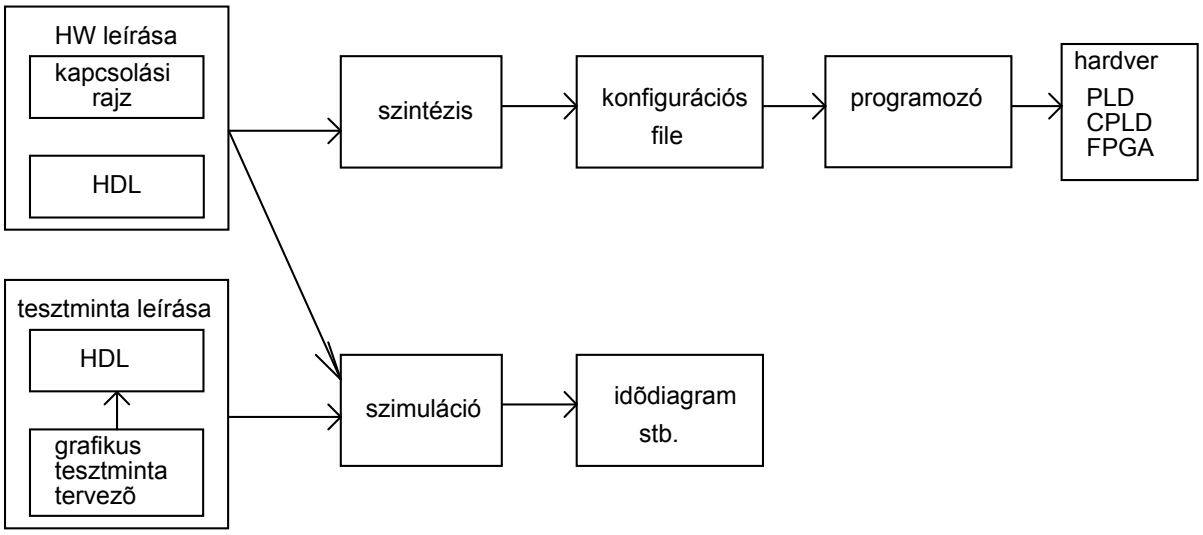
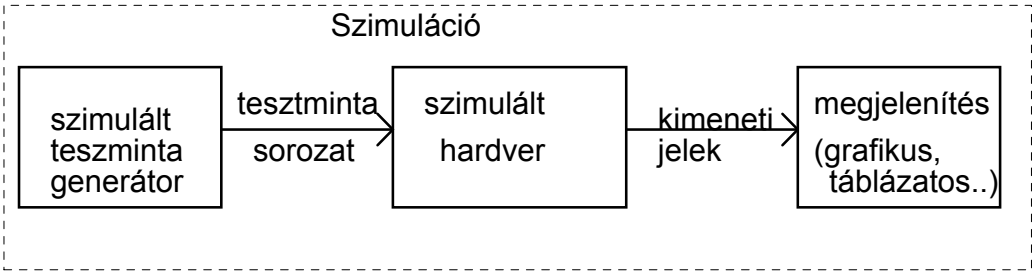
Hardver leíró nyelvek (HDL)

Hardver leíró nyelvek (HDL)

nyelvek: ABEL, VHDL, ***VERILOG*** stb.





Verilog

Az alábbiakban a Verilog nyelv alapjaiba próbáljuk bevezetni az olvasót.

A változók

A leírásban használt változók emlékezet nélküli és tároló változók lehetnek.

emlékezet nélküli változó: **wire** A;
Pl. modulok összekötésére használjuk. Modul kimenete és bemente is lehet ilyen tipusú.

tároló tipusú változó: **reg** B;
Értékét megtartja, amíg újabb értékadás nem történik. Modulnak csak a kimenete lehet ilyen tipusú.

Egyéb emlékezet nélküli és tároló jellegű változók is léteznek.

