Deel 2: Praktische toepasbaarheid

De Kracht van VHDL

De VHSIC Hardware Description Language (VHDL) is halverwege de jaren tachtig ontwikkeld om digitale schakelingen te modelleren. De taal combineert een structurele met een functionele beschrijving en is geschikt als hulpmiddel voor synthese zowel als simulatie van digitale systemen. In een eerder artikel werden de achtergronden van VHDL geschetst; in dit tweede deel wordt nader ingegaan op de toepasbaarheid. Digitale systemen worden in een VHDL-simulatie gemodelleerd als samenwerkende parallelle processen, mogelijkerwijs met tijdvertragingen. Synthese-tools kunnen bepaalde VHDL-constructies afbeelden op elektronische basisschakelingen.

Hans van Thiel

In een beschrijvingstaal als VHDL worden twee hoofdfuncties samengebracht. Ten eerste is het een simulatietaal waarmee een ontwerper, met behulp van een simulatie-tool, digitale schakelingen kan simuleren. Het is dus een hulpmiddel om het ontwikkelproces te ondersteunen en een ontwerp te verifiëren. Ten tweede kan een VHDL-beschrijving dienen als invoer voor een synthese-tool. Een dergelijk pakket kan, ook door hardwarebibliotheken te gebruiken, uit bepaalde VHDL-constructies automatisch corresponderende digitale schakelingen genereren.

```
architecture ...
begin
        hier_mag_een_label:process
        begin
                qi <= r nor qi_not after vtd;  -- sequentieel statement
                wait on r, qi_not;
        end process;

-----------------------------------------
Dit is identiek met het volgende concurrent statement
-----------------------------------------

architecture ...
begin
        qi <= r nor qi_not after vtd;  -- concurrent statement
```

Listing 1. In sommige gevallen kan een VHDL-proces eenvoudiger worden geschreven. De VHDL-gereserveerde woorden zijn vetgedrukt en het symbool — geeft aan dat de tekst erna tot aan het eind van de regel commentaar is.

```
entity sr_latch is
        port (s, r : in bit;
              q, q_not : out bit);

-- waarheidstabel van een sr_latch
--        s  r   |  q  q_not
--        0  0   |  onveranderd
--        0  1   |  0  1
--        1  0   |  1  0
--        1  1   |  0  0
end sr_latch;

architecture gedrag of sr_latch is
        constant vtd :  time := 10 ns;
        signal qi : bit := '0';
        signal qi_not : bit := '1';
begin
        qi <= r nor qi_not after vtd;
        qi_not <= s nor qi after vtd;
        q <= qi;
        q_not <= qi_not;
end gedrag;
```

Listing 2. Entity en Architecture van een sr-latch naar ref. 2. Het gedrag van de sr_latch is gemodelleerd door vier samenwerkende parallelle processen die hier in hun impliciete vorm zijn geschreven.

```
type std_ulogic is ('U'     -- niet geinitialiseerd
                    'X'     -- afgedwongen onbekend
                    '0'     -- afgedwongen 0
                    '1'     -- afgedwongen 1
                    'Z'     -- hoog impedant
                    'W'     -- zwak onbekend
                    'L'     -- zwak 0
                    'H'     -- zwak 1
                    '-'     -- maakt niet uit
                    );
```

Listing 3. De meervoudige logica van IEE Standaard 1164. De standaard definieert ook logische operaties en resolutiefuncties. Alle huidige VHDL-tools ondersteunen deze logica.

```
entity NumberOfOnesGreaterThanTwo is
  generic (Lgt : positive := 4);
  port (inp : in bit_vector(Lgt-1 downto 0);
        o : out bit);
end NumberOfOnesGreaterThanTwo;


architecture behavior OF NumberOfOnesGreaterThanTwo is
  function NmbOfOnes(inp : bit_vector) return natural is
    variable cnt : natural;
  begin
    cnt := 0;
    for i in inp'range loop
      if inp(i)='1' then
        cnt := cnt + 1;
      end if;
    end loop;
    return cnt;
  end NmbOfOnes;
begin
  o <= '1' when NmbOfOnes(inp)>=2 else
       '0';
end behavior;
```

Listing 4 (bron: Molenkamp). Een VHDL beschrijving op functioneel niveau die automatisch gesynthetiseerd kan worden (zie fig. 3). De ingang inp is een bus is met breedte Lgt. De uitgang o is '1' als twee of meer bits van inp '1' zijn en anders '0'. De breedte Lgt is gemakkelijk achteraf aan te passen. Deze beschrijving illustreert ook het gebruik van variabelen en functies in VHDL.

