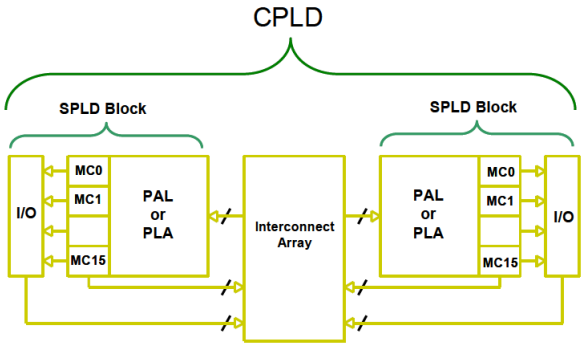
5 Programirljiva vezja

5.1 Kompleksna programirljiva vezja - CPLD

Sodobna programirljiva vezja delimo v dve veliki skupini: CPLD in FPGA. Vezja CPLD (angl. Complex Programmable Logic Device) so manjša in zgrajena iz enostavnih SPLD blokov, ki vsebujejo programirljivo matriko PAL ali PLA. Slika 5.1 predstavlja blokovno shemo vezja CPLD. Kombinacijski izhodi matrike PLA so vezani na makrocelice (MC) znotraj katerih so pomnilni gradniki, s katerimi naredimo sekvenčna vezja. Signali so povezani preko vhodno/izhodnih (I/O) celic na zunanje priključke ali pa na povezovalno mrežo (angl. Interconnect Array), ki med seboj povezuje posamezne SPLD bloke.



Slika 5.1: Blokovna shema vezja CPLD.

Pri programiranju vezja se določijo povezave v matriki PLA, povezave v makrocelicah in I/O blokih ter povezovalni matriki. S takšno strukturo lahko naredimo poljubna digitalna vezja, omejeni smo le z velikostjo CPLD gradnika. Vezja CPLD omogočajo izdelavo logičnih vezij z nekaj

1000 logičnimi vrati in nekaj 100 flip-flopi, ki delujejo pri frekvencah ure do okoli $200\,\mathrm{MHz}$. Imajo programski pomnilnik vrste FLASH, ki ga lahko večkrat zapišemo in ohrani vsebino ob izklopu napajanja. Naštejmo nekaj prednosti programirljivih vezij CPLD pred ostalimi tehnologijami:

- enostaven razvoj prototipa vezja, ki ga opišemo na računalniku, prevedemo in naložimo na razvojni sistem,
- nizki stroški razvoja v primerjavi z integriranimi vezji, ki bi jih ob vsaki spremembi na novo naredili v tovarni polprevodnikov,
- omogočajo hiter razvoj izdelka, saj je danes zelo pomembno da nov izdelek čim hitreje spravimo na tržišče,
- zmanjšanje komponent na tiskanem vezju - namesto množice logičnih gradnikov v obliki posameznih integriranih vezij lahko vse skupaj naredimo z enim vezjem CPLD (najmanjše med vezji CPLD ima 32 priključkov v $5 \times 5\,\mathrm{mm}$ ohišju),
- proizvajalci naprav ne potrebujejo velikih zalog komponent v življenjski dobi naprave, saj lahko načrt vezja hitro prenesemo v drugo programirljivo vezje, ki se bo obnašalo enako.

V knjižnicah programirljivih vezij najdemo še celo vrsto vnaprej pripravljenih gradnikov, ki jih nismo opisali. Razdeljeni so v več kategorij, znotraj katerih so gradniki s podobnimi lastnostmi. Tabela 5.1 prikazuje kategorije gradnikov za programirljiva vezja CPLD Coolrunner-II, njihove angleške oznake in število v posamezni kategoriji. Za risanje logične sheme imamo na voljo več kot 300 različnih elementov, med katerimi so:

- ojačevalniki so potrebni na vsakem vhodu in izhodu integriranega vezja, uporabimo jih tudi za ločevanje signalov v shemi,
- logična vrata imajo 2-8 vhodov v normalni ali invertirani obliki,
- binarni izbiralniki in dekodirniki,
- kombinacijski seštevalniki in odštevalniki ter sekvenčni akumulatorji,
- primerjalniki, ki primerjajo številske vrednosti dveh večbitnih signalov,
- kombinacijski pomikalniki, ki premaknejo večbitni signal za poljubno število bitov,
- zapahi in flip-flopi z različnimi kontrolnimi signali,
- sestavljeni števci z različnimi načini delovanja,
- pomikalni registri in
- delilniki ure, ki so posebne vrste števcev.

Kategorija	angl. oznaka (število)	Vrsta gradnikov
Ojačevalniki	Buffer (15)	ojačenje ali delitev signalov
Vhodno/izhodni	IO (17)	ojačevalniki na priključkih
Logična vrata	Logic (108)	Boolova logična vrata INV, AND2, OR2, AND3...
Izbiralniki	Mux (7)	izbiralniki M2_1, M4_2E, M8_3E...
Dekodirniki	Decoder	binarni dekodirniki D2_4E, D3_8E...
Aritmetični	Arithmetic (12)	seštevalniki, odštevalniki, akumulatorji
Primerjalniki	Comparator (8)	primerjalniki enakosti in magnitude
Pomikalniki	Shifter (2)	4-bitni in 8-bitni pomikalnik
Zapahi	Latch (11)	zapahi LD, Data Gate LDG...
Flip-flopi	Flip-Flop (76)	robno proženi flip-flopi FD, FD8, FDCE...
Števci	Counter (70)	2-, 4- 8- in 16-bitni števci
Pomikalni registri	Shift Register (36)	4-, 8- in 16-bitni pomikalni registri
Delilniki ure	Clock Divider (32)	2-, 4-, 6-, 8-, 10-, 12-, 14- in 16-bitni delilniki

Tabela 5.1: Kategorizacija gradnikov v programirljivih vezjih.