A változók több bitesek (vektor változók) is lehetnek.
wire [3:0] X; // 4 bites változó
reg [7:0] Y; // 8 bites

A modul

Alapegység a modul. Ha HW leírására használjuk, akkor ez egy funkcionális elemhez hasonló, a bemeneti és kimenetei között a leírásnak megfelelően viselkedő logikát ír le.
A szimulációhoz leírhat teszminta generáló egységet is.

module modulnév (bemeneti_változó1, bemeneti_változó2,...kimeneti_változó1...)
// komment
input bementi_változó1;
input bementi_változó2;
...
output kimeneti_változó1;

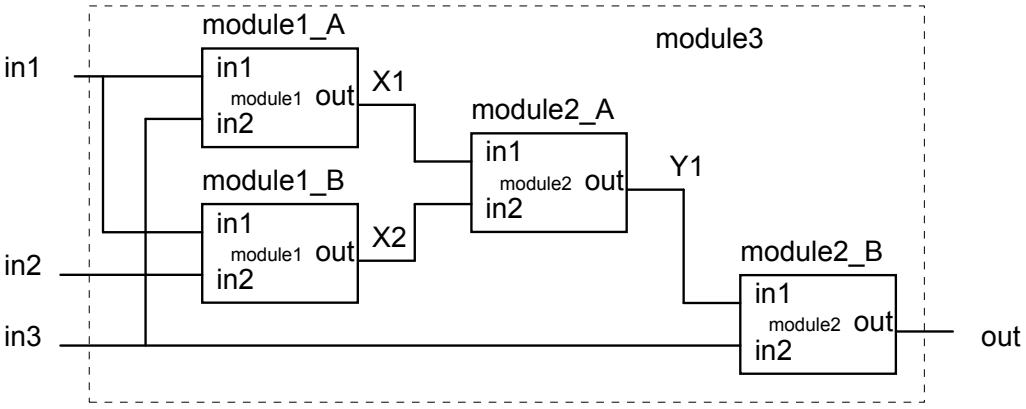
reg regiszter_tipusú_változó; // “emlékező” változó
wire huzal_tipusú_változó; // nem emlékező változó
..
A modul törzse (a modul törzsében írjuk le annak viselkedését
endmodule

Primitívek

A nyelvbe be vannak építve elemi modulok (primitívek). Ezek a logikai alapelemeket reprezentálják. (and, nand, or, nor stb.)

Modulok összekapcsolása (hierarchikus leírás)

Az egyes modulok más modulokat is tartalmazhatnak, tetszőleges mélységben.



module module1(in1,in2,out)

…

endmodule

module module2(in1,in2,out)

…

endmodule

module module3(in1,in2,out)

…

endmodule

```
module module3(in1,in2,in3, out)
...
wire X1;
wire X2;
wire Y1;

module1 module1_A(in1,in3, X1);
module1 module1_B(in1,in2, X2);
module2 module2_A(X1, X2, Y1);
module2 module2_B(Y1, in3, out);
endmodule
```

A module1, module2 tartalmazza egy-egy modul működésének leírását.
A module1_A, module1_B a module1 egy-egy példánya (a module1 leírása szerint viselkedő modulok, melyeket a nevükkel különböztetünk meg egymástól).

A modulra való hivatkozáskor a nem használt paraméterek ,,-vel elhagyhatók. (Pl. több kimenetű modul esetén a nem használt kimenet esetén használjuk.)

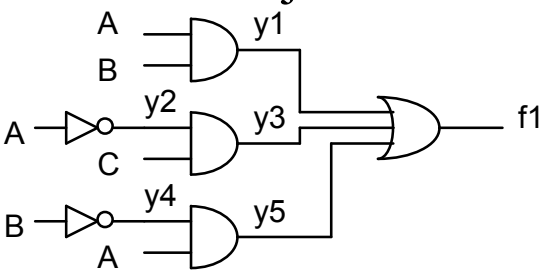
A modulok működésének leírása

A modulok működését többféle módon le lehet írni.

- explicit struktúrális leírás
- implicit struktúrális leírás
- procedurális leírás

Explicit struktúrális leírás

A struktúrális leírás azt adja meg, hogy hogyan kapcsolódnak össze az előre definiált modulok egymással. Az öszekapcsolandó modulok primitívek és tetszőleges modulok lehetnek. Az alábbi példa a kapcsolási rajzzal adott f1 függvény expl. struktúrélis leírását mutatja be.




```
module KH(A, B, C, f1)
input A, B, C;
output f1;
wire y1, y2, y3, y4;

and (y1, A, B);
not (y2, A);
and (y3, y2, C);
not (y4, B);
and (y5, y4, A);
or (f1, y1, y3, y4);
endmodule
```

Implicit struktúrális leírás

Kombinációs elemekből felépülő logikát ír le. Jóval tömörebben írja le mint az explicit megadás.