De notatie en de syntaxis van VHDL zijn gestandaardiseerd in IEEE Std. 1076 - 1993. Weliswaar is ook het simulatiemodel onderdeel van de standaard en zijn er standaardbibliotheken en koppelingen naar andere verwante IEEE normen (zie ref. 1 en 2), maar de toepasbaarheid van VHDL hangt in feite af van de tools. In de VHDL-standaard staat niet beschreven hoe een bepaalde VHDL-constructie moet worden gesynthetiseerd. In de loop der jaren zijn synthese-tools steeds krachtiger geworden en in toenemende mate wordt een groter deel (subset) van VHDL ondersteund. Een nadeel is echter dat niet elke synthese-tool dezelfde subset van VHDL ondersteunt (hoewel de overeenkomsten vaak zeer groot zijn). Tot voor kort was er niets gestandaardiseerd op het gebied van VHDL-synthese. Géén standaard betekent dat een beschrijving tool-afhankelijk gaat worden en dat is zeker voor de onderhoudbaarheid onwenselijk. Standaardisatie van VHDL-synthese is in 1999 gerealiseerd in de IEEE-standaard 1076.6-1999, Standard for VHDL Register Transfer Level Synthesis. Deze norm beschrijft een subset van VHDL en hoe deze moet worden gerealiseerd. Het is de bedoeling in nieuwere versies van deze standaard de subset van VHDL die geschikt is voor synthese verder uit te breiden. In feite zal de standaard, waarschijnlijk, verschillende niveaus van ondersteuning definiëren waarbij een synthese-tool level 1 dan wel level 2 compliant zal zijn. Level 2 zal alle mogelijkheden van level 1 omvatten.

VHDL berust op enkele basisprincipes (zie het eerste deel van dit artikel) en die zullen dan ook in alle synthese- en simulatietools zijn terug te vinden. Zo heeft een digitale schakeling in VHDL een entity-beschrijving waarin de in- en uitgangspoorten (eventueel bi-directioneel) zijn aangegeven met hun type. Zo'n schakeling kan bestaan uit met elkaar verbonden sub-schakelingen die zelf ook weer gedefinieerd zijn in een entity beschrijving. De manier waarop de subsystemen zijn verbonden staat in een architecture; de verbindingen worden aangegeduid met de VHDL-term signal.

Op deze manier geeft een VHDL-beschrijving dus aan hoe een digitaal systeem is samengesteld uit een hiërarchie van subsystemen. In een architecture kan behalve structuur echter ook gedrag worden beschreven op een manier die doet denken aan programmeertalen als Pascal of ADA. Om een systeem te kunnen simuleren moeten de basiscomponenten allemaal een gedragsbeschrijving (in een eigen architecture) bezitten. De structurele opbouw en het gedrag van deze componenten bepalen vervolgens het gedrag van het systeem als geheel.

Simulatie van systeemgedrag is in VHDL altijd gebaseerd op samenwerkende parallelle processen (concurrency). Bovendien kent VHDL tijdmodellering, iets dat uiteraard onmisbaar is bij

simulatie van digitale systemen. De sr-latch uit listing 2 (naar Molenkamp) vormt daarvan een duidelijk voorbeeld.

Simulatie

Een gedragsbeschrijving in VHDL heeft sequentiële en parallelle aspecten. De sequentiële regels, inclusief controleconstructies en lussen, staan binnen een proces. Zo'n process kan een identificerende naam hebben (een label) en wordt opgestart als een of meer signalen veranderen. In sommige gevallen kan een proces eenvoudiger als een concurrent statement worden beschreven (zie listing 1). Dat is in listing 2 ook gebeurd, maar de vier signaaltoekenningen in de architecture tussen begin en end zijn wel degelijk processen die dus parallel verlopen en met elkaar communiceren. De volgorde waarin de vier concurrent statements in de beschrijving staan doet er niet toe. Samenvattend: statements in een proces zijn sequentieel; statements in een architectuur zijn parallel.

