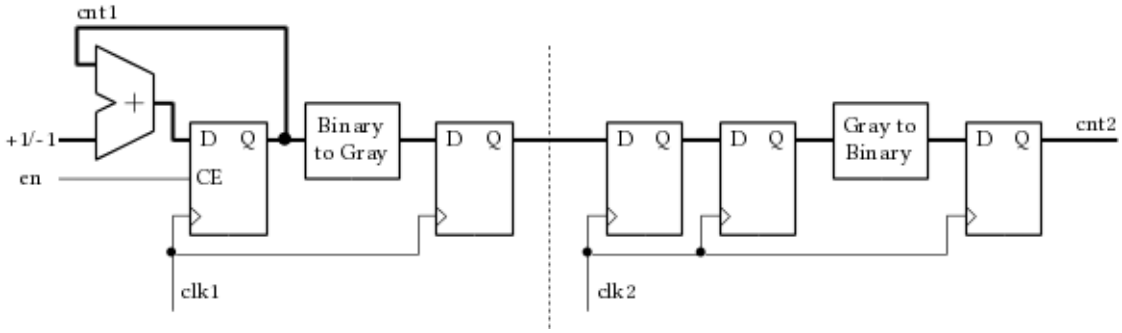


Tento způsob synchronizace je vhodný pouze pokud zůstává vektor *vect* konstantní po dostatečně dlouhou dobu vzhledem k periodě hodinového signálu *clk2*.

Praktická bezpečná hodnota doby, po kterou musejí zůstat hodnoty *vect* a *t* stabilní je typicky $\geq 3\ T_{clk2}$

Čítače

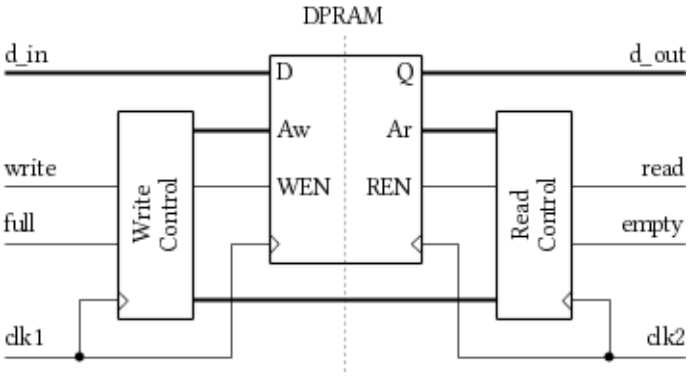
Často je třeba přenést mezi hodinovými doménami hodnotu čítače. Pokud je možné zajistit, že se mezi dvěma aktivními hranami cílového hodinového signálu (ckl2) změní maximálně jediný bit (tj. Čítač o +/- 1), je možné provést jednoduchou synchronizaci pomocí klopných obvodů. Čítače je možno realizovat buď přímo v Grayově kódu, nebo do Grayova kódu převést hodnotu binárního čítače.



Tento obvod synchronizuje hodnotu binárního čítače v hodinové doméně *clk1* do asynchronní hodinové domény *clk2*. V doméně clk2 je synchronizována vůči novému hodinovému signálu dvouúrovňovými registry stejně jako jednoduché signály. Vzhledem k tomu, že se během jedné periody mění hodnota pouze jediného bitu, nebude na výstupu z druhé úrovně registrů nikdy nekonzistentní hodnota.

Sběrnice (rychlé)

Předchozí metody nejdou přímo využít pro přenos hodnot rychle se měnících sběrnic mezi různými hodinovými doménami. Na jejich základě je ale možné k tomuto účelu vytvořit asynchronní FIFO. Zjednodušené blokové schéma asynchronní paměti FIFO viz obr. (asynchronní označuje vztah mezi hodinovými doménami *clk1* a *clk2*. Chování každé části v rámci vlastní hodinové domény je plně synchronní)



Asynchronní FIFO je tvořeno dvouportovou blokovou pamětí, řídící logikou zápisu a čtení. Vlastní FIFO není nutno vyvíjet od začátku. Mnoho současných FPGA umožňuje nakonfigurovat blokové paměti přímo do režimu FIFO a přímo je do návrhu vložit. Všechna vývojová prostředí hlavních výrobců FPGA obsahují také plně konfigurovatelná jádra implementující asynchronní FIFO.

