VHDL voor het voortgezet onderwijs

Een nieuwe lesmodule over hardware voor informatica

Ontwerp van Onderwijs
J.H. Rutgers

Begeleiders:
N.M. van Diepen
dr. J.T. van der Veen

Universiteit Twente

31 augustus 2009

Universiteit Twente
de ondernemende universiteit

Samenvatting

Informatica is een bijzonder vak in het voortgezet onderwijs. Het is een zeer breed (keuze)vak voor de bovenbouw havo en vwo, dat veel aspecten van de ict voorbij laat komen. In tegenstelling tot de studie informatica aan een vervolgopleiding, is het vak op de middelbare school niet neergezet als een technisch vak. Een belangrijk onderdeel dat (wellicht daardoor) ontbreekt, is lesstof over hardware. Bij informatica komen wel termen als 'randapparatuur' en 'processor' aan bod, maar stof over de werking van een chip, bussen, registers en firmware ontbreekt. Dit document beschrijft een nieuwe lesmodule voor informatica die precies die onderwerpen aanpakt.

Deze lesmodule is ontworpen voor de laatste klassen van het vwo. In acht tot tien lessen maken de leerlingen kennis met schakelingen van elementaire poorten, het ontwerp van een chip dat speciaal is gemaakt voor het spelen van een spel, het opstellen van een model van het spel met matrices, het omzetten van wiskundige formules voor het model van het spel naar concrete hardware, de implementatie van een chip met behulp van de taal VHDL, het aansturen van hardware vanuit software en de techniek van de fysieke chip, gerealiseerd in een halfgeleider. De focus ligt hierbij op VHDL, waarmee leerlingen daadwerkelijk een (stukje van een) chip implementeren. Het werk van alle leerlingen wordt gecombineerd tot een enkel spel, dat op een FPGA, een programmeerbare chip, gespeeld kan worden. Nieuw aan deze lesmodule is dat er een nieuw onderwerp binnen de informatica aan bod komt en dat het vakoverschrijdende karakter van informatica getoond wordt. Daarbij is het een pittig en uitdagend onderwerp, waar alle leerlingen intensief mee bezig kunnen gaan.

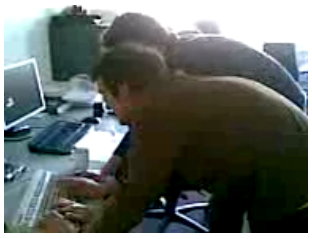
De lesmodule, inclusief opdrachten, presentaties, simulatieframeworks en ander ondersteunend materiaal, is ontwikkeld en getest in een 5 vwo-klas. Hieruit bleek dat men het interessant vond en veel had geleerd en dat men vond dat VHDL eigenlijk wel echt bij informatica hoort en dus onbreekt in de reguliere lesstof. Daarbij wilden de leerlingen ook wel meer lessen hieraan besteden, in plaats van de beschikbare acht lessen.

Dit pakket kan worden aangeboden zoals nu is gepresenteerd, maar het biedt veel mogelijkheden tot uitbreiding. De lesstof biedt vele handvatten tot verdieping in specifieke onderwerpen. Ook kan er vakoverschrijdend gewerkt worden, door bijvoorbeeld de matrices, elektriciteit of de halfgeleider uit te besteden aan respectievelijk wiskunde, natuurkunde en scheikunde. Ook leent de stof zich voor het profielwerkstuk, waarin leerlingen zich zelfstandig kunnen verdiepen in de materie.

Voorwoord

Toen ik tijdens mijn master Embedded Systems begon aan Computer Science Education, viel me al snel op dat hardware op de middelbare school onderbelicht wordt. En dat terwijl hardware toch een essentieel onderdeel is van een computersysteem. Natuurlijk kun je lang discussiëren over de vraag of het apparaat wel van belang is als het gaat om algoritmes en software—er wordt bij software engineering altijd geabstraheerd van de hardware—maar een feit blijft dat het apparaat in de praktijk een rol speelt. En informatica is, vanuit leerlingperspectief, gewoon niet compleet zonder de machine besproken te hebben.

Met die insteek ben ik begonnen aan het ontwerpen van een lesmodule dat zich richt op de techniek, chips en VHDL in het bijzonder. Maar dan niet met een fancy simulator of abstracte tekeningen en modellen, maar gewoon hoe het apparaat werkt en hoe het in de praktijk gemaakt wordt. Dus we hebben ModelSim, een FPGA, VHDL en geen afgeslankte speelgoedversies ervan, met als resultaat een lesmodule op theoretisch hoog niveau en veel uitdaging; iets waar leerlingen zich echt op kunnen uitleven.



Ik wilde een spel maken, waaraan leerlingen iets konden programmeren en toevoegen. Het spel moest wel dusdanig leuk en uitdagend zijn, dat zelfs leerlingen die hardware helemaal niet interessant vinden, tenminste geprikkeld worden door het spel. Tijdens de ontwikkeling van het spel, wat ik grotendeels op de Universiteit Twente heb gedaan, heb ik in ieder geval al mijn kamergenoten kunnen prikkelen. Testversies van het spel die al iets van ruimteschepen lieten zien met halfwerkende zwaartekracht en beperkte mogelijkheid tot schieten, waren voor mijn omgeving al leuk genoeg om veelvuldig zich ongevraagd aan te melden als vrijwilliger om het spel uit te proberen...

Ik denk dat het resultaat leerlingen een goed beeld geeft van hardware en een leuke introductie is voor het voortgezet onderwijs. En het biedt genoeg mogelijkheden om in de toekomst gebruikt te kunnen worden als uitgebreider lesmateriaal of profielwerkstukken.

Jochem Rutgers
Enschede, juli 2009

Inhoudsopgave

Lijst met afkortingen

Hardware

ALU Arithmetic logic unit

ASIC Application-specific integrated circuit

FPGA Field Programmable Gate Array

GPP General Purpose Processor

HDL Hardware Description Language

led Light-emitting diode

LUT Look-up table

VGA Video Graphics Array

VHDL Very High Speed Integrated Circuit Hardware Description Language

VHDL QRC VHDL Quick Reference Card

Onderwijs

C&M Cultuur & Maatschappij

cse centraal schriftelijk eindexamen

E&M Economie & Maatschappij

ict Informatie- en communicatietechnologie

N&G Natuur & Gezondheid

N&T Natuur & Techniek

olg onderwijsleergesprek

slu studielastuur

vo voortgezet onderwijs

Een

Introductie

Informatica is een bijzonder vak. Dit geldt niet alleen voor de studie en het vakgebied, maar zeker ook voor het schoolvak in de tweede fase. Met een grote variëteit aan leerlingen die besluiten om in de bovenbouw het vak te kiezen, ontstaat er een melange aan interesses en verwachtingen in de klas. Het examenprogramma van informatica voorziet in een breed scala aan onderwerpen om de leerlingen een gedegen basis in computerkennis en -vaardigheden te geven voor de huidige informatiemaatschappij. Dit is meer dan gewoon het leren omgaan met een tekstverwerker; het gaat om inzicht krijgen in het ontwerp van de machine, de software, protocollen, netwerken, et cetera.

Hoewel het vak erg breed is, ontbreekt er één onderwerp: de hardware. Er wordt slechts oppervlakkig aandacht besteed aan de werking van de machine, in tegenstelling tot de hoeveelheid tijd en diepgang van bijvoorbeeld het onderdeel programmeren. In dit project is een lesmodule ontwikkeld om de kennis van hardware te verbeteren. De lesmodule is gericht op (eind) 5 en 6 vwo. Het doel is niet om alle details van de hardware te laten zien, leerlingen moeten een introductie krijgen op een andere wereld dan wat ze tot nu toe hebben gezien bij informatica.

Tijdens de lesmodule wordt gewerkt aan een spel. Dit spel is niet gemaakt in software, op een wijze zoals al is behandeld bij het onderdeel software engineering, maar in hardware; er wordt gewerkt aan een chip dat het spel uitvoert. Het ontwerp van de chip is modulair. De leerlingen in de klas kunnen in tweetallen werken aan een eigen module van het spel. Alle modules, en het framework dat wordt aangeleverd, vormen samen één chipontwerp. Dit ontwerp kan worden geprogrammeerd in speciale hardware, zodat het ontwerp daadwerkelijk kan worden getest en het spel dus kan worden gespeeld.

Tijdens de lessen worden de leerlingen in vogelvlucht het proces van het ontwerpen van een chip getoond. Zo komt het wiskundige model van het spel aan bod, het ontwerpen van de chip, het daadwerkelijk 'programmeren' van de chip en hoe een chip in silicium uiteindelijk wordt gemaakt.

Dit document is als volgt gestructureerd. De rest van dit hoofdstuk geeft een overzicht van literatuur aangaande het ontwikkelen van een lesmodule in het algemeen. Hoofdstuk 2 gaat in op wat het vak informatica eigenlijk inhoudt en waar in het programma er wat schort en waar het onderdeel hardware goed ingepast zou kunnen worden. Op basis van deze analyse wordt in paragraaf 2.5 een lijst met doelen opgesteld op basis waarvan de lesmodule is gestoeld. De ontwikkelde code en gebruikte hardware is beschreven in hoofdstuk 3, waarna in hoofdstuk 4 die software en hardware gebruikt wordt in een lessenserie. De lessenserie is geëvalueerd en de resultaten van de expert review en leerlingenenquête en de eigen ervaringen worden beschreven in hoofdstuk 5. Tot slot worden er conclusies getrokken betreffende het project in

hoofdstuk 6 en worden er in paragraaf 6.1 suggesties gedaan tot aanpassing van het lespakket voor een iets ander gebruik dan beschreven in de doelen, mocht daar in de toekomst behoefte aan zijn.

1.1 Literatuur

Dit project omvat het ontwerpen van een onderwijsmodule. Hiervoor zijn vele boeken geschreven en er zijn veel documenten met regels en richtlijnen van de overheid als het gaat om onderwijs. Deze paragraaf geeft een overzicht van de belangrijkste literatuur dat betrekking heeft op dit project.

Informatica is een keuzevak voor de bovenbouw havo/vwo in de tweede fase. Door de overheid is er een redelijk globaal examenprogramma [1] opgelegd. De Handreiking [12] geeft een mogelijke concrete invulling van dit programma. In het Vakdossier [11] worden de resultaten van uitgebreid onderzoek besproken betreffende de uitvoering van en verbeterpunten voor het vak. In het informaticaonderwijs worden drie methoden gebruikt: Enigma (Thieme Meulenhoff) [6], Informatica Actief (Edu'Actief) en Fundament (Instruct).

De lesmodule die is ontwikkeld voor dit project, focust zich voor het belangrijkste deel op het kennismaken en werken met een nieuwe programmeertaal, namelijk VHDL. Aan de didactische aspecten aan programmeeronderwijs is al veel meer aandacht besteed.

Williams et al. [14] beargumenteren waarom in het programmeeronderwijs in tweetallen, *Pair Programming* genaamd, gewerkt moet worden. Uit onderzoek blijkt dat wanneer twee personen achter een pc werken, waarbij de een de code typt en de ander als navigator hem controleert en aanstuurt, er een hogere productiviteit behaald wordt en de cijfers gemiddeld hoger zijn dan wanneer leerlingen zelfstandig werken. Doordat men samenwerkt, wordt er meer gediscussieerd over de stof, lossen ze problemen vaker zelf op en wordt er dus minder aanspraak gemaakt op de docent. De docent speelt een belangrijke rol in deze werkvorm. De docent moet er sterk op aansturen dat men samenwerkt, anders kiezen leerlingen ervoor zelfstandig te werken. Daarnaast is het essentieel voor het begrip dat de rollen regelmatig worden omgedraaid, waar de docent goed op moet letten dat dit gebeurt.

Pair Programming is door verschillende instellingen getest en geïmplementeerd, waaronder door Jacobson en Schaefer [8]. In dat paper wordt bevestigd dat de docent groepen moet forceren om te wisselen van rollen wil Pair Programming effectief zijn. Tevens worden een aantal vooroordelen van deze aanpak tegengesproken: vervolgvakken waarin studenten weer zelfstandig moeten werken, hebben geen last van het feit dat ze hebben leren programmeren in tweetallen; en de teamsamenstelling is geen probleem, verreweg de meeste teams die al dan niet willekeurig zijn samengesteld, bestaan uit studenten die op een 'compatible-manier' werken.

Behalve de werkvorm tijdens de lessen, is de vorm waarin een programmeertaal wordt aangeboden ook van belang. Het programmeren van spellen in de nieuwe programmeertaal is daarbinnen een populaire vorm. Zoals Bayliss en Strout [4] beschrijven, zijn er verschillende taken binnen een spel waar leerlingen aan kunnen werken. Deze taken kunnen specifieke syntaxis en structuren van een programmeertaal laten zien, zoals objecten en overerving in een spel bij object-georiënteerde talen, iteraties van de *game loop*, expressies om scores te berekenen of specifieke datastructuren voor efficiëntie in een spel. Een mogelijk spel is Robocode, waarin leerlingen eigen robots programmeren die autonoom in een virtuele arena elkaar kapot moeten schieten [9].

In plaats van om een spel te maken, kunnen leerlingen ook leren programmeren door het *spelen* van een spel. Squire en Jenkins [13] beschrijven dat bijvoorbeeld het spelen van Civilization III leerlingen inzicht kan geven in begrippen als monotheïsme en monarchie. Een dergelijke methode kan worden toegepast op het leren programmeren, maar dat lijkt veel minder natuurlijk dan het bovengenoemde voorbeeld.

Binnen programmeeronderwijs kan er onderscheid gemaakt worden in het doel van het leren programmeren [10]. Leren programmeren kan betekenen dat de leerling in staat moet zijn om eigenschappen van geschreven programma's wiskundig te kunnen bewijzen. Tekstboeken beschrijven daarentegen voornamelijk de syntaxis van de te leren taal. Anderen vinden dat de focus moet liggen op vaardigheden als probleemoplossen en requirementsanalyse.

Eckerdal et al. [5] onderscheiden vijf begripniveaus die leerlingen kunnen bereiken door programmeeronderwijs. In het onderzoek is met behulp van interviews na afloop van het eerste programmeervak studenten gevraagd wat ze denken dat 'leren programmeren' betekent. Deze niveaus zijn, oplopend in begrip, waarbij een volgend niveau de vorige bevat: 'leren programmeren' is het begrijpen en kunnen gebruiken van een programmeertaal; 'leren programmeren' is ook een manier van denken, maar wat die manier precies is, blijft onduidelijk; daarbij is het ook duidelijk hoe de computer werkt en hoe programma's in het dagelijks leven zich manifesteren; tevens is het een manier en methode van denken, namelijk probleemoplossend; en het probleemoplossend vermogen dat je leert door te leren programmeren is ook toepasbaar in een andere context dan programmeren. Het ultieme doel is om leerlingen in het laatste niveau te krijgen, maar zoals al eerder is beschreven, beperken tekstboeken zich vaak tot slechts het eerste niveau [10]. Het is binnen het programmeeronderwijs van belang dat leerlingen tenminste het vierde niveau bereiken [5].

Er moet worden opgemerkt dat hoewel er veel aandacht is voor programmeeronderwijs, alle onderzoeken zich richten op *software* engineering. Hoewel het maken van software en hardware wat betreft conceptvorming, het aanleren van een nieuwe programmeertaal en didactische aanpak sterk op elkaar lijken, lijkt er geen poging gedaan te zijn om VHDL, Verilog of een andere Hardware Description Language (HDL) in het voortgezet onderwijs (vo) te behandelen.

Twee

Analyse

Zoals in hoofdstuk 1 is beschreven, heeft dit project ten doel om een lesmodule over hardware te ontwikkelen. Dit hoofdstuk zal het vak informatica beschrijven en de context van de lesmodule aangeven. Paragraaf 2.1 beschrijft het vak informatica in het algemeen en paragraaf 2.2 onderzoekt waar in het vak de module of hardware past. Gegeven de context van de lesmodule beschrijft paragraaf 2.3 hoe de lesmodule in grote lijnen gegeven kan worden. Vervolgens wordt de doelgroep beschreven in paragraaf 2.4. De te behalen doelen, de te gebruiken lesmethoden en eventuele beoordeling worden beschreven in respectievelijk paragrafen 2.5 tot en met 2.7. De eisen voor het te gebruiken spel zijn geformuleerd in paragraaf 2.8.

2.1 Informatica als vak

Informatica is tegelijk ingevoerd met de tweede fase. Het is een keuzevak voor de bovenbouw havo/vwo. Sinds 2007 is het aantal studielastuur (slu) verhoogd van 240 voor havo en 280 voor vwo tot respectievelijk 320 en 440. Hierdoor is informatica qua omvang even groot als andere vakken; het beeld dat 'informatica een klein vak [is] in de marge van de tweede fase', is onterecht [11]. Informatica is tevens een profielkeuzevak geworden voor Natuur & Techniek (N&T).

Niet iedere school biedt informatica aan. Tabel 2.1 toont het aantal scholen dat informatica aanbiedt [11]. De tabel laat zien dat sinds het begin van de tweede fase ongeveer 60% van de scholen informatica in het programma heeft.