Het gedrag van de latch is gemodelleerd door twee nor-operaties met een tijdvertraging van 10 ns en twee signaaltoekenningen zonder tijdvertraging. VHDL kent een groot aantal logische en rekenkundige bewerkingen, waaronder nor, en elke VHDL-simulator zal die dus ondersteunen. Elke simulator zal ook een VHDL-beschrijving kunnen controleren op correctheid en eventueel foutmeldingen genereren. Dit gebeurt in de analyse-fase en is vergelijkbaar met de controles zoals compilers die uitvoeren. Na de analyse van de VHDL-beschrijving zal die, meestal samen met andere beschrijvingen uit libraries en packages, in een simulatieprogramma (als netlist gerealiseerd) moeten worden. Dit wordt de elaboratie-fase genoemd. Pas na deze elaboratie en een initialisatie van signalen en (eventuele) variabelen kan de daadwerkelijke simulatie worden uitgevoerd.

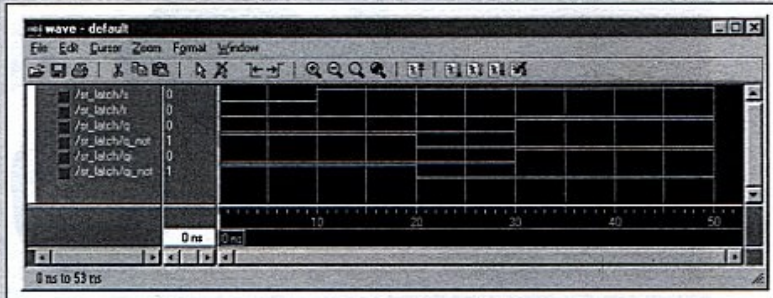
Bij echte parallelle processen, dus ook bij de echte hardware, gebeurt alles tegelijkertijd maar in een simulatie op een computer is ook de concurrency gesimuleerd. Dit gebeurt door de signaaltoekenningen pas te realiseren aan het eind van een simulatiestap. Een simulatiestap bestaat eigenlijk uit twee fasen. In de eerste fase worden alle processen uitgevoerd waarbij de waarden van de signalen niet veranderen. En in de tweede fase, nadat de processen klaar zijn met executeren, worden de nieuwe waarden voor de signalen daadwerkelijk toegekend (zie tabel 1). Werd dat niet zo gedaan, dan zou het bijvoorbeeld uitmaken of in de architectuur de eerste en de vierde regel (tussen begin en end) van plaats verwisseld waren en dat mag juist niet het geval zijn. De vertraging door dit ingebouwde uitstel van transacties wordt delta delay genoemd.

In de architecture van listing 2 is echter een expliciete vertraging aangegeven van 10 ns. Dat betekent dat die betreffende signaalveranderingen niet meteen nadat alle vier processen zijn uitgevoerd actief worden, maar nog eens 10 ns later, op de fictieve tijdsschaal die de simulator hanteert. In tabel 1 is aangegeven hoe de simulatie verloopt op een verandering van ingangspoort s van 0 naar 1, waarbij r de waarde 0 houdt.

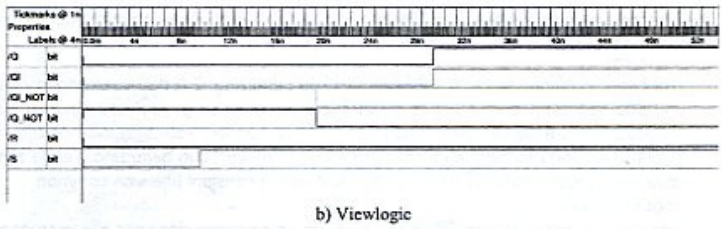
Op tijdstip 10 ns is de verandering van s is het enige event, in de VHDL-terminologie. De simulator bekijkt nu welke processen op een verandering van s wachten. Het blijkt dat alleen de nor-operatie van s met qi moet worden uitgevoerd. Deze operatie is echter ingeroosterd (scheduled) na de vertragingstijd vtd, die 10 ns bedraagt. De eerste simulatieslag verandert dus nog niets in de signaalwaarden. De verandering in qi_not na 10 ns is echter een event dat een tweede simulatieslag aanstuurt.. Dat betekent verandering voor q_not. Deze is zonder vertraging ingeroosterd, maar wordt pas gerealiseerd nadat ook de nor van qi_not met r is uitgevoerd. Deze veroorzaakt echter pas 10 ns later weer een nieuwe simulatiestap. Nu volgt q de verandering van qi. De nor van qi met s wordt ook in deze iteratie uitgevoerd maar veroorzaakt, bij realisatie 10 ns later, geen event omdat de waarde van qi_not blijft wat die was. De verandering van q vormt ook geen event, omdat geen van de vier processen door q wordt aangestuurd. De toestand van het systeem is stabiel geworden. De tijddiagrammen (waveforms) volgens twee daadwerkelijke simulaties met ver-