Az előbbi példa kombinációs hálózatának implicit srtuktúrális leírására az **assign** *folytonos értékadó utasítás* használható.

```
wire f1;
assign f1= A&B | ~A&C | A&~B;
```

Az assign azonban aszinkron sorrendi hálózatot is leírhat.

```
assign Q = S | ~R&Q;
```

Az **assign** több bites ún. vektor változókat is kezel.

```
wire [3:0] X1, X2, Y;
assign Y = X1 | X2; // bitenkénti VAGY kapcsolat
```

A vektor változóknál tartomány is kijelölhető.

```
wire [7:0] A;
wire [2:0] B;
assign B = A[6:4];
```

Az értékadásnál a bitek számának meg kell egyezni az egyenlőség két oldalán.

Változók ***összkapcsolhatók*** a {}-jelekkel egy több bites egységgé, amelyeknek így egyszerre értéket lehet adni egy másik ugyanennyi bites kifejezés alapján.

```
wire [1:0] C;
wire [2:0] D;
wire [4:0] E
assign {C, D} = E;
```

Procedurális leírás

Ezel a fajta leírással az egység viselkedést adjuk meg, nem a struktúráját. A viselkedést utasítások sorozata adja meg.

always @(eseményvezérlő kifejezés)
begin
utasítás_1
utasítás_2
…
utasítás_*n*
end

A fel és lefutó él eseményekhez külön jelölés tartozik,
felfutó él: **posedge** változó
lefutó él: **negedge** változó

Az eseményvezérlő kifejezés alakja:
változó1 **or** változó2 **or** ... **or posedge** változó_i **or** ... **or negedge** változó_j ...
Ha nincs eseménylista, az folytonos eseményt jelöl.

Ha az eseményvezérlő kifejezésben váltzó neve szerepel, akkor az eseményt ennek bármely megváltozása, ill. a posedge vagy negedge-vel kijelölt megváltozása jelenti.

A **begin end** közötti rész ***akkor értékelődik ki, amikor az eseménylistában felsorolt valamely esemény bekövetkezik***.
Az utasítások a leírásuk sorrendjében értékelődnek ki, de a kiértékelés eredménye az esemény hatására okoz változást.

Értékadó utasítások

A változók értékadásánál használjuk.

Blokkoló értékadás, procedurális leírásban blokkolja a további utasítások kiértékelését: A = B;
Nem blokkoló értékadás: A<=B;

Feltételes elágazó utasítások

if (feltétel) utasítás;
if (feltétel) utasítás; **else** utasítás;
if (feltétel) **else** utasítás; **if** (feltétel) utasítás;..***else*** utasítás;

Ha több utasítást kell végrehajtani az egyes ágakon, akkor azokat **begin end** közé kell tenni.

if (feltétel)
begin
utasítás;
utasítás;
end

Többutas elágazás megvalósítása: **case**

case (kifejezés)
konstans1: utasítás;
konstans2: utasítás;
konstans3: utasítás;
default*: utasítás;*
endcase

feltételes operátor:
változó = feltétel **?** igaz_eset **:** hamis_eset

Operátorok

Különféle kifejezésekben használható műveleteket jelölnek ki.

{}	csoportosítás
+ - * /	aritmetikai műveletek
%	maradékképzés
> >= < <=	relációs operátorok
! && || == !=	logikai negálás, és, vagy, egyenlő, nem egyenlő
~ & | ^ ~^	bitenkénti negálás, és, vagy, antivalencia és eqvivalencia
& ~& | ~| ^ ~^	redukáló műveletek, változók bitjei között &(1101)=0 |(1101)=1
<< >>	logikai shift művelet
?:	feltételes kifejezés jel = feltétel ? igaz érték : hamis érték ;

Számok megadása

A szintaxis: bit_szám számrendszer számjegyek
A számrendszer 'b bináris, 'o oktális, 'd decimális, 'h hexadecimális lehet.
Pl. a 12 megadása 4 biten hexadecimálisan: 4 ‘hc
a 12 megadása 6 biten binárisan: 6’b001100

Konstansok definiálása

parameter konstans_név = 4’b1001;