Informatica is een breed vak. In het examenprogramma [1] is een breed scala aan onderwerpen opgenomen. Deze onderwerpen en een suggestie voor de urenverdeling is gegeven in de Handreiking [12] en is samengevat in tabel 2.2. De zwaargewichten binnen het vak zijn software (domein B3), informatiesystemen (C3–6,9), relationele

	2002	2003	2004	2005	2006
Aantal schoolinstellingen	479	478	476	474	464
Aantal instellingen dat informatica aanbiedt	279	289	291	289	283
Percentage instellingen dat informatica aanbiedt	58,2%	60,5%	61,1%	60,0%	61,0%

Tabel 2.1: Aanbod informatica

domein	omschrijving	havo (slu)	vwo (slu)
A	Informatica in perspectief	10	10
B1	Gegevensrepresentatie	8	8
B2	Hardware	7	7
B3	Software	40	70–100
B4	Organisaties	5	5
C1	Communicatie en netwerken	10	10
C2	Besturingssystemen	10	10
C3-6,9	Informatiesystemen	70	70–100
C7	Relationele databases	20	50
C8	Interactie mens-machine	20	20
D,vrij	Toepassingen en samenhang	120	150
totaal		320	440

Tabel 2.2: Samenvatting onderwerpen binnen informatica en aantal slu volgens Handreiking

databases (C7) en een vrij in te vullen projectruimte (D). Geel [7] laat zien dat de daadwerkelijk hoeveelheid bestede tijd op school, nog vóór de urenuitbreiding van 2007 waarin de richtlijn nog 35 uur bedraagt, aan het onderdeel software (B3) varieert tussen 10 en 190 slu, met een gemiddelde van 45 uur.

Opvallend is de beschikbare tijd voor hardware. Domein B2 gaat slechts over het kunnen aanwijzen van hardwareonderdelen, zoals een processor en een printer. Een simpel logisch circuit, zoals een opteller, staat niet in het examenprogramma, maar Enigma [6] besteedt hier aandacht aan met de tool MMLogic[1]. Dit beperkt zich tot spelenderwijs ontdekken hoe een EN- en OF-poort werkt.

In tegenstelling tot de vervolgopleiding informatica en aanverwanten, is het vak op het vo geen technisch vak. Dit betekent dat de techniek wel wordt behandeld, maar de nadruk daar niet op ligt. Het vak moet toegankelijk zijn voor de leerlingen van alle profielen. Daardoor is er geen voorkennis vereist van bijvoorbeeld wiskunde of natuurkunde (bijvoorbeeld over het onderdeel elektriciteit). Dit neemt niet weg dat leerlingen uit de technische profielen het vak vaker kiezen dan uit de niet-technische. Tabel 2.3 toont een overzicht van het percentage leerlingen per profiel dat informatica kiest en de herkomst van alle de leerlingen die informatica hebben gekozen [11]. Uit de tabel is af te lezen dat ruim een kwart van de havo-leerlingen uit het profiel N&T informatica kiest, ten opzichte van slechts 8,7% van de Economie & Maatschappij (E&M)-leerlingen. Echter, aangezien E&M door veel meer leerlingen wordt gekozen, bestaat een derde van de gemiddelde informaticaklas uit E&M-leerlingen.

Er is geen centraal schriftelijk eindexamen (cse) voor informatica, maar er zijn wel argumenten om dit in de toekomst in te voeren [11]. Onder meer wordt het instellen van een cse gezien als mogelijke oplossing om het vak volwaardig(er) te maken en een manier om niveau- en inhoudsverschillen tussen scholen te verminderen. Zolang er geen cse is, kan er door de docent nog geschoven worden met de verdeling van de uren. Om de nieuwe hardwaremodule van dit project in het bestaande programma te passen, kunnen bijvoorbeeld uren van domein D en de vrij te vullen uren worden gebruikt.

[1] Zie www.softronix.com/logic.html

	profiel	leerlingen per profiel heeft deelgenomen aan informaticaexamen	herkomst examenkandidaten informatica
havo[a]	N&T	28,4%	25,8%
	N&G	12,3%	22,7%
	E&M	8,7%	33,5%
	C&M	4,2%	15,5%
	N&T/N&G	8,8%	2,1%
	E&M/C&M	2,1%	0,3%
vwo[b]	N&T	25,3%	29,0%
	N&G	9,2%	27,2%
	E&M	10,1%	31,2%
	C&M	4,0%	7,9%
	N&T/N&G	9,7%	4,6%
	E&M/C&M	1,4%	0,1%

[a] Aantal deelnemers aan het havo-schoolexamen informatica: 4051 (9,3%)

[b] Aantal deelnemers aan het vwo-schoolexamen informatica: 3322 (10,3%)

Tabel 2.3: Verdeling profielen leerlingen informatica in 2006

2.2 Hardware in het examenprogramma

In de vorige paragraaf is de huidige situatie van informatica besproken. Er is geconstateerd dat hardware (te) weinig aandacht krijgt, maar er in het programma wel ruimte is om hier meer aandacht aan te besteden.

Maar, wat is het vak informatica eigenlijk? Een legitieme vraag, maar met een lastig antwoord, hoewel het informaticaonderwijs op het antwoord gestoeld zou moeten zijn. In het examenprogramma, zoals bepaald in 1995, is de doelstelling geformuleerd als [11]:

> ...dat leerlingen zich een beeld vormen van informatica en ict en de wisselwerking van het vak met de maatschappij, andere vakgebieden en technologie, dat leerlingen zich oriënteren op de rol van informatica en ict in studie en beroep en dat de leerlingen zich basisbegrippen en vaardigheden van het vak eigen maken, informatievraagstukken bestuderen, structuren van gegevensverwerkende systemen bestuderen en een systeemontwikkeltraject leren doorlopen.
>
> *Doelstelling examenprogramma*

In het examenprogramma wordt vervolgens over hardware het volgende vermeld [1]:

> **Subdomein B1: Gegevensrepresentatie in een computer**
> De kandidaat kan gangbare digitale coderingen van gegevens beschrijven en toepassen.
>
> **Subdomein B2: Hardware**
> De kandidaat kan de functies van een computer benoemen, aangeven welke hardware en bijbehorende gangbare randapparatuur deze functies uitvoeren en de wisselwerking tussen deze functies beschrijven.
>
> *Examenprogramma*

Wikipedia vermeldt bij het lemma Computer science het volgende:

> Computer science (or computing science) is the study of the theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems.
>
> *Wikipedia*

In de 'practical techniques' lijkt hardware wel een rol te spelen, echter dezelfde pagina vermeldt tevens het volgende: "The design and deployment of computers and computer systems is generally considered the province of disciplines other than computer science. For example, the study of computer hardware is usually considered part of computer engineering." Echter, 'computer engineering' vertaalt naar 'technische informatica' of 'computertechniek' en gaat meer richting elektrotechniek. In deze lijn ligt ook de uitspraak van Edsger W. Dijkstra:

> Computer science is no more about computers than astronomy is about telescopes.
>
> *Edsger W. Dijkstra*

Het Enigma informatieboek beschrijft in het eerste hoofdstuk het vak als volgt [6]:

> Het is lastig om [van informatica] een goede definitie van te geven, maar het komt erop neer dat informatica de leer is van informatieverwerkende systemen. Dergelijke systemen verwerken gegevens die van buiten het systeem komen: de invoer. Het resultaat van de verwerking wordt gepresenteerd aan de gebruiker of een ander systeem: de uitvoer. Vrijwel alle systemen kunnen de informatie kortere of langere tijd bewaren in een geheugen.
>
> *Enigma*

Samengevat, er is geen sluitende definitie wat het vak informatica in zou moeten houden. In de doelstelling van het vak worden 'systeemverwerkende systemen' bestudeerd, waar hardware ook in valt. Daarnaast is hardware van belang in studie en beroep, waar leerlingen zich in het vak op moeten oriënteren. Echter, het examenprogramma hecht blijkbaar minder waarde aan de echte techniek.

Hoewel de 'echte' informaticawetenschap een onderscheid maakt tussen de theoretische informatica en de praktische implementatie, ligt het voor de hand om te zeggen dat hardware *als basiskennis* binnen informatica wel degelijk van belang is. Op basis hiervan is het gerechtvaardigd om binnen het schoolvak informatica meer aandacht te geven aan de technische zaken omtrent het ontwikkelen van chips. Zonder kennis van het platform is de manier van programmeren ervoor of de reden van de manier van het werken ermee veel ondoorzichtelijker.

2.3 Aanpak

Deze paragraaf zal een globaal overzicht geven van de inhoud van de lesmodule en de vormgeving van het onderwijs. Deze onderwerpen zullen in respectievelijk in paragraaf 2.5 en hoofdstuk 4 in detail aan bod komen.

Het centrale motto van de lesmodule is:

> De leerling moet inzicht krijgen in de precieze werking van hardware en de relatie tussen hardware, software en andere vakgebieden die voor de leerling op het eerste gezicht niets met informatica te maken lijken te hebben.

Voorafgaand aan het project zijn een aantal eisen opgelegd aan de inhoud van de lesmodule. Dit heeft geleid tot vaststelling van de de volgende drie onderwerpen binnen de lesmodule, waarvoor in totaal ongeveer tien lessen beschikbaar zijn:

- het wiskundig modelleren van de functionaliteit van de te bouwen chip;
- het implementeren van de chip met behulp van de taal VHDL;
- inzicht geven in de technische werking van een chip in het algemeen.

De bovenstaande onderwerpen zijn vrij theoretisch geformuleerd. Om de lessen smeuïg te maken, worden de onderwerpen behandeld aan de hand van een te bouwen spel. In hoofdstuk 1 is al beschreven dat het maken van een spel op zich uitdagend is wanneer het gaat om het programmeeronderwijs [4]. Dit spel en de drie onderwerpen bepalen de globale structuur en het verloop van de lessenserie.

2.3.1 Het model

De lessenserie begint met het bepalen van het wiskundige model van het spel. De wiskunde moet voor de leerlingen uitdagend en nieuw zijn. Echter, het doel is niet om tijdens de informaticalessen de leerlingen nieuwe wiskunde aan te leren. Mede gelet op het feit dat lang niet iedereen wiskundig onderlegd is door de profielkeuze, zal het model en de bijbehorende wiskunde voornamelijk worden gepresenteerd aan de leerlingen. Wat van belang is, is dat men inziet hoe de ingewikkelde formules tot stand komen en hoe de abstracte wiskunde vertaald kan worden naar de concrete praktijk bij informatica.

De gekozen werkvorm hiervoor is klassikaal les, onderwijsleergesprekken, met individuele opdrachten tussendoor. Hierbij is overleg tussen leerlingen en het uitwisselen van ideeën cruciaal voor het begrip van de stof. Aangezien het wiskundeniveau sterk verschilt onderling, is het waardevol om mensen met elkaar te laten overleggen om misschien tot creatieve oplossingen te komen. De opdrachten zijn qua hoeveelheid stof eenvoudig, maar moeten de werking van de wiskunde inzichtelijk maken.

2.3.2 Implementatie

Op basis van het model kan de chip ontworpen worden. De vertaling van wiskunde naar hardware is puur informatief, zonder de bedoeling dat men het zelf ook kan. Het leren hardware ontwerpen kost veel tijd, iets waarvoor geen tijd beschikbaar is in deze module. Tot zover is het te ontwerpen spel op een hoog niveau bekeken. Vanaf

dit moment wordt het hoge, abstracte niveau verlaten en richten de lessen zich op de implementatiedetails.

De implementatie wordt gedaan met VHDL. Deze taal wordt in de (voornamelijk Europese) industrie gebruikt. De leerlingen krijgen hierdoor een concreet idee hoe de computerhardware gemaakt wordt. Het is niet de bedoeling om VHDL-programmeurs op te leiden. Wederom is het van belang dat men een concreet beeld krijgt van hoe hardware, en een chip specifiek, gebouwd wordt. De leerlingen kunnen met VHDL een afgebakend onderdeel van het totale spel bouwen. De gezamenlijke stukken VHDL-code van de klas kunnen worden gecombineerd tot het uiteindelijke spel.

Wanneer de tijd het toelaat, kan tevens softwareontwikkeling worden toegevoegd. Wanneer een stuk hardware is beschreven met VHDL, dan kan een microprocessor hiermee communiceren. Hiertoe kunnen leerlingen, wellicht met behulp van een programmeertaal die ze al kennen, software maken die over bijvoorbeeld een bus registers kan lezen en schrijven. Hierdoor wordt de relatie tussen software en hardware duidelijk gemaakt.

De werkvorm is pair programming [14]. Programmeren is een praktische bezigheid. Klassikaal lesgeven is minder effectief, men moet gewoon de code intypen en uitproberen om te leren programmeren en pair programming is daarvoor een geschikte manier [8]. Hiervoor is een simulatieomgeving beschikbaar, waarin de leerlingen hun code kunnen uitproberen en fouten kunnen opsporen.

De aandacht ligt hier niet bij het leren werken met de taal VHDL. Het is veel belangrijker dan men een idee krijgt hoe je een chip zou kunnen maken. In het model van Eckerdal et al. [5], hoeven leerlingen dus niet alle details te kennen van de taal (eerste niveau), maar wordt er meer gericht op het derde niveau: de leerling moet snappen hoe een computer, telefoon, of andere embedded systems in hun omgeving gemaakt worden en kunnen werken. Het vierde niveau, waarin de leerlingen daadwerkelijk een probleemoplossend vermogen kweken met de nieuwe programmeertaal, is te hoog gegrepen. Hiervoor is lang niet genoeg tijd en het is niet de opzet van de beknopte lesmodule.

Dit onderdeel zal het grootste gedeelte van de tijd in beslag nemen. Iedere vorm van toetsing moet zich toeleggen op het testen van de kennis opgedaan bij het werken met VHDL.

2.3.3 Integratie andere vakgebieden

Na het experimenteren met VHDL wordt de lesmodule afgerond met een presentatie over de natuurkundige en scheikundige aspecten van een chip, betreffende stroomgebruik en halfgeleiders. Dit zal inzicht geven in de daadwerkelijke opbouw van hardware, iets waar het boek doorgaans blijft steken op een abstract niveau als ‘bits’, ‘een processor die instructies uitvoert’, ‘een videokaart dat beeld maakt’.

Dit is een kort onderdeel en kan goed gedekt worden in een informatieve presentatie in de les. Het is niet erg, maar er moet niet naar gestreefd worden, dat het voor sommige leerlingen te hoog gegrepen zal zijn. Wanneer leerlingen bijvoorbeeld ook scheikunde of natuurkunde hebben, dan zullen opmerkingen over halfgeleider beter aankomen dan wanneer dat niet het geval is.

2.4 Doelgroep

Een nieuwe programmeertaal aanleren is geen sinecure. Daarnaast zijn in het voorstel van aanpak van de vorige paragraaf een aantal wensen geformuleerd die veel

verwachten van leerlingen. Deze houden in dat men al kan programmeren, want de relatie software-hardware moet duidelijk worden; de beschikbare tijd is beperkt, dus er wordt verwacht dat de basiskennis over bits en hardware al aanwezig is; integratie van verschillende vakgebieden is alleen mogelijk wanneer er voldoende kennis al is opgedaan in die andere vakgebieden; in korte tijd moet men in staat zijn om deze volledig nieuwe en complexe stof zich eigen te maken.

De opzet is niet dat er procedurele kennis wordt overgedragen, maar dat men zelfstandig de stof probeert te doorgronden op basis van al eerder opgedane kennis. Op basis van deze verwachtingen moet geconcludeerd worden dat deze lesmodule alleen geschikt is voor 6 vwo en het einde van 5 vwo. De grootte van de klas is in principe alleen beperkt door praktische zaken als de werkdruk van de docent en de beschikbaarheid van computers.

2.5 Lesdoelen

Op basis van de voorgaande analyse is de onderstaande lijst met doelen opgesteld, welke aan het einde van de lessenserie gehaald moeten zijn.

1. De leerlingen moeten kennis hebben van het ontwerpproces: model-ontwerp-implementatie-realisatie.

2. Leerlingen moeten inzien dat een General Purpose Processor (GPP) in sommige situaties niet de juiste chip is om een doel te bereiken.

3. Gegeven een applicatie, in dit geval het spel, moeten leerlingen een idee hebben hoe er een model gemaakt kan worden. Het daadwerkelijk opstellen van wiskundige vergelijkingen is niet nodig.

4. De leerlingen moeten de relatie inzien tussen wiskundige formules en een (mogelijke) implementatie in hardware of software. De leerlingen hoeven niet zelf het model om te kunnen zetten in een chipontwerp.

5. De leerlingen moeten de basisstructuren van VHDL kennen en kunnen toepassen om simpele hardware te maken. De basisstructuren worden beperkt tot: logische operaties op bits (`std_logic`) en vectoren (`std_logic_vector`); geheugenelementen; een `process` met daarin een `if-else-end if`-statement.

6. De leerlingen moeten inzien hoe de relatie tussen software—assembly op een microprocessor—en hardware—eigengeschreven VHDL-code—is.

7. Het moet de leerlingen duidelijk zijn dat er een sterke relatie is tussen verschillende vakgebieden: wiskunde voor het model, informatica en elektrotechniek voor de implementatie, natuurkunde en scheikunde voor de realisatie.

2.6 Lesmethoden

Op basis van de voorgestelde aanpak en te bereiken doelen in paragrafen 2.3 en 2.5, kunnen de onderstaande werkvormen worden bepaald.

- **Introductie**: De aanleiding en achtergrond van de lesmodule wordt klassikaal gegeven als presentatie.