Tijd		10 ns		20 ns		30 nw		40 ns
Iteratie	0	1		2		3		(4)
Fase	Initieel	begin	eind	begin	eind	begin	eind	
Signaal		Signaal waarden						
S	0	*1	1	1	1	1	1	1
r	0	0	0	0	0	0	0	0
qi	0	0	0	0	0	*1	1	1
qi_not	1	1	1	*0	0	0	0	0
q	0	0	0	0	0	0	1	1
q_not	1	1	1	1	0	0	0	0

*Tabel 1. Het simulatieverloop van de sr-latch uit listing 1 op een verandering van s van 0 naar 1 waarbij r de waarde 0 houdt. De signaalveranderingen (events) die een volgende stap aansturen (triggeren) zijn aangegeven met *. De signaaltoekenning na 30 ns veroorzaakt geen event en de simulatie eindigt.*



a) Modelsim



b) Viewlogic

Fig. 1. Tijddiagrammen (waveforms) van de sr_latch uit listing 2 volgens simulaties met Modelsim respectievelijk Viewlogic (nu: Ennoveda). Het event dat de processen opstart (zie tabel 1) is het hoog worden van s na 10 ns.

schillende tools wordt gegeven in figuur 1.
Veranderingen in signalen moeten in VHDL-simulaties bij default gedurende de opgegeven vertragingstijd stabiel blijven. Zo niet dan worden zij door de simulator genegeerd. Deze instelling kan worden opgeheven met het sleutelwoord transport - dan worden dergelijke veranderingen in een wachtrij geplaatst en altijd doorgegeven. Men kan verder in één statement meer vertragingen tegelijk aangeven. Dit is een handig mechanisme om een waveform, bijvoorbeeld een kloksignaal, mee te definiëren.
VHDL heeft een zeer uitgebreid scala aan mogelijkheden om het gedrag van digitale schakelingen mee te kunnen simuleren. Een ontwerper kan bi-directionele poorten specificeren, standaardbibliotheken gebruiken, digitale schakelingen parametriseren en nog veel meer. Met behulp van de door de IEEE gedefinieerde std_logic_1164 package is het bijvoorbeeld mogelijk behalve '0' en '1' ook 'Z' (hoog impedant) en nog andere (zie listing 3) signaalwaarden te gebruiken. Met zogeheten resolved signals kan een ontwerper meer ingangen op een uitgang aansluiten en voor alle mogelijke waarden aangeven wat er dan gebeurt. Voor het type std_logic is hiervoor een resolutiefunctie voorgedefinieerd. Hiermee kan men eenvoudig een bus met meer drivers modelleren.

Synthese en Tools

De kracht van VHDL in combinatie met krachtige synthesetools is dat de ontwerper zich kan abstraheren van de realisatiedetails. Het gedrag van een systeem kan worden beschreven, zonder nog te weten hoe dit gesynthetiseerd moet worden. Vervolgens wordt de beschrijving aan de synthesetool aangeboden om omgezet te worden naar een realisatie (FPGA's / ASICs). Zo geeft figuur 2 het schema van de sr_latch uit listing 1, gesynthetiseerd met FPGA Express.
Een minder triviaal voorbeeld om de mogelijkheden van synthese te illustreren is het volgende probleem.
Stel dat men een systeem wil ontwerpen waarbij de ingang inp een bus is met breedte Lgt. De uitgang o is '1' indien twee of meer bits van inp '1' zijn en anders '0'. Verder wil men de breedte Lgt achteraf eenvoudig kunnen aanpassen. De entity-beschrijving van listing 4 geeft bovenstaande weer. Door de generic Lgt (zie deel 1 van dit artikel) van waarde te veranderen wordt de breedte van inp veranderd.
De klassieke digitale ontwerper kan dit niet zo gemakkelijk realiseren. Voor een kleine waarde van Lgt valt het nog wel mee, maar wat te doen als Lgt groot is en hoe maak je het ontwerp zodanig dat het eenvoudig is aan te passen voor een willekeurige afmeting van Lgt?
Het beschrijven van het gedrag van een dergelijk systeem is in VHDL echter zeer eenvoudig (zie de architecture in listing 4). Er wordt een functie gebruikt die, zoals de naam al suggereert, het aantal '1'-waarden in de input bepaalt. Verder is er slechts één concurrent statement waarmee wordt bepaald of het aantal '1'-en groter of gelijk is aan 2. Documentatie, een belangrijk onderdeel van elk ontwerp, krijgt men bijna cadeau.
Deze beschrijving kan gesynthetiseerd worden. Voor Lgt met waarde 4 is de realisatie weergegeven in fig. 3. De gebruikte symbolen hangen af van de gekozen technologie; in deze figuur zijn het logische poorten zoals nor, not etc. Het is duidelijk dat om optimaal gebruik te kunnen maken van VHDL van de ontwerper een mix van programmeren en inzicht in hardware-aspecten wordt verwacht.
In het algemeen geldt bijvoorbeeld dat hardware bepaalde fysische eigenschappen heeft die niet zomaar te veranderen zijn door de beschrijving aan te passen. Dat is het verschil met een softwareprogrammeertaal. Zo zijn de in een VHDL-ontwerp opgegeven vertragingstijden nooit zonder meer te implementeren.
Bepaalde VHDL-constructies zijn echter altijd synthetiseerbaar, vooral logische en rekenkundige operaties op eenvoudige standaardtypen (bijv. 32 -bits integer). Ook de al genoemde meerwaardige logica van IEEE Std. 1164-1193 wordt op dit moment algemeen ondersteund.
Het bedrijf Transfer Nederland BV specialiseert zich in consultancy en trainingen op het gebied van hardware-ontwerp en levert ook diverse VHDL-hulpmiddelen. Voor synthese zijn dat Innoveda FPGA Express, Synopsys FPGA Compiler II en Synopsys Design Compiler. Verder levert Transfer zes verschillende VHDL-simulatietools en nog een aantal andere pakketten voor hardware-ontwerp, bijvoorbeeld Innoveda Visual HDL en Statecad.