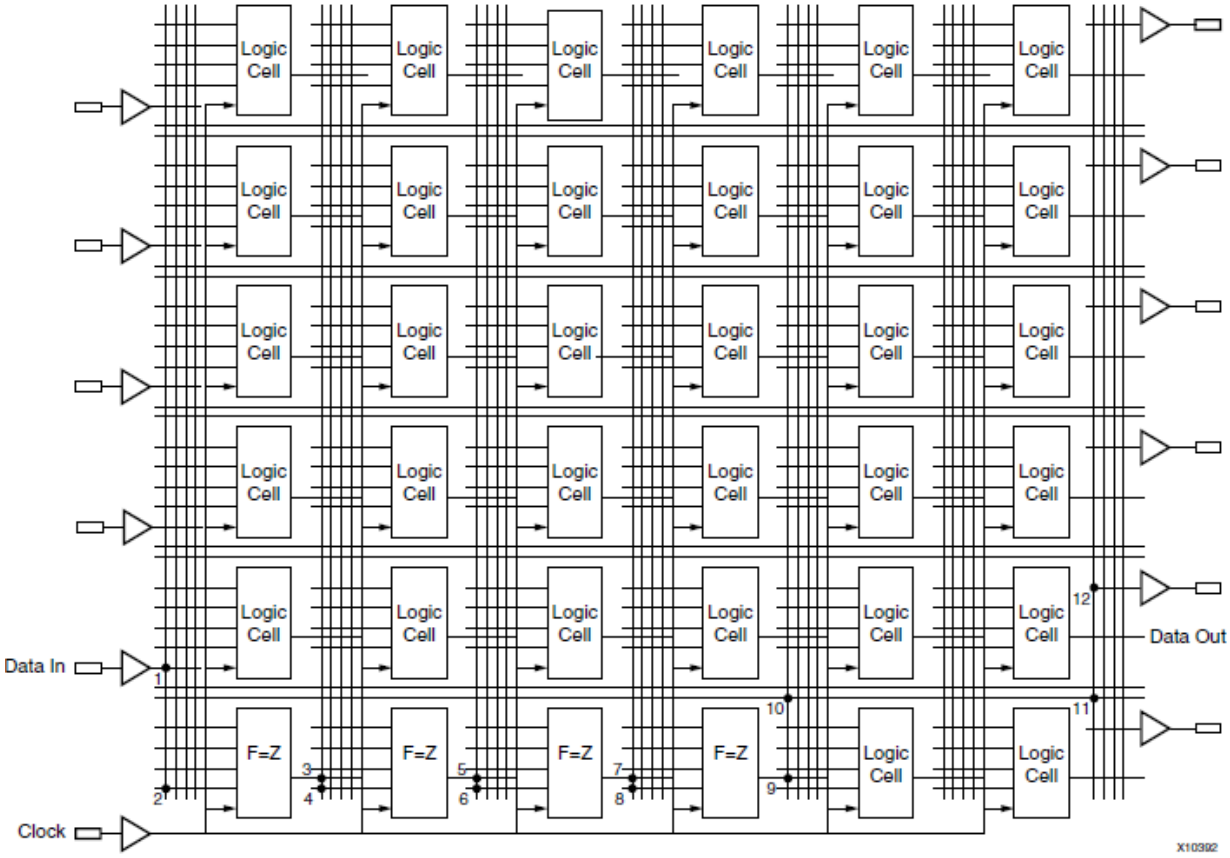
5.2 Programirljive matrike - FPGA

Vezja z oznako FPGA (angl. Field Programmable Gate Array) so narejena v obliki matrike logičnih celic, med katerimi je programirljivo povezovalno polje, kot prikazuje slika 5.2. Logične celice vsebujejo tabele za izvedbo majhnih kombinacijskih gradnikov in flip-flope. Okoli programirljive matrike so še vhodno/izhodne celice, ki povezujejo signale na zunanje priključke. Vezja FPGA imajo v matriki veliko število celic in največja med njimi omogočajo izdelavo vezij z več kot 10 milijoni logičnih vrat! Poznamo različne tehnološke izvedbe, največ pa jih je v standardnem CMOS procesu, kjer se programski podatki zapišejo v zapahe. Zapah ob izklopu napajanja izgubi shranjeno stanje, zato imamo poleg vezja FPGA na plošči še pomnilnik iz katerega se ob startu naloži vsebina.

5.3 Načrtovanje s programirljivimi vezji

Postopek načrtovanja vezja začnemo z opisom vezja v grafični obliki (npr. shema) ali v nekem jeziku za opis strojne opreme. Delovanje opisanega vezja lahko preverimo oz. verificiramo s simulacijo. Naslednji korak pri načrtovanju s programirljivimi vezji je implementacija, s katero pripravimo datoteko za nalaganje v programirljivo vezje, kot prikazuje slika 5.3. Ko vezje naložimo, preverimo delovanje na strojni opremi (npr. na razvojnem sistemu). Ta postopek imenujemo strojna verifikacija.

Delo razvojnega inženirja je predvsem opis vezja in priprava verifikacije. Precej zapleten postopek implementacije vezja, s katero logično vezje prevedemo, prilagodimo in pretvorimo v programsko datoteko, je na srečo avtomatiziran. Programska oprema za računalniško načrtovanje vezij zahteva le nekaj nastavitev, da se implementacija izvede pod želenimi pogoji. Rezultat implementacije lahko ponovno verificiramo s simulacijo, ki tokrat vsebuje tudi ocenjene zakasnitve