- **Model**: Het model is gebaseerd op voor de leerlingen nieuwe wiskunde. Door de docent worden de klassikaal leerlingen door de theorie geleid, waarin door middel van een onderwijsleergesprek (olg) leerlingen geprikkeld worden na te denken over hoe het model precies geformuleerd kan worden. De leerlingen kunnen individueel, maar overleg wordt sterk gestimuleerd, eenvoudige opdrachten waarin de wiskunde de hoofdrol speelt.

- **Implementatie**: Er zijn een aantal opdrachten met VHDL. De introductieopdracht moet individueel worden gemaakt, zodat iedereen zelf kennis kan maken met de simulatieomgeving. Het daadwerkelijk implementeren van de modules van het spel gebeurt in tweetallen (pair programming). Ieder team implementeert een eigen module, die samengevoegd kunnen worden tot het hele spel. De lesstof wordt schriftelijk aangeboden en leerlingen moeten zelfstandig (samen met de partner) zich VHDL eigen maken en de opdracht maken. De docent geeft in principe geen klassikale instructie over VHDL, maar is beschikbaar om vragen te beantwoorden en mensen op weg te helpen.

- **Realisatie**: Synthese, placement en routing voor Field Programmable Gate Array (FPGA) en de FPGA zelf worden niet behandeld. Als leerlingen hun code willen testen, dan programmeert de docent de FPGA, zonder tussenkomst of hulp van de leerling. Als afsluiting wordt een presentatie gegeven over onder andere CMOS-technologie en halfgeleiders, zonder benodigde interactie met leerlingen.

2.7 Beoordeling en controle behalen doelen

De lesmodule heeft een informatief en praktisch karakter. Hoewel er tamelijk veel theorie in wordt gepresenteerd, wordt er van leerlingen niet verwacht dat ze deze zelf kunnen reproduceren. Het is voornamelijk van belang dat men kennis maakt met een andere kant van informatica. Aangezien de beschikbare tijd beperkt is, is er niet genoeg tijd om bijvoorbeeld de wiskunde, modellering of chipontwerp genoeg te oefenen.

Mocht er een cijfer gekoppeld moeten worden aan de prestaties van leerlingen, dan moet voornamelijk de inzet en de kennis van VHDL beoordeeld worden. Echter, het aantal lessen is beperkt, dus ook de voortgang in VHDL zal beperkt zijn; de verwachting is niet dat in slechts enkele lessen een module voor het spel volledig wordt afgerond. Daarom kan alleen de praktische vooruitgang in VHDL becijferd worden, waarbij de inzet in de les een belangrijke rol speelt.

Eventueel is het mogelijk om met een diagnostische toets de vergaarde kennis te peilen, maar aangezien er alleen wordt verwacht dat men kennis heeft gemaakt met een nieuw onderwerp en er geen precieze feiten geleerd hoeven worden, is een toets lastig te geven. Ook een toets over geleerde VHDL-constructies is moeilijk te verantwoorden, aangezien de groepen verschillende opdrachten maken en niet iedere constructie gelijkwaardig terugkomt in ieder onderdeel. Verschillende groepen zullen verschillende problemen tegengekomen zijn en er is niet genoeg tijd om ieder onderwerp uitputtelijk door te nemen. Een enkele toets zal dus geen juiste afspiegeling kunnen geven van de vooruitgang in kennis en kan niet gebruikt worden als beoordeling. De docent zal iedere groep individueel moeten beschouwen.

Aangezien er maar een beperkte tijd is voor de lesmodule, speelt het bovenbeschreven probleem dat er moeilijk getoetst kan worden. Wanneer er meer tijd is, dan

kan er van leerlingen verwacht worden dat ze meer hebben gepresteerd en hebben begrepen van de stof. In dat geval kan de kennis over VHDL wel getoetst worden en kunnen leerlingen daarop worden afgerekend. Deze toets kan eenvoudig worden vormgegeven door iedere groep mondeling te laten toelichten wat ze hebben gemaakt en hoe het werkt. Op basis hiervan kan de docent inschatten in hoeverre ze de stof hebben begrepen.

2.8 Eisen aan spel

Het te kiezen spel waar men in de les aan kan werken, moet voldoen aan de onderstaande eisen.

- Het spel moet eenvoudig zijn; men moet het idee, spelregels en implementatie goed kunnen bevatten.
- Het spel moet uitdagend zijn, zodat men bij het spelen gemotiveerd is te blijven spelen. Bij een te simpel spel zal men snel de aandacht verliezen.
- Het spel moet wiskundig uitgebreid genoeg zijn om het eerste onderdeel goed uit te kunnen voeren. Een spel als bijvoorbeeld Tetris is te eenvoudig en bevat weinig wiskunde.
- Het spel moet in een goedkope FPGA passen. De kosten van de hardware moeten laag blijven, dus het volledige spel moet te implementeren zijn in een goedkope, kleine FPGA.
- Het spel moet modulair op te bouwen zijn, zodat verschillende teams verschillende onderdelen kunnen bouwen.

Daarnaast zijn er nog enkele eisen aan de omgeving waarin de leerlingen hun VHDL-module moeten maken:

- Voor alle door leerlingen te implementeren modules van het spel moeten moet er een framework aanwezig zijn met testbenches en `entity`- en `architecture`-declaraties.
- Iedere leerling moet beschikken over een hardwaresimulator om het hele spel of alleen hun eigen module te kunnen simuleren.

Drie

Software-/hardwareontwerp

Dit hoofdstuk beschrijft het spel. De bedoeling van het hoofdstuk is om achtergrondinformatie te geven voor de docent die de lesmodule gaat geven in de klas. De leerlingen worden niet geacht om, voornamelijk paragrafen 3.3 en 3.4, volledig te kunnen begrijpen. Welke lesstof wél aan bod komt, is beschreven in hoofdstuk 4.

In paragraaf 3.1 wordt een beschrijving gegeven van het ontwikkelde spel. Paragraaf 3.2 geeft een overzicht van de benodigde hardware en software om het spel te kunnen maken en spelen. Paragrafen 3.3 en 3.4 beschrijven het technische ontwerp van het spel zelf en paragraaf 3.5 geeft een lijst met alle door leerlingen te implementeren modules.

3.1 Beschrijving van het spel

In paragraaf 2.8 zijn alle eisen van het spel aangegeven. Samengevat moet het spel eenvoudig, maar uitdagend zijn, het moet een goede wiskundecomponent hebben, het moet makkelijk uitbreidbaar en modulair te implementeren zijn voor een kleine FPGA. Daarnaast moet er ook een duidelijke relatie zijn tussen hardware en software.

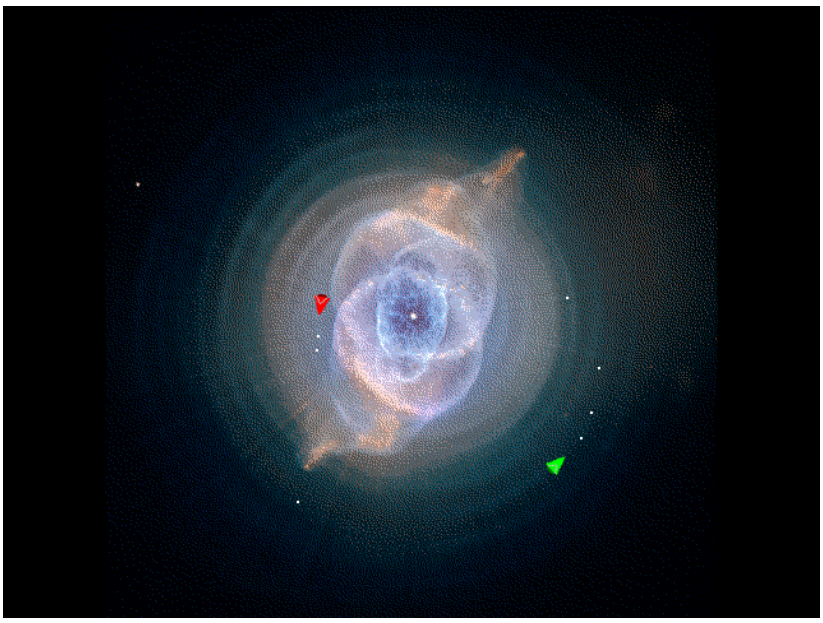
Er bestaan eindeloos veel spellen, waarvan er hier twee besproken worden. Ten eerste, een bordspel als dammen, schaken of go. Dit kan gemaakt worden door op het beeldscherm het bord te tonen en dan spelers om beurten een zet doen. Dit is niet zozeer een eenvoudig spel, maar wel een bekend spel. Vanuit de hardware gezien, is het overigens wel eenvoudig; het enige dat de FPGA hoeft te doen is het bord tekenen en de spelregels controleren. Dit geeft direct een duidelijke scheiding tussen hardware en software: de hardware kan het bord en stenen tekenen terwijl de software de spelregels controleert. Afhankelijk van hoe een embedded processor is geprogrammeerd, kan een ander spel gespeeld worden. Hierdoor wordt inzichtelijk dat dergelijke specifieke (spel)hardware toch generiek genoeg kan zijn om ieder bordspel te kunnen spelen.

Echter, de wiskundecomponent is wat lastig. Een mogelijkheid is om het bord in 3D te tekenen, maar dat is erg complex en voegt weinig toe aan het spel. Voornamelijk om dit laatste punt valt dit spel af.

Het tweede spel, dat wel aan alle eisen voldoet, is een ruimtespel waarin twee om een zon draaiende ruimteschepen elkaar moeten afschieten (naar het idee van KSpaceDuel[1]). Figuur 3.1 toont het scherm waarin er één zwaartekrachtpunt[2] is en twee

[1] Een spel dat bij KDE wordt meegeleverd, zie http://docs.kde.org/stable/en/kdegames/kspaceduel

[2] Het zwaartekrachtpunt is onzichtbaar, maar wordt gerepresenteerd door de achtergrondafbeelding: een opname van de kattenoognevel (NGC 6543) door de ruimtetelescoop Hubble [2].



Figuur 3.1: Screenshot van het ruimtespel met een zwaartekrachtpunt in het midden en twee spelers

spelers.

Het spel is eenvoudig: beide ruimteschepen (bestuurd door twee spelers) draaien in een 2D vlak onder invloed van de zwaartekracht om een zwaartekrachtpunt. Dit punt kan een ster, planeet of zwart gat zijn, afhankelijk van hoe het wordt weergegeven op het scherm, maar dat is voor het spel niet relevant. Ieder ruimteschip kan om zijn as draaien, kan gas geven en daardoor accelereren in de richting waarin het gedraaid is, en kan een kogel afschieten. Een schip kan niet remmen, hiervoor moet het schip 180 graden gedraaid worden om vervolgens gas te geven. Kogels verlaten het ruimteschip aan de voorkant, dus afhankelijk van de draaiing van het schip. Een kogel krijgt een iets hogere snelheid dan het ruimteschip dat hem afschiet en is vervolgens, net als de ruimteschepen zelf, onderhevig aan de zwaartekracht.

Het spel bestaat uit meerdere rondes. In iedere ronde beginnen de twee ruimteschepen op een bepaalde plaats en de ronde is af wanneer er (tenminste) één ruimteschip stuk is. Wanneer een ruimteschip geraakt is door een bepaald aantal kogels, onafhankelijk van wie de kogel heeft afgevuurd, gaat het stuk, is de ronde af en krijgt de tegenstander een punt. Door de puntentelling wordt bijgehouden hoeveel rondes gewonnen zijn door welke speler. Bij een botsing tussen de twee ruimteschepen is het gelijk spel en verandert de puntenstand niet. Als een ruimteschip in het zwaartekrachtpunt terecht komt, is het schip direct stuk. Kogels daarentegen verdwijnen als ze het zwaartekrachtpunt raken. Er is geen einde aan het spel; na afloop van iedere ronde volgt er een volgende.

Het spel is qua spelregels en speelbaarheid eenvoudig, maar uitdagend. Aangezien alles onderhevig is aan de zwaartekracht, is het lastig om kogels op het juiste moment af te vuren, zodat de tegenstander wordt geraakt. Lukraak kogels afschieten is doorgaans niet zinvol, aangezien deze blijven ronddraaien in het spel en het schip dat de

kogel afvuurde er evengoed door geraakt kan worden.

De wiskundige component van het spel is duidelijk: alle objecten (de ruimteschepen en kogels) bewegen afhankelijk van de zwaartekracht, de versnelling door gas geven en de eigen draaiing. Om de volgende positie te berekenen, afhankelijk van deze invloeden, moeten er een aantal relatief ingewikkelde formules bepaald worden.

Daarnaast kan functionaliteit zowel in software als in hardware geprogrammeerd worden. Bijvoorbeeld, wanneer twee objecten elkaar raken, moet bepaald worden of een ruimteschip stuk is, moeten kogels verdwijnen, de puntentelling moet worden bijgehouden, etc. Dit kan gedaan worden in hardware, met VHDL, of in software, op de embedded microprocessor.

De opbouw en inhoud van de modules van het spel en de benodigde resources in de FPGA zullen later bepaald worden. Concluderend kan gezegd worden dat dit ruimtespel voldoet aan de eisen, zoals deze gesteld zijn in paragraaf 2.8.

3.2 Overzicht benodigde hardware en software

Het spel zoals beschreven in paragraaf 3.1 moet worden uitgevoerd door een FPGA, zodat er tijdens het spelen geen pc meer nodig is. Dit betekent dat deze FPGA alle taken moet uitvoeren, van toetsenbordafhandeling tot baanberekening en VGA-beeldgeneratie. Het FPGA-bord moet goedkoop zijn, zodat het achtergelaten kan worden op school en het geen probleem is wanneer er een bord stuk gaat. Daarnaast is het belangrijk dat het handig is in gebruik met voldoende mogelijkheden tot aansluiten van randapparatuur.

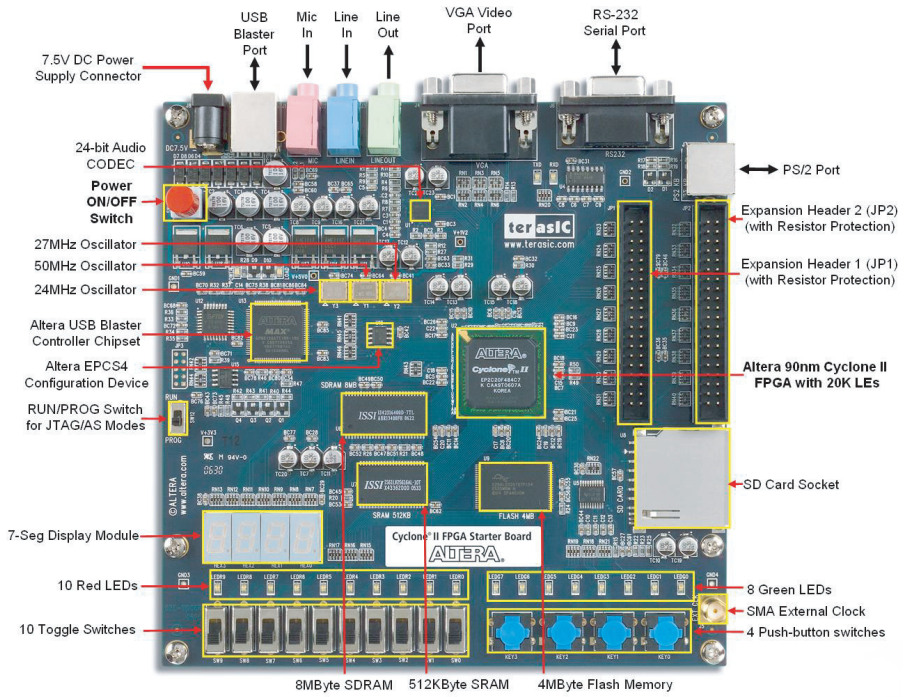
Elk bord met VGA- en toetsenbordaansluiting voldoet in principe aan de functionele eisen. De keuze is gevallen op de Altera Cyclone II FPGA Starter Development Kit [3]. Het bord is afgebeeld in figuur 3.2. De functionele eigenschappen van dit bord zijn:

- bevat de Cyclone II EP2C20F484C7N FPGA
- kan geprogrammeerd worden over USB
- bevat 8 MiB SDRAM, 512 KiB SRAM en 4 MiB flash geheugen
- heeft een VGA- en PS/2-aansluiting
- heeft 10 switches, 4 drukknoppen, 18 leds en 7 7-segment displays
- heeft microfoon- en line-ingang en line-uitgang (niet noodzakelijk voor dit spel)

Het bord wordt op de Altera-website[3], inclusief kabels en de benodigde synthese- en simulatiesoftware, aangeboden voor $ 150. De benodigde hardware en software om het spel te kunnen spelen en om de lesmodule op school te kunnen geven is als volgt:

- het bovenstaande FPGA-bord (eventueel met verloopstekker voor Europees stopcontact)
- een standaard VGA-beeldscherm
- een PS/2-toetsenbord

[3] http://www.altera.com



Figuur 3.2: Het gebruikte FPGA-ontwikkelbord: Altera Cyclone II FPGA Starter Development Kit

- één computer met de Altera Quartus II software om de FPGA te kunnen programmeren (de software is meegeleverd met het FPGA-bord)

- voor iedere leerling een computer met de Mentor ModelSim (Altera Edition) simulator (de software is meegeleverd met het FPGA-bord)

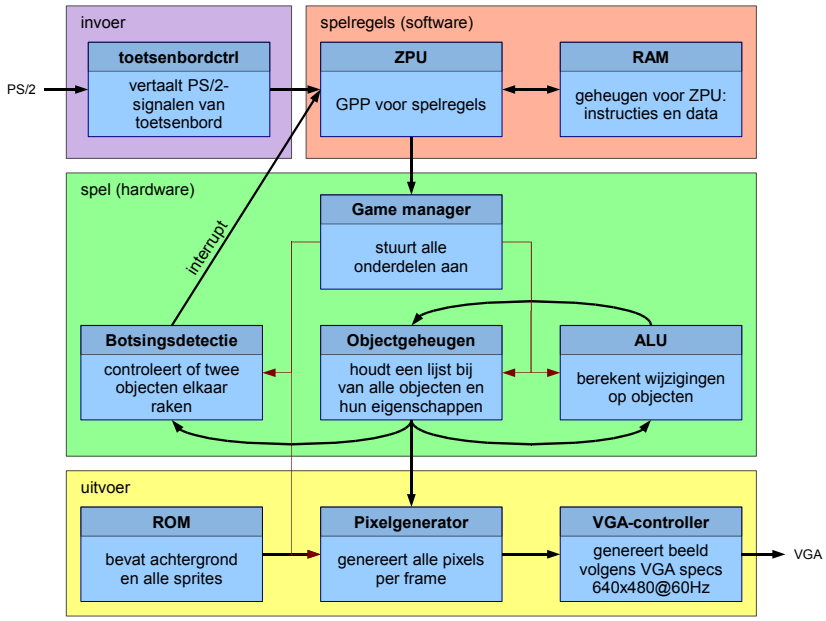
3.3 FPGA-ontwerp

Het ontwerp van het spel is voor de leerlingen niet zo van belang. Als achtergrondinformatie en kennis voor de docent is deze bijgevoegd.