Přechod rychlých datových cest nebo sběrnic mezi asynchronními doménami je realizován právě umístěním asynchronní FIFO do cesty. Tím sice dojde ke zvýšení latence celé cesty, ale zamezí se případným problémům s metastabilitou nebo nekoherencí.

Redukce počtu hodinových domén

Často je možné počet hodinových domén zredukovat. Redukce počtu domén výrazně omezí možnost vzniku chyb a problémů způsobených metastabilitou nebo nekoherencí a také často zrychlý běh implementačních nástrojů (časová analýza, ale i optimalizace časování během syntézy a implementace proběhne tím rychleji, čím méně je v návrhu přechodů mezi asynchronními hodinovými doménami)

Redukce počtu hodinových domén je možná v zásadě dvěmi způsoby. V případě, že je skutečně nutné použít fyzicky rozdílné hodinové signály, je dobré zvolit jeden hlavní hodinový signál a ten použít pro většinu návrhu. V ostatních hodinových doménách pak řešit jenom nejnutnější minimum, jako jsou například vnější rozhraní a co nejrychleji přejít do hlavní, systémové, hodinové domény.

Druhá možnost je použitelná tam, kde skutečně fyzicky rozdílné hodinové signály nutné nejsou. V takovém případě je možné použít pouze jediný hodinový signál o dostatečně vysoké frekvenci a pomocí děliček k němu vyrobit několik pomocných signálů typu *clock-enable*. Jednotlivé části návrhu pak budou pracovat vždy na společném hodinovém signálu, ale mohou využívat různé signály clock-enable. Každý clock-enable aktivuje danou část obvodu pouze v některých periodách systémových hodin. Jednotlivé části obvodu však mohou být přímo propojeny, protože se jedná o plně synchronní obvod s jediným hodinovým signálem.

HDL jazyky

V současnosti jsou nejpoužívanější jazyky z rodiny HDL (system)Verilog a VHDL – které se liší syntaxí i sématikou. Oba poskytují obdobné prostředky k popisu logických obvodů.

Umožnují popsat logický obvod na různých úrovních abstrakce a jsou využitelné částečně i k simulaci a verifikaci.

Obvykle se rozlišují podmnožiny jazyka použitelné pro

- syntézu – HDL popis je syntetizovatelný a použitelný pro návrh HW

- modelování – HDL popis modeluje log.obvod (možno včetne zpoždění log.členů atp)

- simulaci/verifikaci – HDL popis slouží pro řízení simulace a ověření modelu

Jazyky HDL (Verilog i VHDL) umožňují popis obvodů na různých úrovních abstrakce:

- Behaviorální – popis na vysoké úrovni abstrakce – podpora částečne omezená (ekv.algorimu)

- RTL – typická úroveň , popis kombinace komb. a seq. Prvků

- Strukturální – popis vzájemného propojení bloků a modulů (ekvivalent blok.schématu)

- Hradla – nejnižší úroveň (ekvivalent podrobného schématu)

(! HDL není prog.jazyk (sekv.vykonávané příkazy) – je to jazyk pro popis HW (vše paralelně))

Verilog

Vývoj od 80-let. Poprvé standarizován v r.1995. Rozšiřován a dále standarizován v r.2001,2005.