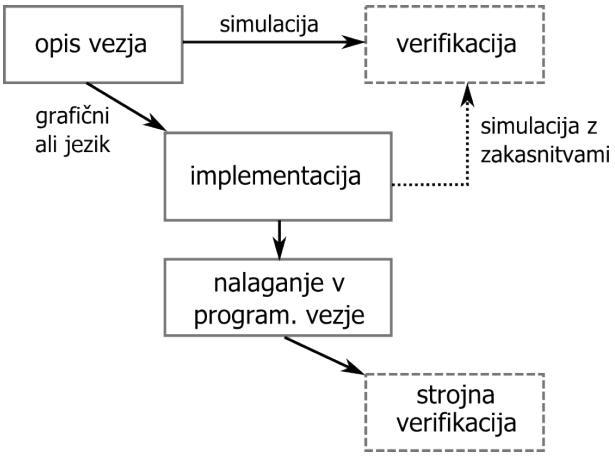


Slika 5.2: Blokovna shema vezja FPGA.

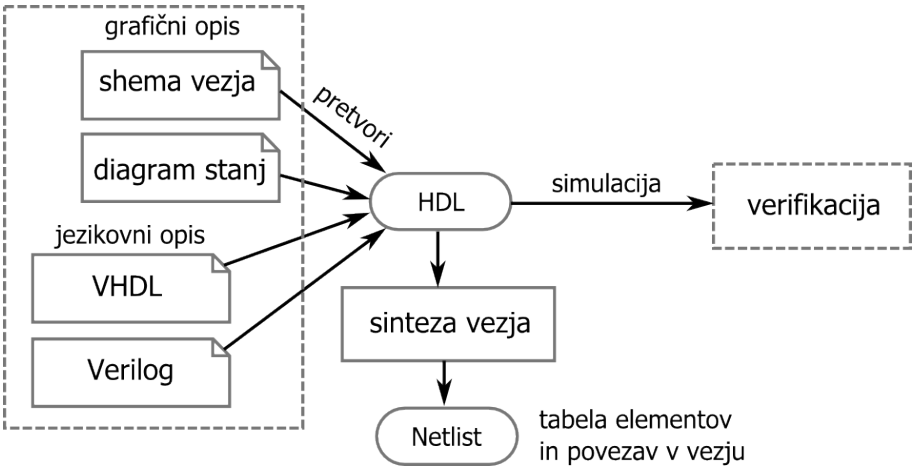
gradnikov vezja. Postopek nalaganja programske datoteke v programirljivo vezje je odvisen od strojne opreme in izvedbe komunikacije z računalnikom. Običajno ta korak ni zahteven, saj moramo le izbrati pravilne nastavitve in datoteko, ki jo programska oprema naloži v vezje. Postopek strojne verifikacije je odvisen predvsem od razvite aplikacije (nalog, ki jih vezje izvaja). V najbolj preprosti obliki zadošča ročno nastavljanje signalov in vizualni pregled delovanja v zahtevnejših primerih pa potrebujemo posebno merilno opremo.

5.3.1 Opis vezja

Programska oprema za računalniško načrtovanje vezij pozna različne načine vnosa vezja, ki jih v grobem razdelimo v grafični in jezikovni opis. Primer grafičnega opisa je shema vezja. Shemo narišemo v grafičnem urejevalniku s postavljanjem elementov iz knjižnice digitalnih gradnikov in risanjem povezav. V tem poglavju bomo spoznali še en grafični opis vezja v obliki diagrama stanj. Jezikovni opis logičnega vezja predstavljajo npr. Boolove enačbe. Jezik za opis strojne opreme HDL (angl. Hardware Description Language) določa pravila takšnega opisa. V današnjem času se uporabljata predvsem jezika VHDL in Verilog, ki omogočata opis vezja z enačbami ali pa v obliki algoritma.



Slika 5.3: Osnovni koraki načrtovanja digitalnih vezij.



Slika 5.4: Načini opisa digitalnega vezja.