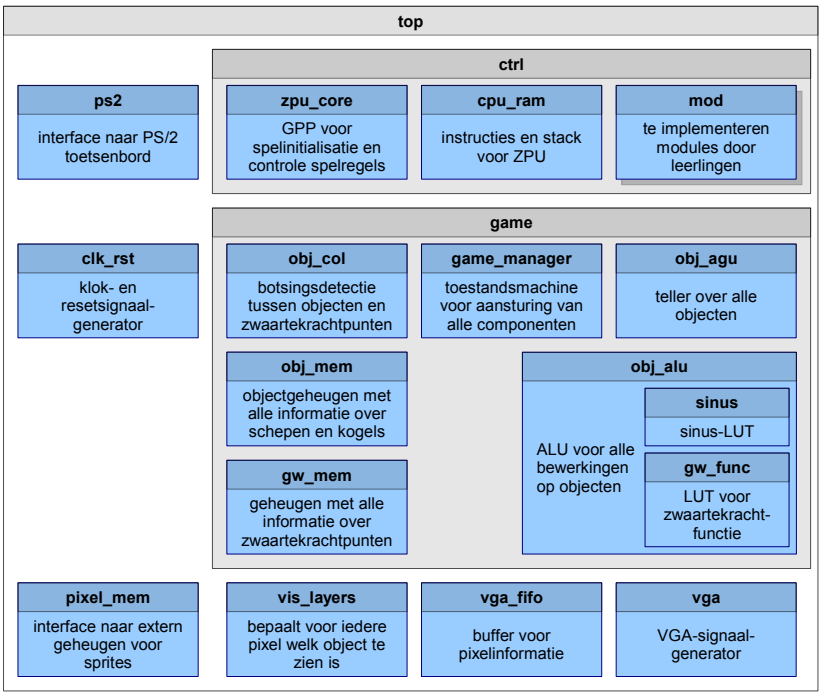
3.3.1 Ontwerp

Figuur 3.3 toont het high-level ontwerp van het spel. Dit ontwerp moet begrijpbaar kunnen zijn voor leerlingen. Echter, de blokken komen niet zo direct terug in de VHDL-code. Figuur 3.4 toont ieder blok zoals in de VHDL is te vinden met de juiste hiërarchie. De interconnecties tussen de blokken zijn voor overzichtelijkheid achterwege gelaten.

Het ontwerp `top` bestaat uit ruwweg drie grote onderdelen: het beheer van het spel (`ctrl`), het spelmodel zelf (`game`) en de in- en uitvoer (overige blokken in `top`). Ieder blok zal kort beschreven worden.



Figuur 3.3: Architectuur spel



Figuur 3.4: Architectuur spel, volgens de VHDL-code

Control

- **top**: Dit is de top-level van het ontwerp, zoals het in de FPGA geprogrammeerd kan worden.
- **ctrl**: De *control unit* initialiseert het spel en controleert de spelregels. Ook de punten worden hier bijgehouden. Alle spelregels kunnen op twee manieren worden gecontroleerd: met de ZPU (een GPP) of met in VHDL door leerlingen geschreven modules.
- **zpu_core**: De ZPU is een zeer kleine GPP. Leerlingen kunnen hiervoor software schrijven. De ZPU kan communiceren met de spelhardware om het spel te starten door onder andere schepen hun initiële positie en snelheid te geven. Zie paragraaf 3.4 voor details hoe de ZPU kan communiceren met de rest van de hardware.
- **cpu_ram**: De programmacode en de processorstack staan in een geheugen in de FPGA. Dit geheugen wordt tijdens het programmeren van de FPGA gevuld met de geschreven software.
- **mod**: De collecties van de door leerlingen geschreven modules vallen ook in het control-gedeelte.

Spellogica

- **game**: De `game` bevat alle logica om het spel zelf uit te voeren.
- **obj_mem**: Alle informatie van de objecten, zoals de positie, snelheid en draaiing, worden opgeslagen in dit geheugen. Ieder object heeft een uniek nummer, waaronder het object in het geheugen is opgeslagen. Adressering van objecten in het geheugen gebeurt door de `obj_agu`.
- **gw_mem**: Net zoals er een geheugen is voor objecten, is er een ook een geheugen voor de zwaartekrachtpunten (gravity well memory). In tegenstelling tot het objectgeheugen, verandert dit geheugen niet tijdens het spel. Alleen tijdens de initialisatie van het spel kan dit geheugen worden beschreven en kan er bepaald worden hoeveel van de zwaartekrachtpunten in het spel gebruikt worden.
- **game_manager**: De `game_manager` stuurt alle onderdelen binnen de `game` aan. De stappen die worden doorlopen tijdens het spel en de verantwoordelijkheden van de `game_manager` worden beschreven in paragraaf 3.3.2.
- **obj_agu**: De `obj_agu` (object address generation unit) is in essentie een teller die over de object- en zwaartepuntgeheugens heen kan lopen. Dit kan worden gebruikt om na elkaar alle objecten door de Arithmetic logic unit (ALU) te halen.
- **obj_alu**: Alle berekeningen op de objecten worden uitgevoerd door de object-ALU. Voor ieder beeld dat wordt gegenereerd, verplaatsen de objecten zich door de ruimte. De ALU berekent de volgende positie van een object, gegeven de positie, snelheid, draaiing, acceleratie en zwaartekracht. In paragraaf 3.3.3 wordt de ALU in meer detail besproken.

- **obj_col**: Wanneer twee ruimteschepen of kogels elkaar raken of wanneer een object in het zwaartekrachtpunt valt, dan zal de `obj_col` (object collision detection) dit aan de ZPU rapporteren.

In- en uitvoer

- **ps2**: De invoer van het toetsenbord wordt afgehandeld door `ps2`. Deze geeft de ontvangen toetsenbordcodes door aan de ZPU of een module, wanneer die is geïmplementeerd.

- **pixel_mem**: De achtergrond van het spel en alle sprites (kleine 2D-afbeeldingen die spelobjecten op het scherm representeren) van de ruimteschepen en kogels zijn opgeslagen in dit geheugen. Aangezien de achtergrond te groot is om in de FPGA zelf op te slaan, wordt hiervoor het externe 512 KiB SRAM-geheugen voor gebruikt.

- **vis_layers**: Om te bepalen welke kleur een pixel moet krijgen, moet per pixel bekeken worden of en welk object daar ligt. `vis_layers` doet dit. In paragraaf 3.3.4 wordt beschreven hoe dit in de hardware is gemaakt.

- **vga_fifo**: De gegenereerde pixels van `vis_layers` worden gebufferd in `vga_fifo`, waarna de `vga` de pixelinformatie daadwerkelijk op het scherm zet.

- **vga**: Het spel ondersteunt alleen een resolutie van 640 bij 480 pixels op 60 Hz. `vga` genereert het VGA-signaal op basis van de (gebufferde) pixelinformatie van `vis_layers`.

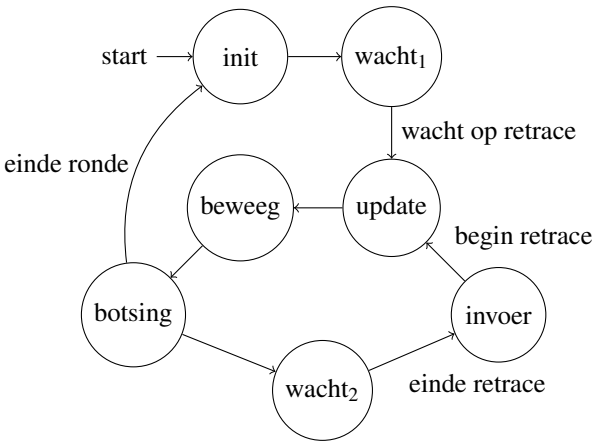
- **clk_rst**: De klokken en het resetsignaal worden gegenereerd door dit onderdeel. Het resetsignaal kan worden bestuurd door op het bord SW1 in te drukken. De klok waarop de ZPU en het spelmodel draait is ongeveer 50 MHz. De VGA-controller draait volgens de VGA-specificaties op 25,175 MHz.

3.3.2 Verloop van het spel

De werking van het spel wordt gedicteerd door het beeldscherm. Dit klinkt de omgekeerde wereld, maar is goed uit te leggen.

Het beeld wordt door de VGA-controller getekend per beeldlijn van links naar rechts, van boven naar beneden. Aangezien de gekozen beeldschermresolutie op 60 Hz draait, wordt ongeveer iedere zestigste van een seconde een volledige *frame* getekend. De VGA-controller genereert het beeld op basis van een geheugen met posities van objecten. Afhankelijk van de positie van deze objecten, wordt gekozen of de achtergrond wordt getekend, of een pixel uit de sprite (het plaatje) van een ruimteschip of kogel. Wanneer dit geheugen tijdens het tekenen van een beeld, of frame, wordt veranderd, dan kunnen halfgetekende objecten verspringen. Dit lelijke fenomeen is goed te zien, want objecten zijn dan 'gebroken', doordat bijvoorbeeld de bovenste helft is getekend op basis van de oude positie-informatie en de onderste helft van het object op basis van de nieuwe.

Dit probleem kan worden opgelost door de posities van de objecten alleen tijdens de *vertical retrace* te updaten. Deze vertical retrace, ook wel *vertical blanking interval*



Figuur 3.5: Interne toestanden van het spel, gesynchroniseerd door de vertical retrace van het beeldscherm

genaamd, is de periode dat een elektronenkanon in een beeldbuis[4] nodig heeft om te verplaatsen van de pixel rechtsonder op het scherm tot de pixel linksboven. In deze tijd, wat ongeveer 33 ms duurt, wordt er niets op het scherm getekend en dus is dit het uitgelezen moment om de objectinformatie te updaten.

Om deze reden wordt het spel gesynchroniseerd met deze vertical retrace. Per frame moeten de volgende zaken gebeuren, welke ook zijn gevisualiseerd in het toestandsdiagram van figuur 3.5:

1. wegschrijven van de berekende posities van alle objecten naar de VGA-controller (toestand *update*);
2. berekenen van de volgende objectposities (*beweeg*);
3. controleren of objecten botsen (*botsing*);
4. wacht tot het einde van de vertical retrace, wanneer we al klaar waren voor het einde (*wacht*$_2$);
5. afhandelen van invoer van de gebruikers (*invoer*). Merk op dat wanneer het spel niet in de toestand *invoer* is, de ingedrukte toetsen even worden bewaard, totdat de toestand weer is bereikt.

Tijdens het spelen van het spel worden de bovenstaande stappen voortdurend doorlopen. Voorafgaand aan een ronde moet het spel worden geïnitialiseerd en wacht het systeem totdat een speler op enter drukt om het spel te starten. Dit gebeurt tijdens de *init*-toestand, waarna in de *wacht*$_1$ wordt gewacht tot de eerstvolgende vertical retrace.

3.3.3 Object ALU

Het belangrijkste rekenwerk gebeurt in de ALU, namelijk in het onderdeel `obj_alu` van figuur 3.4. Deze ALU wordt bestuurd door de `game_manager` die de toestanden

[4]Merk op dat een TFT-scherm geen 'last' heeft van een elektronenkanon dat moet bewegen, maar deze timing-informatie ligt vast in de VGA-specificaties.

zoals getoond in figuur 3.5 doorloopt. In de toestand *beweeg* wordt de nieuwe positie van ieder object—de ruimteschepen en de kogels—berekend. De ALU is verantwoordelijk voor de volledige berekening van deze nieuwe positie. De theorie achter het model wordt ook aan de leerlingen aangeboden in les 2 (zie paragraaf 4.4).

De nieuwe positie van een ruimteschip hangt af van een aantal invloeden, of vectoren, te weten:

- de vorige positie van het object;
- de snelheid en richting van het object;
- in het geval van een ruimteschip, de versnelling en draaiing van het schip;
- de zwaartekracht van alle aanwezige zwaartekrachtpunten.

Merk op dat het spel een (in de VHDL-code) configureerbaar aantal zwaartekrachtpunten ondersteunt, welke nu is ingesteld op 4. Tijdens de initialisatie van het spel kan er voor gekozen worden om meerdere punten in te stellen. In het spel, zoals beschreven in paragraaf 3.1, wordt momenteel maar één van die vier punten gebruikt.

De zwaartekracht is gedefinieerd[5] als

$$F = G\frac{m_1 m_2}{d^2} \tag{3.1}$$

waarbij G de gravitatieconstante is, m_1 en m_2 de massa's zijn van de twee elkaar aantrekkende objecten, d de afstand tussen de twee objecten en F de kracht. Neem aan dat m_1 het bewegende object is en m_2 de veel zwaardere planeet, die nauwelijks verplaatst. De tweede wet van Newton is gedefinieerd als

$$\vec{F} = m\vec{a} \tag{3.2}$$

met de massa m van een object, de versnelling $\vec{a}$, vanaf nu te noemen $\vec{g}$ om latere verwarring te voorkomen met de versnelling van het ruimteschip a, en de bijbehorende benodigde kracht $\vec{F}$. Voor de benodigde beweging van de objecten in het spel is niet de kracht van belang, maar de versnelling. Gecombineerd levert dit

$$\vec{F} = G\frac{m_1 m_2}{d^2} = m_1\vec{g} \tag{3.3}$$

$$\vec{g} = \frac{1}{m_1}G\frac{m_1 m_2}{d^2} \tag{3.4}$$

$$= G\frac{m_2}{d^2} \tag{3.5}$$

Hierin is te zien dat de massa van het object m_1 niet van belang is voor de versnelling.