(Syntaxí lze přirovnat k jazyku C)

SystemVerilog

rozšíření Verilog – rozšíření ve směru podpory modelování a verifikace přidává podporu objektů, vektorů a rozšíření RTL popisu ve směru nových dat.typů, operátorů, výrazů a procedurálních bloků.

r.2009 byly oba standardy (verilog a systemverilog) sloučeny, další standard je z r.2012 (SytemVerilog-2012)

VHDL

zkratka - Very High Speed Integrated Circuit Hardware Description Language. Vývoj začal v r.1981 US ministerstvo obrany – popis pro popis log.systemů. První standartd v r.1987, další revize 1993, 2000, 2002 a 2008.

(syntaxí lze přirovnat k jazyku ADA)

Příklad synchronního 4bitového čítače s resetovacím vstupem.

VHDL	Verilog (SystemVerilog)

`library ieee;`
`use ieee.std_logic_1164.all;`
`use ieee.numeric_std.all;`
 	
`entity counter is`
`  port(clk, rst : in  std_logic;`
`       q       : out std_logic_vector(3 downto 0));`
`end counter;`
 	
`architecture rtl of counter is`
`  signal cnt: unsigned(3 downto 0);`
 	
`begin`
`  CNT_PROC: process (clk)`
`  begin`
`    if rising_edge(clk) then`
`      if rst = '1' then`
`        cnt <= "0000";`
`      else`
`        cnt <= cnt + 1;`
`      end if;`
`    end if;`
`  end process;`
 	
`  q <= std_logic_vector(cnt);`
`end rtl;`
	`module counter (clk, rst, q);`
	
	`  input clk, rst;`
	`  output [3:0] out;`
	
	`  reg [3:0] cnt;`
	
	`  always @(posedge clk)`
	`  begin`
	`    if (rst)`
	`      cnt <= 4'b0000;`
	`    else`
	`      cnt <= cnt + 1'b1;`
	`    end`
	
	`  assign q = cnt;`
	
	`endmodule`

Vysokoúrovňové jazyky

Jazyky určené k popisu logických obvodů, které ale popisují obvody na vyšší úrovni abstrakce