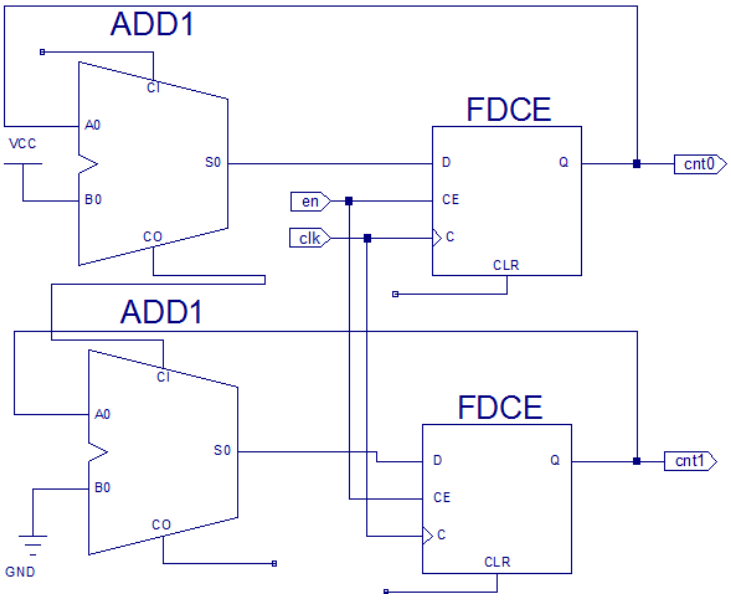
Programska oprema datoteke z grafičnim opisom avtomatsko prevede v jezikovno obliko (HDL), ki je osnova za izvedbo simulacije in ostalih korakov prevajanja, kot prikazuje slika 5.4. Jezikovni opis vezja v koraku sinteze pretvorimo v datoteko, ki vsebuje vse elemente končnega vezja in povezave med njimi (angl. netlist). Ta datoteka je osnova za implementacijo vezja.

Primer 4.1

Prikazali bomo različne načine opisa vezja 2-bitnega števca. Števec ima vhod za omogočanje en in uro clk. Ob prednji fronti ure naj se vrednost na izhodnem signalu cnt poveča za 1. Izpis 5.1 predstavlja opis števca v jeziku VHDL, kjer uporabimo proces v katerem se vrednost signala cnt poveča ob naraščajoči fronti ure in pogoju $en = 1$. Slika 5.5 prikazuje shematski opis števca, ki je sestavljen iz dveh seštevalnikov in flip-flopov. Shemo vezja programska oprema pretvori v strukturni opis v jeziku VHDL ali pa Verilog, kot je prikazano na izpisu 5.2.

Listing 5.1: Opis 2-bitnega števca v jeziku VHDL

```
entity stevec is
    Port ( clk : in  STD_LOGIC;
           en : in  STD_LOGIC;
           cnt : buffer  STD_LOGIC_VECTOR(1 downto 0));
end stevec;

architecture opis of stevec is
begin
 p: process(clk)
 begin
   if rising_edge(clk) and en='1' then
     cnt <= cnt + 1;
   end if;
 end process;
end opis;
```



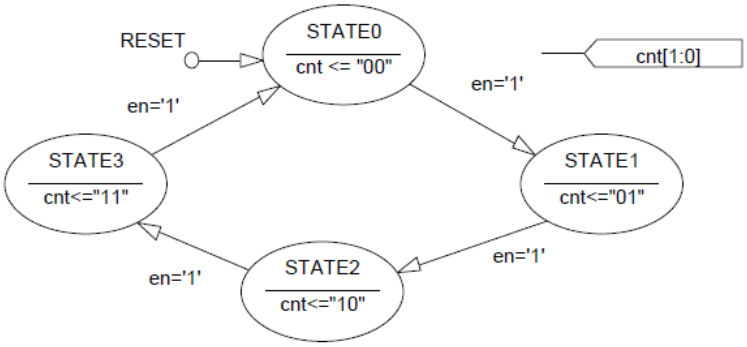
Slika 5.5: Shematski opis 2-bitnega števca.