Aangezien dit spel in een artificiële ruimte speelt, worden de gravitatieconstante en de massa van het zwaartekrachtpunt vervangen door een instelbare 'constante' G' per zwaartekrachtpunt. Deze wordt zo ingesteld dat het spel 'natuurlijk' lijkt te verlopen. De formule voor de zwaartekracht wordt herschreven naar $\vec{g} = \frac{G'}{d^2}$. Gegeven dat $d = \sqrt{\Delta x^2 + \Delta y^2}$, is de x- en y-component van de zwaartekrachtvector, aangeduid met

[5] De formule is eigenlijk $F = G\frac{m_1 m_2}{r^2}$, maar om later verwarring te voorkomen met de rotatie r wordt voor de afstand tussen de objecten het symbool d gebruikt.

respectievelijk $\vec{g}_x$ en $\vec{g}_y$, gelijk aan:

$$\vec{g} = \frac{G'}{d^2} \tag{3.6}$$

$$\vec{g} = \vec{g}_x + \vec{g}_y \tag{3.7}$$

$$\frac{\vec{g}}{d} = \frac{G'}{d^3} = \frac{\vec{g}_x}{\Delta x} = \frac{\vec{g}_y}{\Delta y} \tag{3.8}$$

$$\vec{g}_x = \frac{\Delta x G'}{d^3} \tag{3.9}$$

$$\vec{g}_y = \frac{\Delta y G'}{d^3} \tag{3.10}$$

De formule voor de nieuwe positie is in feite een optelsom van de eerder genoemde vier vectoren, die van invloed zijn op een object. In verband met de generieke transformaties die ook in een videokaart worden gebruikt[6], wordt dit aan de leerlingen aangeboden als matrices. Door deze transformatiematrices te vermenigvuldigen, kan de complete formule voor de complete beweging eenvoudig worden bepaald. De matrices hebben altijd de volgende vorm:

$$T = \begin{pmatrix} \cos r & -\sin r & x \\ \sin r & \cos r & y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

waarbij r staat voor de gewenste rotatie van het te transformeren punt, x en y voor de gewenste translatie. In dit formaat staat de identiteitsmatrix I_3 voor een object in de oorsprong met draaiing 0. Op deze wijze kunnen alle benodigde vectoren die invloed hebben op de beweging van een object, worden uitgedrukt als matrix. Zodoende toont vgl. 3.12 de translatiematrix voor de acceleratie van a_x eenheden, vgl. 3.13 is de rotatiematrix waarin r de draaiing is van het schip, vgl. 3.14 is de translatiematrix voor de beweging met vector (b_x, b_y) die het object al had, vgl. 3.15 is de translatiematrix voor de invloed van een zwaartekrachtpunt, zoals bepaald in vgl. 3.9 en 3.10 en vgl. 3.16 is de translatiematrix die de vorige positie (x, y) bevat.

$$T_a = \begin{pmatrix} 1 & 0 & a_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

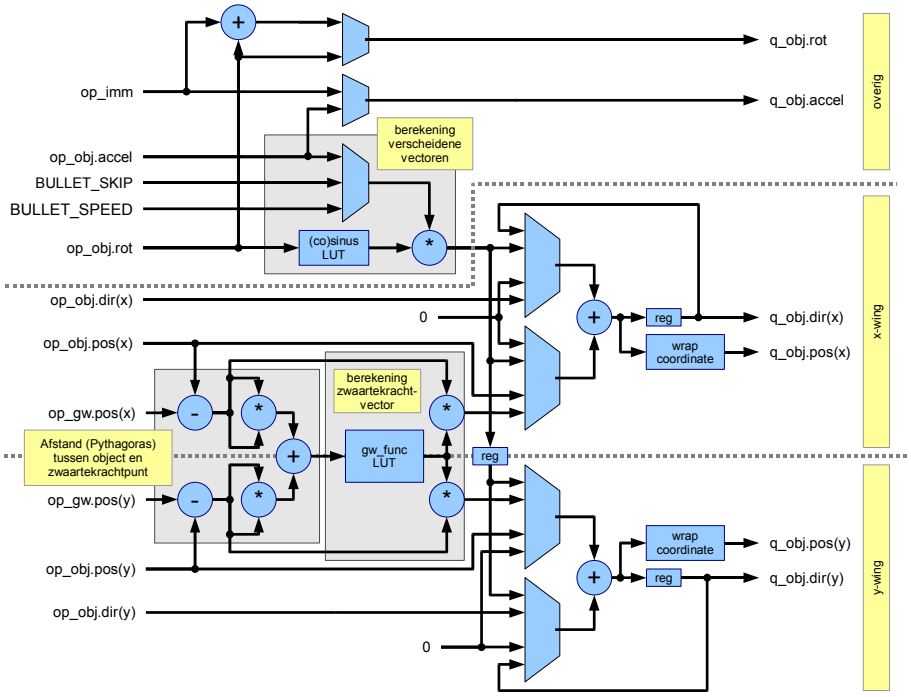
$$R = \begin{pmatrix} \cos r & -\sin r & 0 \\ \sin r & \cos r & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.13}$$

$$T_b = \begin{pmatrix} 1 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.14}$$

$$T_g = \begin{pmatrix} 1 & 0 & \vec{g}_x \\ 0 & 1 & \vec{g}_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.15}$$

$$T_p = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.16}$$

[6]The OpenGL Programming Guide - The Redbook: http://www.opengl.org/documentation/red_book/



Figuur 3.6: Ontwerp ALU

In de matrices T_a en T_b is er sprake van een versnelling in $\frac{\mathrm{m}}{\mathrm{s}^2}$ en in T_b een snelheid in $\frac{\mathrm{m}}{\mathrm{s}}$. Aangezien deze berekening wordt uitgevoerd per tijdseenheid, worden al deze eenheden gereduceerd tot een invloed op de positie in m. De eenheid van x en y is dus in principe m. Echter in deze artificiële ruimte heeft de meter geen betekenis en wordt er een abstracte afstandseenheid gebruikt. De matrices zijn zo ingericht dat wanneer deze worden vermenigvuldigd in de juiste volgorde tot de volledige transformatiematrix P, de formule voor de volgende x- en y-positie eruit komt:

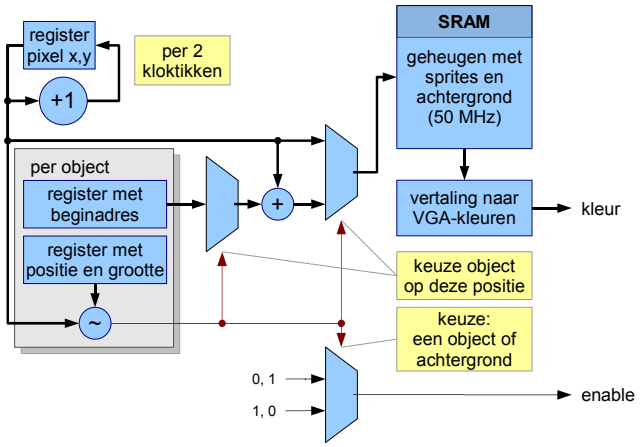
$$P = T_p T_g T_b R T_a I_3 \tag{3.17}$$

$$= \begin{pmatrix} \cos r & -\sin r & a_x \cos r + b_x + \frac{\Delta x G'}{d^3} + x \\ \sin r & \cos r & a_x \sin r + b_y + \frac{\Delta y G'}{d^3} + y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.18}$$

$$x' = a_x \cos r + b_x + \frac{\Delta x G'}{d^3} + x \tag{3.19}$$

$$y' = a_x \sin r + b_y + \frac{\Delta y G'}{d^3} + y \tag{3.20}$$

De formules voor x' en y' zijn geïmplementeerd in de ALU volgens figuur 3.6. Voor een beter overzicht zijn in het figuur de input G', de controlsignalen die de multiplexers en registers aansturen en de aanwezige pipeline weggelaten. De input-signalen zijn `op_obj.pos` voor de huidige positie van het object (x,y), `op_obj.dir` voor de richting van de huidige beweging (b_x,b_y), `op_pos.rot` voor de draaiing van het



Figuur 3.7: Ontwerp pixelgeneratie door `vis_layers`

ruimteschip r en `op_obj.accel` voor de versnelling a, welke resulteren in twee paar outputsignalen voor de nieuwe positie (x', y') en de beweging van het schip $(x', y') - (x, y)$. De symmetrie van de berekening van x' en y' is te herkennen in het figuur in de *x-wing* en *y-wing*.

Twee elementen van de te berekenen x' en y' verdienen nog extra aandacht. Een sinus is lastig in hardware uit te rekenen, dus is daar een Look-up table (LUT) voor gebruikt, waarin een aantal samples van een kwart van een sinus is opgeslagen en een volledige sinus en cosinus kan worden gegenereerd. De zwaartekrachtfunctie is eveneens moeilijk in hardware te berekenen en is daarom op een soortgelijke manier geïmplementeerd: de `gw_func LUT` bevat een aantal punten van de formule $\frac{G'}{d^3}$ (geven d als input) waarna door vermenigvuldiging van Δx en Δy respectievelijk x' en y' gevonden kunnen worden.

3.3.4 Pixelgeneratie

De kleurinformatie van de pixels voor de VGA-controller worden gegenereerd door `vis_layers`. Deze module houdt een lijst bij met de positie op het scherm en grootte van alle objecten die op het scherm staan. Deze lijst wordt tijdens de vertical retrace geüpdatet (zie paragraaf 3.3.2). Figuur 3.7 toont het (gesimplificeerde) ontwerp van `vis_layers`.

De VGA-controller draait op ongeveer 25,175 MHz, de rest van het ontwerp op ruim 50 MHz. Dat wil zeggen dat er per pixel ruim twee klokcycli beschikbaar zijn om de kleurinformatie aan de VGA-controller aan te bieden. Het scherm heeft een statische achtergrond, waar objecten overheen bewegen. De sprites van de objecten kunnen transparant zijn. Voor iedere pixel moet dus worden bepaald of de achtergrond moet worden getoond of een object.

Ieder object heeft een vierkante sprite, welke transparante pixels kan bevatten. Welke sprite bij een object hoort, is opgeslagen in het basisregister, welke de locatie in het geheugen van de eerste pixel van de sprite bevat. Zoals figuur 3.7 laat zien, loopt er een teller over alle pixelcoördinaten. Per pixel wordt gekeken of en welk object zich daar bevindt. Wanneer dat het geval is, wordt het geheugenadres berekend waar de

juiste pixel te vinden is, op basis van het basisregister. Tegelijkertijd wordt de kleurinformatie van de achtergrondafbeelding opgehaald. Het geheugen levert dus dubbel zoveel pixels dan nodig, welke door het `enable`-signaal eruit gefilterd worden.

Wanneer er zich geen object bevindt in de betreffende coördinaat of wanneer het object op die plaats transparant is, wordt de achtergrond getoond, anders de sprite van het object. Merk op dat twee objecten elkaar niet kunnen overlappen. Althans, in die situatie worden de transparante pixels van het bovenste object niet vervangen door het onderliggende object, maar door de achtergrondafbeelding. In de praktijk gebeurt dit niet; als twee objecten elkaar raken, dan zal tenminste een van beide verdwijnen uit het spel.

3.4 Firmware: ZPU

Met als doel om de relatie tussen hardware en software te laten zien, is er een GPP aan het systeem toegevoegd: de ZPU[7]. Aangezien de GCC toolchain beschikbaar is voor deze processor, kan er eenvoudig software voor geschreven worden in assembly of C/C++. In het lesmateriaal is de documentatie beschikbaar van deze processor, welke hier niet verder beschreven zal worden. Wat wel van belang is, is hoe de ZPU kan communiceren met de rest van het systeem.

De processor is verbonden met een bus, waaraan allerlei modules hangen. Een gedeelte van de modules kan worden geschreven door leerlingen, een aantal is noodzakelijk om het spel te kunnen starten en spelen. Tabel 3.1 toont een overzicht van alle apparaten aan de bus, welke in de VHDL-code is te vinden in `ctrl`. De ZPU kan de registers bereiken door een IO-read- of -write-operatie uit te voeren.

module	adres	RW[8]	omschrijving
Voert operaties uit op de ALU.	`0x1000`	W	operatie: doorgaans alleen voor accelereren. Wanneer dit register wordt geschreven, wordt de operatie uitgevoerd
	`0x1004`	W	het nummer van het object
	`0x1008`	W	een operand
	`0x100c`	R	de uitkomst van de operatie
Initialisatie van de 16 objecten.	`0x1100`	W	nummer van object, pas wanneer dit wordt geschreven, worden alle andere registers verwerkt
	`0x1104`	W	initiële rotatie van het object
	`0x1108`	W	initiële acceleratie
	`0x110c`	W	grootte van het (vierkante) object
	`0x1110`	W	initiële *x*-positie van middelpunt
	`0x1114`	W	initiële *y*-positie van middelpunt
	`0x1118`	W	initiële beweging in *x*-richting
	`0x111c`	W	initiële beweging in *y*-richting
	`0x1120`	W	type object: schip of kogel

[7] Zie het project "ZPU - the worlds smallest 32 bit CPU with GCC toolchain" op OpenCores: http://www.opencores.org

[8] Mogelijke toegang tot register: Read (lezen) of Write (schrijven)

Initialisatie van de 4 zwaartekrachtpunten.	`0x1200`	W	nummer van het zwaartekrachtpunt, pas wanneer dit wordt geschreven, worden alle andere registers verwerkt
	`0x1204`	W	grootte van het punt
	`0x1208`	W	*x*-positie van het middelpunt
	`0x120c`	W	*y*-positie van het middelpunt
	`0x1210`	W	vermenigvuldigingsfactor voor de zwaartekrachtfunctie
PS/2 toetsenbord afhandeling.	`0x1300`	RW	lezen: geeft 1 wanneer er een toets is ingedrukt; schrijven: wist de vlag en maakt klaar voor volgende toets
	`0x1304`	R	leest de volgende scancode (byte) van het toetsenbord
	`0x1308`	W	schrijft de volgende scancode (byte) naar het toetsenbord
Afhandeling van botsingsdetectie: stuurt een interrupt naar de ZPU zolang er een botsing is gedetecteerd.	`0x1400`	RW	lezen: geeft het type botsing (geen, object-object of object-zwaartekrachtpunt); schrijven: wist botsingsnotificatie
	`0x1404`	R	geeft het eerste gebotste objectnummer
	`0x1408`	R	geeft het tweede gebotste objectnummer, wanneer er een botsing is tussen twee objecten
Spelbesturing	`0x1500`	R	initialiseert een nieuwe ronde; lezen: data heeft geen betekenis, maar blokkeert totdat de operatie is uitgevoerd
	`0x1504`	R	start de nieuwe ronde; lezen: idem
Spriteselectie door schrijven naar basisregisters in `vis_layers`.	`0x1600`	W	nummer van het te updaten object, pas wanneer dit register wordt geschreven, worden de andere registers verwerkt
	`0x1604`	W	het geheugenadres waar de eerste pixel van de sprite staat
Bijhouden puntentelling.	`0x1e00`	W	het geschreven ruimteschipobjectnummer krijgt er een punt bij

Tabel 3.1: Overzicht registers aan de I/O-bus van de ZPU

De ZPU heeft twee essentiële taken: het afhandelen van toetsenbordinvoer en het toepassen van de spelregels als objecten botsen. De laatste is qua responsietijd kritiek. Wanneer de ZPU dit niet op tijd afhandelt, dan kan het zijn dat het spel stokt doordat de deadline van de vertical retrace wordt gemist. Het toetsenbord is van belang voor de responsie voor de spelers, maar is niet tijd-kritisch. Om deze reden moet in de assembly gecontroleerd worden op toetsenbordinvoer en veroorzaakt botsingsnotificatie een interrupt. Deze interrupt onderbreekt de ZPU altijd binnen enkele klokcycli, ook als de ZPU een blocking read of write uitvoert over de I/O-bus.

Voor de ZPU is alle benodigde assembly-code en tooling beschikbaar: de GNU-assembler kan worden gebruikt, de `crt0.S`-opstartcode is beschikbaar en het framework ligt klaar. Hoewel er nog geen opdrachten ontwikkeld zijn voor de ZPU, kan uit

de bestaande code eenvoudig stukken weg worden geknipt.

3.5 Uitbreidbaarheid voor leerlingen

De leerlingen kunnen opdrachten kiezen uit twee gebieden: VHDL-modules en de ZPU-programmacode. Merk op dat bijvoorbeeld zowel de ZPU als een VHDL-module de toetsenbordafhandeling kan doen. Dit wordt beide ondersteund, maar er moet wel besloten worden welk onderdeel deze taken op zich neemt, anders kan er onverwacht gedrag optreden; als zowel de ZPU als een module kijkt naar bijvoorbeeld input op de PS/2, dan kan het zo zijn dat één toets twee keer wordt verwerkt. Tabel 3.2 toont de kant-en-klare modules die leerlingen kunnen implementeren, maar hier kunnen eenvoudig modules aan toegevoegd worden.

naam	omschrijving	onderwerpen
draaien[a]	Bij een bepaalde toetsaanslag moet het juiste ruimteschip worden gedraaid.	VHDL, toetsenbordprotocol
draaien2[b]	Hetzelfde als *draaien*, maar dan in assembly voor de ZPU geïmplementeerd.	assembly, toetsenbordprotocol
gas[a]	Wanneer spelers de gas-knop indrukken en vasthouden, moeten hun ruimteschepen versnellen.	VHDL, toetsenbordprotocol
gas2[b]	Hetzelfde als *gas*, maar dan in assembly voor de ZPU geïmplementeerd.	assembly, toetsenbordprotocol
geluid[c]	Afspelen van geluiden bij bepaalde gebeurtenissen.	VHDL, geluidsformaat
init[b]	Spel initialisatie: aantal spelers, locatie spelers, snelheid, type zwaartekrachtpunten. Dit kan aangegeven worden via het toetsenbord voordat het spel begint.	assembly
punten[a]	Als een speler een punt haalt, dan moeten de 7-segment-leds (creatief) aangestuurd worden.	VHDL, creatief
regels[b]	Spelregels implementeren: wat gebeurt er als twee objecten botsen?	assembly, stackmachine
serieel[c]	Enige functie toevoegen aan de seriële poort.	afhankelijk van gekozen functie
sprite[a]	Als een ruimteschip draait, dan moet de sprite van dat object worden geüpdatet. Afhankelijk van de draaiing moet daartoe een nieuwe basisadres worden geprogrammeerd in `vis_layers`.	VHDL, inzicht in geheugenmanagement
sprite2[b]	Hetzelfde als *sprite*, maar dan in assembly voor de ZPU geïmplementeerd.	assembly, stackmachine
sprite3[c]	Sprites van de objecten aanpassen, afhankelijk van hun toestand, bijvoorbeeld door de schade aan een ruimteschip te tonen of kogels te kleuren afhankelijk van wie hem afschoot.	assembly/VHDL, creatief
tbleds[b]	leds aansturen van het toetsenbord	assembly, toetsenbordprotocol
toestanden[a]	Gegeven de speltoestanden, kunnen leds aangestuurd worden. Dit kan heel creatief met animatie of fade-in en fade-out.	VHDL, creatief
vuur[a]	Bij het indrukken van de vuur-knop van een van de spelers, moet er een nieuwe kogel in het spel worden gebracht.	VHDL, toetsenbordprotocol, administratie
vuur2[b]	Hetzelfde als *vuur*, maar dan in assembly voor de ZPU geïmplementeerd.	assembly, toetsenbordprotocol, administratie
zwkracht[b]	Veranderen van de zwaartekrachtfunctie.	wiskunde

[a] De opdracht is klaar voor leerlingen om aan te beginnen.