Tenslotte verzorgt de PATO (Post Academisch Technisch Onderwijs) jaarlijks in het najaar een VHDL cursus (zie www.pato.nl).
Met dank aan ir. E.Molenkamp voor zijn zeer gewaardeerde correcties en aanvullingen bij de totstandkoming van dit artikel. Eventuele onjuistheden blijven uiteraard geheel voor rekening van de auteur. ■

Referenties

1) Ashenden, J.P: The Designer's Guide to VHDL. Morgan Kaufmann Publishers (1996). ISBN 1-55860-270-4
2) Molenkamp, E.: VHDL, VHDL '87/93 en Voorbeelden (1997). ISBN 90-802634-3-5 (te bestellen via: www.cs.utwente.nl/~molenkam)

Trainingen en Tools op het gebied van HDL Transfer Nederland BV, Goorseweg 5, 7475 BB Markelo, tel.: (0547) 367367, e-mail: info@transfer.nl, web: www.transfer.nl.

Websites

http://tech-www.informatik.uni-hamburg.de/vhdl
www.cs.utwente.nl/~molenkam
www.eda.org
www.vhdl.org
www.ovi.org
www.pato.nl

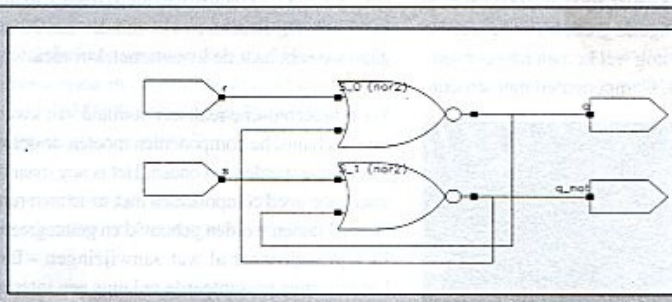


Fig. 2. Schema van sr_latch (uit listing 2) gesynthetiseerd met FPGA express.

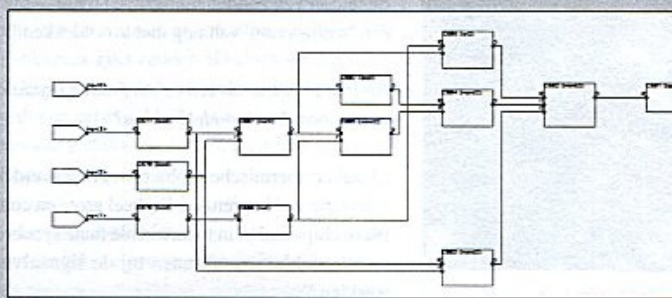


Fig. 3. Synthese van het systeem uit listing 4 met input busbreedte Lgt vastgesteld op 4. (bron: Molenkamp)