Listing 5.2: HDL datoteka iz sheme vezja (Verilog)

```
module cnt(clk, en, cnt0, cnt1);
   input clk, en;
   output cnt0, cnt1;

   wire XLXN_1, XLXN_2, XLXN_3, XLXN_8, XLXN_9;
   wire cnt0_DUMMY, cnt1_DUMMY;

   assign cnt0 = cnt0_DUMMY;
   assign cnt1 = cnt1_DUMMY;

   ADD1_MXILINX_cnt XLXI_1 (.A0(cnt0_DUMMY),
                            .B0(XLXN_9),
                            .CI(),
                            .CO(XLXN_1),
                            .S0(XLXN_2));

   ADD1_MXILINX_cnt XLXI_2 (.A0(cnt1_DUMMY),
                            .B0(XLXN_8),
                            .CI(XLXN_1),
                            .CO(),
                            .S0(XLXN_3));

   FDCE XLXI_3 (.C(clk),
                .CE(en),
                .CLR(),
                .D(XLXN_2),
                .Q(cnt0_DUMMY));

   FDCE XLXI_4 (.C(clk),
                .CE(en),
                .CLR(),
                .D(XLXN_3),
                .Q(cnt1_DUMMY));

   VCC XLXI_5 (.P(XLXN_9));
   GND XLXI_6 (.G(XLXN_8));
endmodule
```

Delovanje sekvenčnega vezja, kot je 2-bitni števec, lahko opišemo tudi z diagramom stanj. Diagram prikazuje stanja v registrih vezja in pogoje za prehode. Slika 5.6 predstavlja diagram stanj s štirimi stanji: STATE0, STATE1, STATE2 in STATE3, ki predstavljajo kombinacije na izhodu števca od ”00”do ”11”. Puščice med stanji predstavljajo prehode. Pogoj za prehod v novo stanje je vedno fronta ure in pogoj na puščici en = ’1’. Signal $reset$ določa začetno stanje. Programska oprema prevede grafični opis diagrama stanj v opis vezja (HDL), ki je primeren za

verifikacijo ali implementacijo.

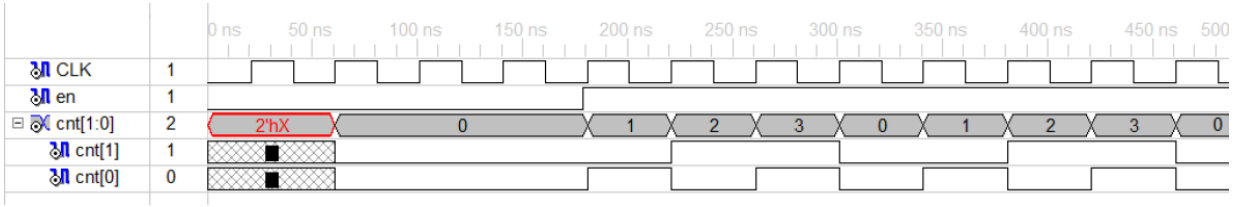


Slika 5.6: Diagram stanj 2-bitnega števca.

5.3.2 Verifikacija

Računalniška simulacija modela vezja je osnovni postopek preverjanja oz. verifikacije vezja. Simulacijo pripravimo tako, da določimo časovni potek spreminjanja signalov na vhodu vezja. Časovni potek vhodnih signalov določimo v testni strukturi (angl. test bench) in je zapisan v grafični obliki, v jeziku HDL ali pa v obliki simulacijskih makrojev. Ko je testna struktura pripravljena, poženemo simulator in pregledamo rezultate na časovnem diagramu signalov (angl. waveform).