[b] Er is nog geen opdracht uitgewerkt, maar het is eenvoudig toe te voegen aan het bestaande framework.

[c] Het huidige framework is hier nog niet klaar voor, dus het aanbieden van de opdracht kost nog veel voorbereiding.

Tabel 3.2: Modules door leerlingen te implementeren

Vier

Ontwerp lessenserie

In dit hoofdstuk zal de lessenserie worden besproken zoals die gegeven kan worden in een 5 vwo- of 6 vwo-klas. De globale planning en de voorbereidende enquête, alsmede het te maken huiswerk en de verwachting wat leerlingen wel en niet weten of kunnen, worden besproken in paragrafen 4.1 en 4.2. De lesinhoud en -opbouw wordt beschreven in paragrafen 4.3 tot en met 4.8. Paragraaf 4.9 geeft een leidraad om de lessen te geven. Hiertoe zijn bij de meeste lessen presentaties ontwikkeld en lesplannen gemaakt. De lesplannen en stappenplannen zijn te vinden in respectievelijk appendices A en E.

4.1 Overzicht

Door de beperkte hoeveelheid tijd en de harde deadline aan het einde is het noodzakelijk om de lessen goed gestructureerd te houden. Dit geldt zowel voor individuele lessen als het gehele programma voor de lesmodule. De globale planning voor een serie van acht lessen is gegeven in tabel 4.1.

Voor iedere les is er een portie huiswerk; er wordt geen gebruik gemaakt van globale studiewijzers. Het huiswerk is een verlengde van les, waarin voornamelijk praktische zaken nogmaals aan bod komen. Er wordt niet verwacht dat mensen thuis nieuwe lesstof doornemen. Dit heeft ten doel om de behandelde stof te laten bezinken zonder dat er kostbare lestijd verloren gaat.

4.2 Voorbereiding: enquête

Ter voorbereiding wordt er een enquête gehouden. Dit heeft ten doel om te onderzoeken wat de leerlingen al kunnen en waar nog extra aandacht aan besteed moet worden. Op basis van de resultaten kunnen eventueel de lessen aangepast worden; specifieke onderdelen behoeven dan meer of juist minder aandacht.

Iedere vraag, betreffende een specifiek onderwerp of specifieke term, kan beantwoord worden met een cijfer op de volgende schaal:

1. Het onderwerp is onbekend.
2. Ik heb er wel eens van gehoord, maar het moet nog goed uitgelegd worden.
3. Ik heb het eerder wel gekend, maar het moet weer even opgefrist worden.
4. Het onderwerp is bekend; er hoeft niets over uitgelegd te worden.

les	onderwerp	huiswerk
0	enquête voorkennis	
1	• introductie onderwerp • eisen aan spel (brainstorm) • aanzet wiskundig model	uitwerken wiskundig model
2	• model afronden • formules omzetten naar hardware • hardware ontwerp maken	opfrissen kennis bits, bytes en binaire operaties
3	• introductie ModelSim • introductie VHDL • introductieopdracht	introductieopdracht VHDL afronden
4	• introductie stackmachine en ZPU • opdrachten simpele assembly	opdrachten simpele assembly afronden
5	zelf module bouwen	module
6	zelf module bouwen	module
7	zelf module bouwen	module
8	• modules afronden • uitleg chiptechnologie	enquête evaluatie

Tabel 4.1: Globale planning lessenserie

De enquête bevat de onderstaande vragen cq. onderwerpen. Per vraag wordt aangegeven wat wel of niet bekend wordt verondersteld. Op basis van deze aanname is de lesmodule afgesteld.

1. *Een computer rekent met bits en bytes. Weet je hoe je binaire, decimale en hexadecimale getallen in elkaar kunt omzetten?*
   Dit is basiskennis van de vierde klas. Het kan weggezakt zijn, maar leerlingen moeten dit eenvoudig en zelfstandig kunnen opfrissen. In de les wordt hier geen aandacht aan besteed.

2. *Een chip is gemaakt van een halfgeleider, bijvoorbeeld silicium. Weet je hoe een transistor in een halfgeleider werkt?*
   Dit onderwerp kan bij scheikunde aan bod gekomen zijn. Echter, aangezien lang niet iedereen scheikunde in zijn pakket heeft en het onderwerp (nog) niet aan de orde geweest kan zijn, zal het concept halfgeleider onbekend zijn. Wel kan worden aangenomen dat men weet wat atomen zijn.

3. *Als een chip wordt gemaakt, dan worden er altijd groottes genoemd. De laatste Intel Pentium Core2 processor is gemaakt met een 45 nm-proces. Weet je wat dit betekent?*

Wellicht dat een aantal mensen ‘45 nm’ bekend voorkomt, maar wat het betekend zal onbekend zijn.

4. *Een processor in de pc loopt op een klok van enkele GHz’en. Weet je hoe die snelheid wordt bepaald en wat de beperkende factor is om niet gewoon meteen naar 10 GHz te gaan?*
   De precieze technische achtergrond zal niet bekend zijn, maar dat er een strijd wordt geleverd om de kloksnelheid omhoog te krijgen, zal wel bekend zijn.

5. *Ken je de logische operaties als* and *(en),* or *(of),* xor *(exclusieve of),* not *(niet)?*
   Dit is 4V-basiskennis, net als de eerste vraag.

6. *Van die paar logische operaties kun je heel veel ingewikkelde dingen maken in een chip, zoals geheugen, een opteller, een mp3-speler en een grafische processor voor een spelcomputer. In hoeverre kun je met je kennis over bits en de logische operaties bedenken hoe dergelijke chips in elkaar zitten?*
   Een opteller wordt (meestal) besproken bij de logische operaties. Hoe een computer in elkaar zit, processor, bus, geheugen, etc., is ook in de vierde klas al behandeld. Het is aannemelijk dat men wel een globaal idee heeft hoe het zou kunnen werken.

7. *Zoals je software programmeert met Java, C++ of Delphi, kun je hardware programmeren met VHDL of Verilog. Weet je hoe je met VHDL of Verilog hardware kan maken?*
   VHDL zal onbekend zijn.

8. *Apparaten kunnen met elkaar communiceren over een draad. Hierin zitten vaak maar enkele aders (draadjes), vaak maar twee. Weet je hoe het protocol van de (digitale) communicatie werkt tussen bijvoorbeeld pc–muis, pc–printer, pc–beeldscherm, pc–pc via een netwerkkabel?*
   Er wordt doorgaans weinig aandacht besteed aan de precieze details van dergelijke communicatie. Een netwerkkabel (en de TCP/IP-stack) is waarschijnlijk wel bekend, maar details als bijvoorbeeld de front en back porch van VGA zal nooit behandeld zijn. Het is goed mogelijk dat met de reeds aanwezige basiskennis men wel redelijk snel een nieuw protocol kan begrijpen.

9. *Een processor in een pc kan via een bus communiceren met het geheugen en de videokaart. Weet je hoe een bus werkt?*
   Een bus is basiskennis van 4V.

10. *Een processor moet je programmeren. Je hebt de programmeertaal Java al gezien. Om je geschreven programma uit te voeren, moest je de code* compilen. *Wist je dat de computer* assembly *(machinecode) maakt?*
    Aan de compiler wordt geen aandacht besteed, maar bij het behandelen van de processor van een pc kan assembly wel ter sprake gekomen zijn. Machinecode zal wel bekend voorkomen, zonder de details ervan te kennen.

11. *Weet je hoe assembly werkt?*
    Waarschijnlijk heeft men nog nooit concreet assembly gezien of gemaakt.

12. *De processor voert een programma uit. Ken je de stappen van de processor: instruction fetch, decode, execute, writeback?*
    Op dit niveau is de processor niet of nauwelijks besproken, dus dit zal onbekend zijn.

13. *In de wiskunde wordt veel gebruik gemaakt van matrices en vectoren. In hoeverre ken je de basisoperaties als matrix-matrix- en matrix-vectorvermenigvuldiging?*
    Matrices behoren tot de stof van wiskunde B en D, maar dat komt waarschijnlijk pas in 6 vwo aan bod. Dus alleen de N&T-leerlingen kunnen het eventueel gehad hebben, afhankelijk wanneer deze lesmodule gegeven wordt.

4.3 Les 1: introductie en model

Voor de eerste les staat een hoop op het programma. Tijdens de lessenserie is de tijd beknopt, dus de werkdruk hoog. Dat moet tijdens de eerste les duidelijk zijn. Wat er moet gebeuren is: de organisatie duidelijk maken, een introductie geven op het onderwerp en een start maken met het model van het spel.

Onder de organisatie vallen een aantal zaken. De leerlingen moeten in groepjes van twee werken aan VHDL. Tijdens de eerste les moeten deze groepjes gemaakt worden. Mocht er een cijfer gegeven moeten worden, dan gebeurt dat op basis van de gemaakte VHDL voor de modules van het spel. Alle andere stof, de matrices van het model en de introductie over halfgeleiders en CMOS, is alleen ter informatie en zal niet getoetst worden.

Voor de les is er een presentatie beschikbaar. Deze zal hier besproken worden, ten einde een overzicht te geven wat in de les behandeld moet worden. Te beginnen met de introductie: waarom is het belangrijk om te weten hoe een chip in elkaar zit. Men heeft geen idee hoe zo'n ding nu daadwerkelijk werkt, men weet alleen globaal iets over bussen, geheugen en software. Om dit te verduidelijken is deze lessenserie ontwikkeld.

Het belangrijkste argument om een Application-specific integrated circuit (ASIC) te maken in plaats van alles in software te schrijven, is de simpele rekensom hoeveel Pentium-processoren je nodig hebt voor HDTV. Uit de rekenstappen op de slide blijkt dat er ongeveer tien Pentiums nodig zijn. Iedereen moet inzien dat er niet zoveel stroom wordt verbruikt en dat er niet tien ventilatoren op de achter kant zitten van een tv voor alleen de berekeningen.

Het spel is simpel: een zwaartekracht punt met twee ruimteschepen eromheen. De leerlingen kunnen zelf tijdens de brainstorm bedenken wat daarvoor gemaakt moet worden. Gedacht kan worden aan invoer via het toetsenbord, geluiden afspelen als een ruimteschip wordt geraakt, puntentelling op de leds. Men kan zonder voorkennis alle creativiteit kwijt in wat ze graag zouden willen maken. Niet alles kan worden gerealiseerd; het toevoegen van geluid kost gewoon teveel tijd. De ideeën die hier naar voren komen, kunnen input zijn voor de te ontwikkelen spelmodules.

Op basis van het te maken spel, moet een wiskundig model bepaald worden. We beperken ons hier tot de bepaling van de nieuwe positie van één ruimteschip gegeven de vorige positie, de richting van de beweging, de eventuele acceleratie van het ruimteschip, de draaiing van het ruimteschip en de zwaartekracht. Op een slide staat een overzicht van al deze vectoren. Men moet zelf, op basis van hun voorkennis van wiskunde en natuurkunde, een paar minuten nadenken hoe je op basis van deze gegevens de volgende plaats bepaalt.

Men zal er niet uitkomen, aangezien dit vrij complex is. Daarom introduceren we nieuwe stof: matrices. Het is van belang om te benadrukken dat het niets meer is dan een veld van cijfers, letters of formules; een matrix zelf is niets, alleen maar een handig hulpmiddel. Tijdens deze lessenserie zullen alleen 3×3-matrices worden

behandeld. Voor wat we ermee willen is de matrix-matrixvermenigvuldiging van belang. De docent moet voordoen hoe dit werkt, zonder op de details in te gaan. Het is van belang dat men weet dat matrices vermenigvuldigd kunnen worden met simpele optellingen en vermenigvuldigingen, maar hoeven het niet zelf te kunnen. Voor de leerlingen moet een matrix niets 'mysterieus' zijn, maar gewoon een simpele wiskundige berekening die ze zelf kunnen doen.

Omdat de volgorde van de vermenigvuldiging van belang is en om wat hands-onervaring te krijgen, is er een website beschikbaar die de leerlingen kunnen gebruiken om matrices te vermenigvuldigen. Deze website moet gebruikt worden om het model voor het spel op te zetten. De leerlingen moeten hiermee bepalen in welke volgorde welke type matrices vermenigvuldigd moeten worden. Hier kunnen de leerlingen de rest van de les mee bezig en het is tevens huiswerk.

4.4 Les 2: model en bijbehorende architectuur

In deze les moet de hardware centraal staan. Wat leerlingen waarschijnlijk nooit hebben begrepen is het nut van de wiskunde. De relatie tussen de abstracte wiskunde en een concreet hardwareontwerp moet in deze les duidelijk worden.

De les begint met het huiswerk. Althans, de formules voor de beweging van ruimteschepen moeten worden opgesteld aan de hand van de matrixvermenigvuldigingswebsite. Dit kan worden gedaan door klassikaal de website erbij te pakken, waarna de leerlingen kunnen aangeven in welke volgorde de vermenigvuldigingen moeten worden uitgevoerd. Dit resulteert in ingewikkelde formules, maar van belang is op te merken dat de leerlingen wel weten hoe ze tot stand zijn gekomen.

Op basis van de formules kan de hardware bepaald worden. De meest simpele manier is door alle optellers naast elkaar neer te leggen. Dit is de qua rekentijd de snelste, maar qua FPGA-resources de grootste oplossing. Het alternatief is langzamer, maar kleiner. Wat de docent moet laten zien is het creatieve proces van hardware maken, maar dat de wiskundige formule heel handig is om te hebben.

Optellen en vermenigvuldigen in hardware is eenvoudig. Echter, we hebben ook een deling en sinus nodig. De leerlingen kunnen hier even over nadenken hoe je dit zou kunnen implementeren. Merk op dat ze hier niet uit zullen komen, maar het is goed om even 'in hardware' te proberen te denken. De oplossing is wederom creatief: een tabel waarin alle waarden zijn voorgeprogrammeerd.

Tot slot moet de chip ontworpen worden. De invoer en uitvoer zijn bekend: respectievelijk het toetsenbord en beeldscherm. Daartussen zitten allerlei stappen, als de eerder behandelde hardware om de bewegingen te berekenen, controle van spelregels en geheugens om objecten op te slaan. In een olg kunnen leerlingen de blokken identificeren om zo tot een compleet ontwerp te komen. Er wordt niet verwacht dat men het na afloop zelf kan, alleen het hardwareontwikkeltraject moet worden getoond. Het complete overzicht op de laatste slide is een goed moment om even stil te staan bij wat we al allemaal hebben gezien.

De volgende les komt VHDL aan bod, dus moeten de leerlingen zelfstandig hun kennis over bits, bytes en logische operaties opfrissen. Dit hebben ze waarschijnlijk vorig jaar al bij informatica behandeld, dus al het lesmateriaal hebben al.

4.5 Les 3: signalen in de chip

Na de eerste twee intensieve lessen, gaan de leerlingen nu zelf aan de slag. In tegenstelling tot de vorige lessen, waarin voornamelijk een introductie is gegeven van stof die leerlingen niet hoeven te kunnen reproduceren, wordt de stof over VHDL wel getoetst. Er is een VHDL Quick Reference Card (VHDL QRC) beschikbaar en een zestal opdrachten die de leerlingen begeleiden met de eerste kennismaking met VHDL en de simulatieomgeving ModelSim. Tijdens deze eerste les is het goed als iedereen zelf werkt en nog niet in de tweetallen die eerder zijn gevormd.

De leerlingen moeten zelf een nieuwe taal aanleren. Dat is zeer inspannend en vergt doorzettingsvermogen en creativiteit. De VHDL QRC is verre van compleet als het gaat om de taal VHDL, maar is dusdanig afgestemd dat alle opdrachten ermee gemaakt kunnen worden. De docent moet tijdens de les de leerlingen stimuleren om te experimenten met de taal en de simulator en te proberen te doorgronden wat de VHDL-constructies doen.

Halverwege de les is het goed om even klassikaal VHDL te bekijken. Dit heeft ten doel om de relatie tussen de programmeertaal en de hardware, zoals men die ken met en- en of-poorten, duidelijk te krijgen. Tot slot kan er aandacht besteed worden aan wat het kloksignaal precies is en hoe geheugen werkt. Dit is niet echt noodzakelijk om de opdrachten te kunnen maken, maar het kan wel verhelderend werken voor het begrip wat er precies gebeurt.

Het is goed om de leerlingen te stimuleren om hun geschreven VHDL-code te testen op de FPGA. Het blijkt dat het erg motiverend werkt als hun eigen code ook daadwerkelijk leds aanstuurt.

Het huiswerk is het afmaken van deze introductieopdrachten.

4.6 Optioneel: les 4: ZPU

Mocht er genoeg tijd beschikbaar zijn, dan is het interessant om de ZPU te bespreken. Deze processor is ingebouwd in het spel en kan het spel initialiseren en de spelregels controleren. Door software te schrijven voor deze processor, wordt de relatie tussen hardware en software zichtbaar gemaakt.