System C - vychází z C++, standarizováno 2005

HLS – vychází z C, od firmy Xilinx

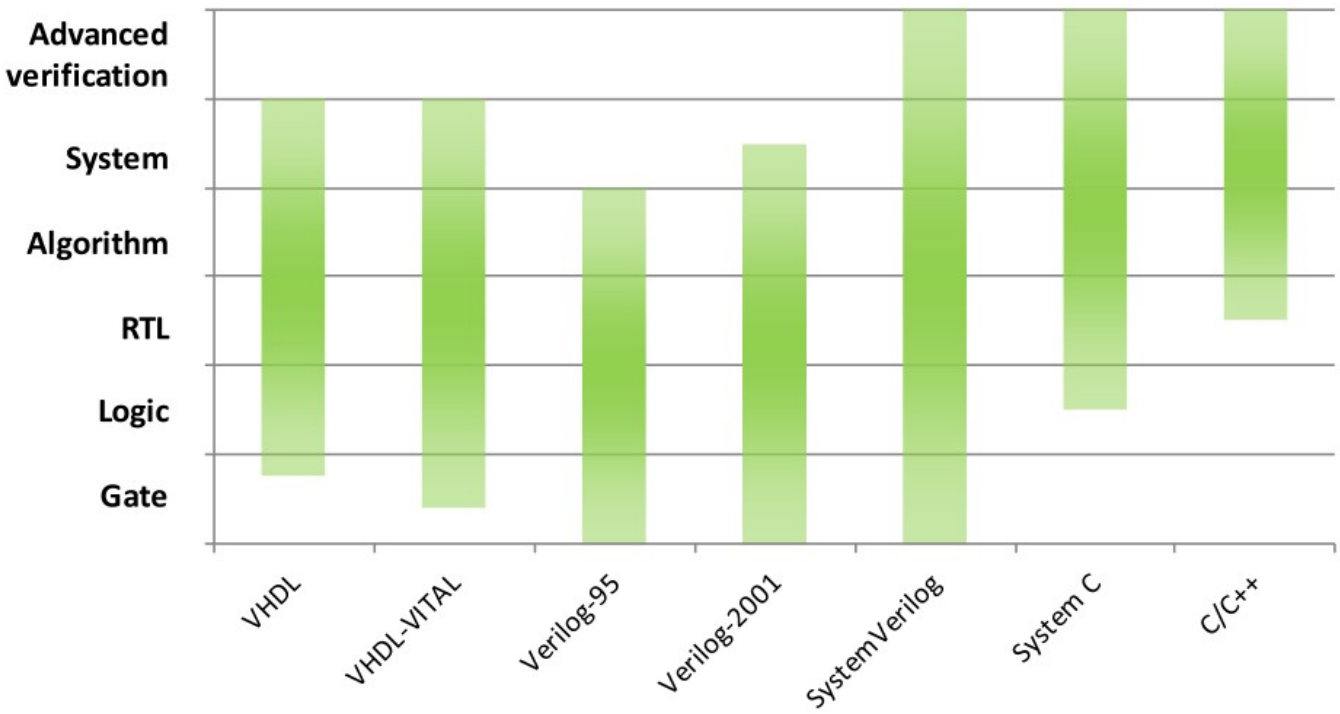
Handel-C – vychází z C

myHDL – open source, podmožina jazyka Pythom

Verifikační a kombinované jazyky

S růstem složitosti obvodů a programovatelné logiky přestávalo stačit ověření funkce logických obvodů pomocí jednoduchých simulací. Začaly vznikat různé metody formální verifikace a společně s nimi i jazyky umožňující tyto metody implementovat. Jedná se o jazyky jako *Vera*, *e*, částečně *PSL*, ale také

třeba *C* nebo *C++* a z něj vycházející *SystemC*. Tyto jazyky jsou často kombinovány s klasickými HDL. Jak Verilog, tak i VHDL například obsahují podporu pro PSL (Property Specification Language) a umožňují tak psát samotestovací kód přímo v popisu logického obvodu. Vlastní verifikační prostředí pak bývá realizováno v některém verifikačním jazyce, který umožňuje formální verifikaci obvodu případně i modelování celých systémů (kromě logického obvodu např. i procesor s programem, sensorické vstupy atp)



Užitečné informace a odkazy

Jazyky VHDL, Verilog, SystemVerilog, SystemC, PSL jsou standardizovány organizací IEEE. Souhrn odkazů na jednotlivé standardy je uveden na webu organizace Accellera, která vývoj těchto jazyků zastřešuje - Accellera IEEE Standards.

Webové stránky výrobců programovatelných logických obvodů:

- Altera Corporation

- Xilinx, Inc.

- Lattice Semiconductor Corporation

- Microsemi Corporation

Vývojová prostředí:

- Xilinx ISE - starší vývojový systém firmy Xilinx, který již nepodporuje jejich nejnovější FPGA. Plná podpora obvodů končí u řad Spartan/Virtex-6. Varianta WebPACK je k dispozici zdarma.

-Xilinx Vivado - aktuální vývojový systém firmy Xilinx. Vivado podporuje pouze novější obvody řad 7 a UltraScale. k dispozici je varianta WebPACK zdarma.

- Altera Quartus II - kompletní vývojové prostředí firmy Altera. na rozdíl od systémů firmy Xilinx neobsahuje vlastní HDL simulátor. Altera proto dodává speciální verzi simulátoru ModelSim firmy Mentor Graphics označovanou jako ModelSim-Altera Edition. je k dispozici zdarma ve variantě Web Edition.

- Lattice Diamond - je vývojové prostředí pro FPGA a CPLD firmy Lattice. Opět je k dispozici i licence zdarma, která v tomto případě podporuje pouze starší obvody.

- Microsemi Libero SoC - je kompletní vývojové prostředí pro novější programovatelné obvody firmy Microsemi. Pro starší obvody existuje systém Libero IDE. Libero SoC/IDE existuje opět ve verzi zdarma, která nepodporuje některé větší obvody.