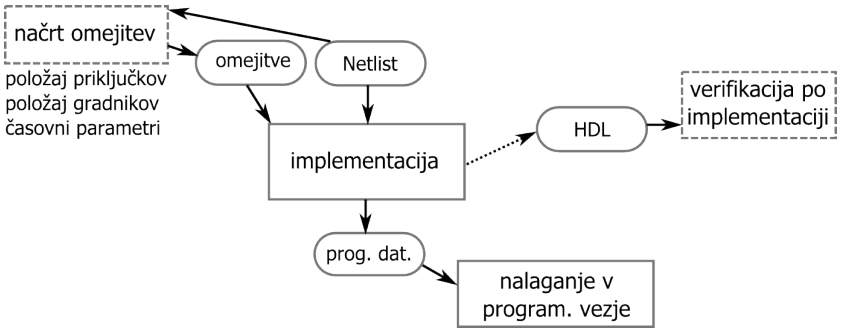
Slika 5.7 prikazuje časovni diagram simulacije 2-bitnega števca. V testni strukturi smo nastavili uro in signal en, opazujemo pa vrednost izhoda cnt. Večbitne signale, kot je $cnt(1:0)$, lahko opazujemo v obliki dvojiških, desetiških ali šestnajstiških številskih vrednosti, možen pa je tudi prikaz posameznih bitov $cnt(1)$ in $cnt(0)$.



Slika 5.7: Rezultat simulacije 2-bitnega števca.

S pregledom časovnega diagrama ugotovimo, ali se vezje obnaša tako, kot smo pričakovali. V testni strukturi lahko tudi vnaprej predpišemo pričakovane vrednosti izhodov ali pa določimo pravila spreminjanja signalov in tako avtomatiziramo postopek verifikacije. Če želimo z verifikacijo res preveriti delovanje vezja, moramo dobro poznati signale, ki jih lahko pričakujemo na vhodu realnega vezja in pripraviti testne strukture za veliko različnih primerov.

5.3.3 Implementacija



Slika 5.8: Priprava implementacije vezja.

Pred avtomatsko implementacijo moramo podati zahteve oz. datoteko z omejitvami (angl. constraints file) v kateri določimo položaje zunanjih priključkov, lahko pa tudi notranjih gradnikov ali želene časovne parametre končnega vezja. Rezultat implementacije je programska datoteka, ki jo naložimo v programirljivo vezje. Delni rezultat implementacije je lahko tudi ponoven jezikovni opis vezja (HDL), ki pa sedaj vsebuje še ocenjene zakasnitve in ga uporabimo za bolj natančno verifikacijo vezja po implementaciji.

5.3.4 Nivoji opisa vezij

Programska oprema za računalniško načrtovanje digitalnih vezij omogoča opisovanje vezij na različnih nivojih. Čeprav so vezja fizično narejena iz osnovnih stikalnih gradnikov - transistorjev, digitalna vezja raje obravnavamo na nivoju logičnih vrat ali pa na višjih nivojih, kot so prikazani v tabeli 5.2. Načini opisa so grafični v obliki sheme ali diagrama ali pa jezikovni v obliki enačb ali HDL kode. Orodja za analizo in obravnavo vezij so simulatorji, na višjih nivojih opisa pa tudi orodja, ki izvajajo sintezo opisa na nivo logičnih vrat.

Nivo opisa	**transistorji**	**logična vrata**	**gradniki**	**registri - RTL**
primer	PMOS	AND, OR	izbiralnik	števec, avtomat
način opisa	shema stikal	shema, HDL	blok. shema, HDL	diagram, HDL
orodja	simulator vezij	log. simulator	HDL sim. in sinteza	HDL sim. in sinteza
velikost	20 stikal	20 vrat	10 gradnikov	10 registrov, stanj
št. log. vrat	5	20	200	2000

Tabela 5.2: Nivoji opisa digitalnih sistemov.

Višji nivoji opisa uporabljajo manj podrobne, *poenostavljene* (abstraktne) *modele*, ki omogočajo opis večjih vezij z manj elementi. Tipično število elementov, ki jih opišemo na enem listu je nekaj 10, z uporabo modelov na višjem nivoju pa lahko hitro opišemo vezje z nekaj 1000 logičnimi vrati. Za večja vezja uporabljamo *hierarhičen* način načrtovanja, pri katerem kompleksno vezje razdelimo na več sklopov, ki jih ločeno opišemo in kasneje povežemo med seboj.