De opzet is gelijk aan de vorige les: de leerlingen kunnen zelfstandig werken aan een serie opdrachten die de lesstof uitleggen. Hierbij ligt de nadruk op het schrijven van assembly in tegenstelling tot een hoog-niveau taal als Java en hoe de software communiceert met hardware. Hiervoor is een simulator en de toolchain met assembler en linker beschikbaar. Echter, door tijdsdruk is deze les niet verder ontwikkeld en uitgevoerd in de testklas.

4.7 Les 5, 6 en 7: module bouwen

In deze lessen kan men zelfstandig aan de slag om een module te ontwikkelen. Men werkt in tweetallen. Er is een lijst met modules waar men aan kan werken. Mensen die snel klaar zijn, kunnen meerdere modules aanpakken. Het huiswerk voor deze lessen is tevens het verder werken aan de te maken modules.

Tijdens de les kan iedereen componenten die goed simuleren aan de docent geven. Die zal vervolgens de synthese uitvoeren en de FPGA programmeren, waarna de leerlingen kunnen zien of het werkt. De leerlingen hoeven geen kennis te hebben van de FPGA-flow.

In de lijst van tabel 3.2 staan enkele mogelijke modules die uitgevoerd kunnen worden. Tijdens de brainstormsessie in de eerste les kunnen er ook nog ideeën gekomen zijn die aan de lijst toegevoegd kunnen worden.

Tijdens de les heeft de docent een ondersteunende rol. Het is van belang om de leerlingen te sturen bij het uitwerken van de modules. Veel VHDL-modules moeten een aantal zaken tegelijkertijd doen, zoals het zowel linksom als rechtsom draaien van beide ruimteschepen. Leerlingen hebben de neiging om alle problemen tegelijkertijd op te lossen, maar de docent moet ze stimuleren om eerst bijvoorbeeld te focussen op alleen linksom draaien van ruimteschip 1.

Daarnaast is het van belang dat leerlingen niet de hele les nadenken, maar ook experimenteren. Het is te verwachten dat men moeite heeft met de nieuwe taal en niet alle syntaxis precies begrijpen. Daarom moeten ze gewoon code uitproberen en kijken wat ModelSim of de FPGA ermee doet.

4.8 Les 8: afronden

Tijdens de laatste les is er één belangrijk onderwerp: hoe VHDL wordt vertaald naar een daadwerkelijke chip. Hiervoor komen een aantal zaken voorbij: VHDL, een schakeling, transistors, elektriciteit, CMOS-technologie, een wafer, lithografie, op atoomschaal chips bouwen, met als conclusie dat het maken van een chip vakoverschrijdend is.

In deze presentatie worden al deze onderwerp kort aangestipt. Het is van belang om rekening te houden met het feit dat lang niet iedereen natuurkunde en scheikunde in het pakket heeft en dus de basiskennis van bijvoorbeeld elektriciteit niet aanwezig hoeft te zijn. Het is wel van belang dat men inziet dat je met informatica alleen niet zoveel kan bereiken; andere vakgebieden spelen een grote rol bij de ontwikkeling van hardware.

4.9 Uitvoering

Dit hoofdstuk geeft een overzicht hoe de lesmodule globaal is opgebouwd. Voor de uitvoering van het bovenbeschreven ontwerp zijn er een aantal documenten beschikbaar. In appendix E zijn een aantal stappenplannen opgenomen die ter voorbereiding gevolgd moeten worden, te weten een stappenplan voor de uitvoering van de lesmodule, een voor het uitvoeren van een simulatie van het spel, een voor de synthese en een voor het opstarten van de FPGA en het spel. Appendix A bevat alle lesplannen voor de lessen waarin de leerlingen niet geheel zelfstandig werken. Tezamen met dit document zijn de presentaties beschikbaar, inclusief opmerkingen wat er verteld moet worden bij de slides en waar wel of geen nadruk op gelegd moet worden. Naast de bovengenoemde documenten zijn alle bronbestanden van het spel en lesmateriaal voor leerlingen beschikbaar, waaronder de VHDL-opdrachten en de VHDL QRC.

Vijf

Review

Dit hoofdstuk beschrijft hoe bekeken kan worden of de doelen, zoals geformuleerd in paragraaf 2.5, behaald zijn. De te toetsen doelen en beschikbare middelen daarvoor zijn beschreven in paragraaf 5.1, waarna de dataverzameling voor de middelen beschreven worden. De eigen ervaringen worden beschreven in paragraaf 5.2. In paragraaf 5.3 wordt beschreven welke vragen de inhoudelijke en vakdidactische expert zijn voorgelegd en hun reactie. De enquête die de leerlingen is voorgelegd en de resultaten zijn beschreven in paragraaf 5.4. Paragraaf 5.5 geeft een overzicht van de volledige review.

5.1 Methode

Aangezien dit een praktische opdracht betreft en inhoudelijke en theoretische kennis niet relevant is, zijn niet alle gestelde doelen gemakkelijk te toetsen. Dit geldt voor doelen 1 tot en met 4; deze doelen worden al bereikt wanneer de leerlingen in de les geluisterd hebben. Doel 5 is wel belangrijk om te bereiken en doel 7 is dat zeer gewenst. Daarnaast, wegens tijdsdruk is er geen aandacht aan de ZPU (les 4), waardoor doel 6 op voorhand niet te behalen is. Overigens, gezien de aard van de lesmodule is het wellicht belangrijker dat men op een prettige manier hebben kennisgemaakt met een nieuw vakgebied, dan dat men inhoudelijk veel heeft opgestoken.

Naast deze doelen, is het tevens van belang om te controleren of de lessen van voldoende inhoud zijn, of de lessen goed gegeven zijn en hoe de leerlingen deze module hebben ervaren. Tabel 5.1 toont alle te verifiëren doelen en de middelen die gebruikt kunnen worden om deze te controleren.

5.2 Eigen ervaringen

Per onderdeel zal beschreven worden hoe ondergetekende de lessenserie, zoals gegeven in de 5 vwo-klas, heeft ervaren. Dit zal beschreven worden per onderdeel: het model, de implementatie en de afrondende presentatie betreffende de realisatie van een chip.

5.2.1 Model

Tijdens de eerste twee lessen is er een grote hoeveelheid aan informatie over de leerlingen uitgestort. Het was voor de leerlingen duidelijk dat dit alleen ter kennisgeving aangeboden zou worden en het niet terug zou komen in een toets. Om de leerlingen

	eigen ervaringen	vakinhoudelijke expert	onderwijskundige expert	enquête leerlingen	toets leerlingen
Is de inhoud volledig en correct?		x			
Is de aanpak en opzet van de lessen juist?		x	x	x	
Is de lesstof leuk en interessant?	x			x	
Is de lesstof van voldoende niveau?	x	x		x	
Is de lesstof duidelijk gebracht?			x	x	
Sluit de lesstof goed aan bij voorkennis?	x			x	
Is de lesmodule goed gegeven?				x	
Hebben de leerlingen voldoende VHDL geleerd (doel 5)?	x				x
Is de relatie tussen de verschillende vakgebieden duidelijk (doel 7)?				x	

Tabel 5.1: Te verifiëren doelen en beschikbare middelen

actief te houden tijdens de les, heb ik geprobeerd om regelmatig de leerlingen aan het denk te zetten over allerlei onderwerpen. Dit is maar matig gelukt. Over het algemeen komen de leerlingen met weinig ideeën of oplossingen. “Ik snap het niet” is een veel gehoorde opmerking, maar men probeert de reeds aanwezige kennis over informatica, wiskunde of natuurkunde niet toe te passen op het probleem. Natuurlijk verwacht ik niet dat men alle problemen even oplost in vijf minuten, maar er is nauwelijks voortgang of aanzet tot oplossingen te bespeuren. Men blijft voornamelijk passief zitten in afwachting tot het antwoord van mij komt.

De leerlingen hebben het stuk over matrices goed opgepikt. Uit reacties uit de klas bleek dat men—in ieder geval een aantal leerlingen—bijvoorbeeld de matrixvermenigvuldiging begreep en wist hoe die uitgevoerd moest worden. Met de matrixvermenigvuldigingswebsite kon men goed werken, maar het is onduidelijk of men de bedoeling ervan heeft begrepen. De tweede les, waarin het model afgerond moest worden op basis wat men als huiswerk met de website gedaan heeft, bleek dat het huiswerk nauwelijks is gemaakt en dat men maar wat gokt wat er vermenigvuldigd moet worden. Het lijkt er niet op dat men er aandachtig naar heeft gekeken en er mee geëxperimenteerd.

De vertaling van de formules naar hardware was duidelijk. In de les heb ik twee alternatieve implementaties gegeven: een waarbij iedere optelling in de formule wordt vertaald naar een opteller in hardware en een waarin één opteller in hardware is geïmplementeerd die in tijd na elkaar alle operanden optelt. Dit sloeg goed aan. Men zag in dat er een verschil is in de benodigde hoeveelheid hardware en het gevolg in benodigde tijd.

De opbouw van de hele chip is waarschijnlijk niet echt aangekomen. Men zegt wel begrepen te hebben hoe het high-level ontwerp eruit ziet en waarom het zo in elkaar

zit, maar omdat men geen vergelijkingsmateriaal heeft wat betreft chipontwerpen, komt de architectuur uit de lucht vallen en blijft het waarschijnlijk niet hangen bij de leerlingen. Afwegingen wat betreft ontwerp en welke implicaties dat heeft op het resource-gebruik of performance blijven onbelicht. Meer tijd besteden aan het chipontwerp is niet gewenst, omdat het VHDL-gedeelte anders te weinig tijd krijgt. Misschien moet dit gedeelte meer naar de achtergrond of uit het materiaal gehaald worden.

Concluderend, de matrixvermenigvuldiging was wel duidelijk, maar men weet waarschijnlijk niet goed wat je ermee moet doen. De vertaling van model naar hardware was interessant. Vanwege het informatieve karakter met hoge informatiedichtheid van dit gedeelte hebben de leerlingen veel moeten luisteren en weinig kunnen doen. Een verbetering kan zijn om meer (kleine) opdrachten te geven dat aansluit bij deze stof.

5.2.2 Implementatie

Na de theoretische introductie volgt er een groot stuk praktijk: VHDL. Tijdens de les heb ik de VHDL QRC, opdrachten en bestanden voor ModelSim digitaal aangeboden. Dit bleek geen succes. Doordat de leerlingen in twee PDF-bestanden én ModelSim moesten werken, bleek men het overzicht kwijt te raken. De eerste VHDL-les is daardoor weinig productief geweest. Voor de volgende les heb ik al het materiaal uitgeprint, zodat men alleen ModelSim voor zich kon hebben op de computer. Dat werd als zeer prettig ervaren bij de leerlingen.

Aangezien ik nauwelijks iets heb uitgelegd over VHDL en ModelSim moest men zelfstandig de stof eigen maken. Dit is men blijkbaar niet echt gewend. De eerste vraag van de ModelSim-introductieopdracht was in de trant van “start de simulatie en kijk wat er gebeurt”. De reactie van veel leerlingen is direct: “ik snap het niet”, zonder eerst een paar minuten te kijken. Men is niet geneigd om eerst eens tien minuten te proberen het te snappen, ze geven het direct op.

Dit geldt niet alleen voor het werken met ModelSim, maar ook met VHDL. De VHDL QRC was slechts één A4. Bij het maken van de VHDL-opdrachten wordt er weinig geëxperimenteerd met wat op de VHDL QRC staat. Ik heb er vaak op moeten wijzen dat als ze niet weten wat ze aan VHDL moeten typen, ze gewoon de VHDL QRC moeten overtypen en kijken of het werkt. Tijdens de paar lessen werd men hier wel handiger in en krijgen ze de benodigde basisstructuren van VHDL wel onder de knie.

De te schrijven modules voor het spel zijn duidelijk afgebakend qua functionaliteit, maar zijn groot genoeg om er een paar lessen aan te besteden. Bijvoorbeeld, een module is dat toetsenbordinvoer afgehandeld moet worden, waarbij met vier knoppen beide ruimteschepen linksom en rechtsom gedraaid kunnen worden. Als eenmaal is uitgevonden hoe één speler zijn schip één richting op kan draaien, dan volgt de rest vanzelf. Echter, men probeert alle knoppen, alle opdrachten en alle functionaliteit tegelijkertijd te maken. Dit lukt doorgaans niet, waardoor de leerlingen het overzicht verliezen en uiteindelijk niets implementeren. Ik heb de leerlingen goed moeten sturen door bijvoorbeeld de tip te geven dat ze eerst eens moeten kijken wat er gebeurt als één van de vier knoppen wordt ingedrukt en pas als dat werkt door te gaan met het loslaten van de knop en de andere knoppen. Dit heeft de leerlingen goed op weg geholpen, maar zelfstandig volgen ze een dergelijke stapsgewijze methode niet.

De opstelling met FPGA-bord, toetsenbord en beeldscherm heeft alle lessen achterin de klas gestaan. Ik heb gezegd dat als men iets wilde testen, dat ze mij de

code konden geven, zodat ik de FPGA zou programmeren. Uiteindelijk hebben twee groepjes één keer een test uitgevoerd. De tests waren redelijk succesvol, wat bij de leerlingen grote voldoening gaf.

Het is jammer dat zo weinig mensen iets hebben getest op de FPGA. Ik had men meer moeten stimuleren om ook kleine vooruitgang in de implementatie toch maar direct te testen op de FPGA. Daarnaast moet de VHDL-introductieopdracht ook uitgebreid worden met een FPGA-test, zodat iedereen tijdens de eerste les al iets met de FPGA heeft gedaan. Dit toont misschien de mogelijkheden van de FPGA en kan het enthousiasme vergroten bij de leerlingen al tijdens de eerste les.

De inzet van de leerlingen was tijdens de eerste les VHDL een probleem. Zoals beschreven heeft men problemen gehad met de digitale stof, maar daarnaast was men ook niet actief bezig tijdens de les. Aangezien de beschikbare tijd uiterst beperkt is, heb ik besloten om vanaf de tweede les iedereen per les een cijfer te geven voor de vooruitgang die ze hebben geboekt. Dit leidde direct tot een actieve houding en het heeft in dat opzicht het doel bereikt. Echter, hierdoor wordt de druk door externe factoren (cijfers) op de leerlingen vergroot, terwijl het beter zou zijn als de leerlingen door intrinsieke motivatie zouden werken, zoals het mogelijkerwijs bereikt zou kunnen worden door de FPGA een grotere rol te laten spelen.

Uiteindelijk hebben de meeste groepen een simuleerbare module geschreven met beperkte functionaliteit. Het is duidelijk dat de Java-voorkennis invloed heeft op de VHDL-code; een if-else-structuur wordt gebouwd zoals in Java, terwijl dat in hardware net anders werkt. Het zet mensen aan het denken over de anderen manier van programmeren, dat nodig is voor hardware ten opzichte van software.

Men had veel begeleiding nodig. Tijdens deze lessen vragen ze veel meer dan in andere informaticalessen. Mijn werkdruk was daardoor hoog en nog denk ik dat bepaalde leerlingen meer aandacht nodig hadden. Toch hadden de leerlingen zelfstandiger kunnen werken als ze zelf hun problemen op proberen te lossen met behulp van de geboden lesstof op papier en als commentaar in de VHDL-code. Ik verwachtte een zelfstandiger houding van de leerlingen dan dat ze hebben laten zien.

Over het algemeen ben ik tevreden met wat men heeft gedaan in de les. Iedereen heeft met ModelSim gewerkt en men heeft ontdekt dat hardware toch duidelijk anders is dan software. De opdrachten zijn goed afgebakend en begrijpbaar voor de leerlingen en de ontwikkelomgeving was goed te gebruiken. De belangrijkste verbeterpunten zijn om de FPGA een grotere rol te laten spelen en duidelijker de stapsgewijze oplossingsmethode aan te bieden aan de leerlingen.

5.2.3 Afronding

Tijdens de laatste heb ik de afrondende presentatie gegeven. Men vond het interessant en gaf aan dat het leuk was om te zien hoe een chip nu eigenlijk écht werkt. Over de inhoud van de presentatie en de reactie van de leerlingen ben ik tevreden.

Overigens bleek wel dat de kennis van andere vakken een duidelijke rol speelt bij het begrijpen van de stof. Bijvoorbeeld, bij het uitleggen hoe een transistor werkt werden twee opmerkelijke opmerkingen gemaakt: “Waarom moet je zowel de 3,3 V als de 0 V aansluiten? Alleen de 3,3 V is toch genoeg, 0 V heeft toch geen enkele zin?”, “Wat merkwaardig dat een transistor soms wel en soms niet geleidt, wie heeft dat nu bedacht?” Hieruit is af te leiden dat kennis van respectievelijk natuurkunde de scheikunde ontbreekt, maar dat belemmert men niet om de presentatie (in grote lijnen) te kunnen volgen.

5.3 Expertreview

Dit verslag, het lesmateriaal en de beschikbare VHDL is bekeken door een aantal reviewers. Hun (cursief gedrukte) commentaar en mijn reactie is in deze paragraaf gegeven.

5.3.1 Review code door vakinhoudelijk expert: Bert Molenkamp

Bert Molenkamp is medewerker aan de Universiteit Twente en verzorgt verscheidene VHDL-vakken voor studenten en cursussen voor bedrijven.

Is de inhoud volledig en correct?

Toelichting: Het doel is niet om VHDL-programmeurs op te leiden, dus de inhoud zal nooit volledig zijn in de zin dat bijvoorbeeld de volledige VHDL-specificatie aan bod komt. Echter, de vraag is of de aangeboden lesstof 'af' is en of de lesstof een duidelijk afgebakend gebied omvat, maar daarbinnen consistent en volledig genoeg is om als afgeronde module aan te bieden. Het te evalueren lesmateriaal hiervoor is: vanaf slide 5 van de presentatie van les 2, alle documenten van les 3, de opdrachtsomschrijving van les 5, de `.vhd`-bestanden behorende bij de opdracht van les 5 (te vinden in de map `fpga/modules/build/modules`) en de afsluitende presentatie van les 8.

Ter informatie kan dit verslag als achtergrondkennis worden aangenomen en zijn alle VHDL-sources beschikbaar in de map `fpga/core`. Hoewel de ZPU (gepland voor les 4) niet in de les is behandeld, is de processor zelf wel opgenomen in het ontwerp. In de map `fpga/modules/zpu/src` zijn de sources daarvan te vinden.

Ik ga er vanuit dat docenten met dit materiaal zelfstandig aan de slag gaan: Ik mis een handleiding/stappenplan voor een docent om een demo van het spel aan de praat te krijgen.

Reactie: Deze is bijgevoegd in appendix E.

Paragraaf 3.3.3 is wel erg summier. Je begint direct met te verwijzen naar figuur 3.6. Ik zou dat pas doen nadat de formules zijn afgeleid.

Reactie: Dat is nu herschreven.

Verder leidt in vgl. 3.3 de formule $F = \frac{G'}{d^2}$ af. In G' zitten dan de beide massa's. Je hebt het vervolgens over een 'constante' G' per zwaartekrachtpunt. Ik heb dan de indruk dat de massa's van een schip en de kogels gelijk zijn? Immers anders had je iets gehad als $F = ma$ met m de massa van het schip, resp. kogel?

Reactie: De afleiding is nu in meer detail en beter beschreven.

Vgl. 3.8 geeft aan dat $\frac{\vec{g}_x}{\Delta x}$ gelijk is aan $\frac{\vec{g}_y}{\Delta y}$. Ik zou hier de versnelling $\vec{g}$ (vector) willen ontbinden in $\vec{g}_x$ en $\vec{g}_y$.

Reactie: Het is eenvoudig in te zien dat de verhouding tussen de vector en zijn lengte en de verhouding tussen de x- of y-component en het bijbehorende verschil in x- en y posities van de massa's gelijk zijn. Dat is eenvoudiger dan de afleiding te geven.

Je hebt het over op pagina 24 over 'eerder genoemde vier vectoren'. Je laat het aan

de lezer over uit te puzzelen welke vectoren je bedoelt.

Reactie: Dat is nu herschreven.

Op dezelfde pagina komt matrix T in vgl. 3.11 en op de volgende pagina's ook nog. Je verwijst naar OpenGL, maar ik ga er vanuit dat de meeste docenten dit niet zullen lezen. Als ik aan een rotatie denk in 2D dan kan ik dat met een 2×2*-matrix beschrijven. Zoiets als:*

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos r & -\sin r \\ \sin r & \cos r \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{5.1}$$

(x,y) *is de huidige positie,* r *is de hoek waarover wordt gedraaid en* (x',y') *is de nieuwe positie. Het is op pagina 24 onduidelijk waarom een* 3×3*-matrix nodig is.*

Reactie: De 2×2-oplossing werkt ook, maar dan moeten de leerlingen niet alleen matrices kennen, maar ook vectoren. De 3×3-oplossing wordt ook gebruikt in OpenGL, behoeft minder uitleg en vind ik eleganter.

Midden op pagina 25 heb je het over 'pipeline weglaten', maar is dit eerder genoemd?

Reactie: Nee. Dit zijn specifieke implementatiedetails die te ver gaan om in dit document te beschrijven. Echter, wanneer men gaat kijken naar de code, dan zal de pipeline wel degelijk aanwezig zijn. Men wordt er op deze manier op geattendeerd dat het figuur een versimpelde weergave is van de implementatie, maar dat dit voor het begrip van het ontwerp niet uit zou moeten maken.

Figuur 3.7, 'register met beginadres', 'register met positie': Ik kan me voorstellen dat je een beginadres hebt, met vervolgens nog de grootte (is het altijd vierkant). Maar ik snap niet wat het verschil is tussen beginadres en positie.

Reactie: Deze tekst is herschreven.

Is de aanpak en opzet van de lessen juist?

Toelichting: Hierbij moet gelet worden op de volgorde en methode van aanbieden van de stof. Merk op dat zonder de mondelinge toelichting van de docent in de klas, het wellicht lastig is te beoordelen of de aanpak goed (genoeg) is. Deze vraag kan beantwoord worden op basis van dezelfde documenten als de vorige vraag. Met name de VHDL QRC en de stapsgewijze aanpak van de (introductie)opdrachten kan van belang zijn.

Op zich ziet de aanpak er goed uit. Op basis van alleen de sheets en de VHDL QRC, hebben ze door wat het verschil is tussen concurrent en sequentieel?

Reactie: Ik vermoed van niet, maar ik vind dat minder relevant. Het is in deze lessenserie veel belangrijker dan ze kennismaken met de materie dan om alle details van VHDL te kennen. Als er meer tijd beschikbaar zou zijn, dan kan ik me voorstellen dat dit wel een van de eerste onderwerpen is om meer aandacht aan te geven, tezamen met het syntheseresultaat van specifieke VHDL-constructies.

Als ik het goed heb begrepen doet de docent de syntheseslag. Ik zou er voorstander van zijn om de leerling ook een klein ontwerp zelf te laten synthetiseren en het gegenereerde schema (RTL) te laten bekijken. Ik denk dat ze het dan eerder door hebben.

Reactie: Dit houdt in dat de leerlingen ook met Quartus aan de slag moeten. Voor deze beperkte lessenserie lijkt me het ongeschikt om twee nieuwe, zeer complexe tools te introduceren: ModelSim én Quartus. Echter, dit is wel degelijk een goede kandidaat voor een uitbreiding van de lessenserie als er meer tijd beschikbaar is in de klas.

Bij de sheets. Dit is altijd lastig. Sommigen beweren dat je weinig op een sheet moet zeggen, anderen willen de sheets dus ook graag als naslagwerk kunnen gebruiken (en dus meer info). Heb je in ieder geval bij de sheets extra commentaar voor de docent opgenomen (wat moet er verteld worden).

Reactie: Deze informatie is nu opgenomen.

Is de lesstof van voldoende niveau?

Toelichting: Hierbij kan de beschikbare tijd en (beperkte) voorkennis van de leerlingen een rol spelen in de beantwoording van deze vraag.

Op basis van het materiaal is het eerder te zwaar dan te licht. Dat betekent dus dat de docent een selectie moet maken. Je geeft zelf al aan dat de ZPU niet echt aan bod komt. Dat lijkt me ook niet mogelijk met slechts acht lesuren.

5.3.2 Review les door vakdidactisch expert: Nico van Diepen

Nico van Diepen is vakdidacticus voor informatica aan de Universiteit Twente.

Is de aanpak en de opzet van de lessen juist?

Toelichting: Hierbij moet voornamelijk gelet worden op de didactische aanpak, zoals die is beschreven in dit verslag, en de presentaties, lesplannen en de matrixvermenigvuldigingswebsite.

(Zie volgende vraag.)

Is de lesstof duidelijk gebracht?

Toelichting: Wellicht is dit lastig te beoordelen zonder een les daadwerkelijk ervaren te hebben, maar het geboden lesmateriaal kan wel een rol spelen in de review. Hierbij zijn de documenten van les 3 en 5 voornamelijk van belang, aangezien leerlingen dit in principe zelfstandig moeten kunnen doornemen.

Aanpak en opzet van de module is in grote lijnen ok. Het materiaal heeft echter een ontzettend grote informatiedichtheid. Dit levert voor mij twee vragen op:

- *Is het uitvoerbaar in de geplande acht lessen?*
- *Kunnen leerlingen er zelfstandig mee aan de slag?*

Op beide vragen tendeert mijn antwoord, louter op basis van het gepresenteerde materiaal, naar de negatieve kant.

Verder heb ik twijfels over de inbedding van de lesstof in de bestaande kennis en ervaring van de leerlingen (Vygotski: zone van naaste ontwikkeling). Bij de introductie van het onderwerp komen veel nieuwe begrippen naar voren. En ook de beginoefeningen, met de matrixwebsite, bevatten veel nieuwe stof voor leerlingen. Hoe kunnen de leerlingen dat koppelen aan hun bestaande kennis, ervaringen, concepten? (Overigens: dit is per se niet bedoeld als kritiek op de matrix-website. Die vind ik sterk.)

Reactie: Ik ben het eens met de opmerking dat het een hoge informatiedichtheid heeft. Het is aan de docent om te bepalen in hoeveel detail door de presentaties gegaan wordt en hoeveel nadruk er op verschillende onderwerpen gelegd wordt. In de praktijk is ook gebleken dat acht lessen erg krap is. Ik denk dat zonder vermeerdering van de stof de module in bijvoorbeeld vier lessen meer gegeven kan worden. Het belangrijkste vind ik om wat te laten zien van hardware, niet om de leerlingen experts te maken op dat gebied. Daarom is er, mijns inziens, een verschil in informatie dat gepresenteerd wordt—waar de leerlingen niet zelf mee aan de slag hoeven—en de stof die in de opdrachten aan bod komt. De leerlingen zouden zelfstandig aan de slag moeten kunnen met de VHDL-opdrachten, maar in de praktijk blijkt dat ze wel veel ondersteuning nodig hebben; de VHDL QRC is compact en ze hebben weinig VHDL-voorbeelden gezien om na te maken.

5.4 Leerlingreview

Behalve observaties die in paragraaf 5.2 zijn beschreven, is er een enquête gehouden onder de leerlingen. De vragen en resultaten zijn te vinden in appendix D. In deze paragraaf zullen de belangrijkste resultaten worden besproken.

De eerste vijftien vragen konden worden beantwoord op een schaal tussen oneens (score 1) en eens (score 4). Geen van de vragen is extreem beantwoord (met een score minder dan 1,5 of meer dan 3,5). Als een score van 3 of hoger als ‘eens’ wordt beschouwd, dan kunnen de volgende uitspraken worden gedaan.

- Men vond deze lesmodule interessant (vraag 1) en men heeft veel geleerd (vraag 3, 6 en 15), maar de lesstof was moeilijk (vraag 4).
- Het tempo in de les was hoog (vraag 5) en er hadden meer lessen aan dit onderwerp moeten worden besteed (vraag 7).
- VHDL hoort bij informatica (vraag 13).
- De mening zijn verdeeld over of de onderdelen over matrices en VHDL interessant zijn (vraag 8 en 9). Over het spel zelf is men verdeeld, maar gemiddeld wel positief (vraag 12).
- De structuur van de lessen was goed (vraag 10), maar niet voor iedereen was het duidelijk wat er precies verwacht werd (vraag 11).

Als verbeterpunten (vraag 16) komen twee onderwerpen steeds terug: te weinig tijd en men had graag meer uitleg willen hebben. Blijkbaar vindt men het prettig om

klassikaal uitleg te krijgen over hoe je met ModelSim werkt en hoe VHDL werkt in plaats van dat ze zelf alles willen uitzoeken. Bij de positieve punten (vraag 17) wordt voornamelijk geantwoord dat de opbouw van de lessenserie en de presentaties goed zijn. Mondeling hebben leerlingen aangegeven dat “dit het moeilijkste onderwerp van het vwo is”.

5.5 Samenvatting

Is de inhoud volledig en correct?

Na aanpassing naar aanleiding van de review in paragraaf 5.3 is de inhoud als volledig en correct beoordeeld.

Is de aanpak en opzet van de lessen juist?

De inhoudelijke expert en de leerlingen hebben aangegeven dat de lessenserie goed in elkaar zat (zie paragrafen 5.3 en 5.4). De vakdidactische expert heeft twijfels over de informatiedichtheid en tempo waarin de stof wordt aangeboden, welke beide hoog zijn.

Is de lesstof leuk en interessant?

De leerlingen geven aan de lesstof interessant te vinden, maar zijn verdeeld over of de onderwerpen over matrices en VHDL interessant zijn (zie paragraaf 5.4). Het onderdeel over matrices kan nog verbeterd worden door meer aandacht te besteden aan wat je hieraan hebt, door bijvoorbeeld de relatie te laten zien met OpenGL. VHDL kan interessanter worden door de leerlingen meer te laten experimenteren met de FPGA (zie paragraaf 5.2)

Is de lesstof van voldoende niveau?

De lesstof was complex en van hoog niveau, hebben de leerlingen aangegeven (zie paragrafen 5.2 en 5.4). De expert beaamt dat en verwacht eerder een te hoog niveau dan te laag (zie paragraaf 5.3).

Is de lesstof duidelijk gebracht?

De leerlingen hebben aangegeven dat de presentaties duidelijk waren. Men had graag ook nog extra (klassikaal) uitleg willen hebben over VHDL in plaats van dat men dat zelf moest uitzoeken (zie paragraaf 5.4). De expert uit twijfels over de aansluiting bij voorkennis (zie paragraaf 5.3).

Sluit de lesstof goed aan bij voorkennis?

Tijdens de les is gebleken dat het stuk over VHDL goed te doen is en dat men de voorkennis over programmeren met Java proberen toe te passen. Bij de matrices en de afronding is gebleken dat bij sommige leerlingen wat voorkennis ontbreekt. Dat gaat niet om essentiële kennis en het wordt opgevangen door de docent (zie paragraaf 5.2).

Is de lesmodule goed gegeven?

De structuur en presentaties waren goed, maar men bleek bij de opdrachten niet altijd te weten wat ze moesten doen (zie paragraaf 5.4). Daarnaast hadden de leerlingen meer uitgedaagd moeten worden tijdens de les en had de noodgreep van het geven van cijfers per les wellicht voorkomen moeten worden (zie paragraaf 5.2). Deze twee onderdelen moeten nog bijgeschaafd worden met bijvoorbeeld meer aandacht aan de FPGA. Het is tevens sterk aan te bevelen om zo veel mogelijk lesstof op papier uit te delen in plaats van digitaal aanbieden.

Hebben de leerlingen voldoende VHDL geleerd (doel 5)?

Zoals in paragraaf 2.7 is beschreven, moeten leerlingen kennis hebben gemaakt met het ontwikkelen van hardware. De nadruk ligt niet op het opleiden van VHDL-programmeurs. Er is geen toets gegeven over de lesstof, maar uit de ingeleverde opdrachten kan worden afgeleid hoe ver men met VHDL is gekomen.

Qua aantal regels code is er niet zo veel gedaan, maar uit observatie is gebleken dat men veel heeft gewerkt aan die code. Men is goed bezig geweest met de simulator en er is iets dat werkt, maar niemand heeft de opdrachten afgerond. De `if-else`-constructie is het meest gebruikt, wat te verwachten is gezien de Java-voorkennis.

De leerlingen hebben minder gepresteerd dan verwacht. Dit kan verbeterd worden door de leerlingen beter te sturen door te wijzen op de VHDL QRC en door ze stapsgewijs de opdracht op te laten lossen (zie paragraaf 5.2).

Is de relatie tussen de verschillende vakgebieden duidelijk (doel 7)?

In de enquête hebben leerlingen aangegeven dat ze zien dat het maken van hardware vakoverschrijdend is (zie appendix D).

5.5.1 Overzicht

In tabel 5.2 is deze paragraaf samengevat. Concluderend zijn (na aanpassing) alle doelen behaald, behalve dat de mate waarin VHDL is geleerd is tegengevallen. Echter, er is wel wat gepresteerd, dus wordt het niet met een - beoordeeld, maar met +/-.

	eigen ervaringen	vakinhoudelijke expert	onderwijskundige expert	enquête leerlingen	toets leerlingen
Is de inhoud volledig en correct?		+[a]			
Is de aanpak en opzet van de lessen juist?		+	+/-	+	
Is de lesstof leuk en interessant?	+[a]			+	
Is de lesstof van voldoende niveau?	+	+		+	
Is de lesstof duidelijk gebracht?			+/-	+[a]	
Sluit de lesstof goed aan bij voorkennis?	+			+	
Is de lesmodule goed gegeven?				+[a]	
Hebben de leerlingen voldoende VHDL geleerd (doel 5)?	+/-				+/-
Is de relatie tussen de verschillende vakgebieden duidelijk (doel 7)?				+	

[a] na aanpassing

Tabel 5.2: Resultaat te halen doelen

Zes

Conclusies

Deze lesmodule is ontworpen voor het vak informatica voor een 5 en 6 vwo-klas. In de acht tot tien beschikbare lesuren voor de module, komen een aantal onderwerpen aan bod die niet bij het vak informatica behandeld worden. Uit de test met de 5 vwo-klas en de review, zoals beschreven in hoofdstuk 5, is gebleken dat dit een pittige lesmodule is, maar een goede toevoeging aan de lesstof. Leerlingen hebben aangegeven dat het niveau van de stof hoog is en de tijd vrij beperkt, maar de lessen interessant waren.

De module laat leerlingen kennismaken met een nieuw onderwerp. Zodoende komen verschillende onderwerpen kort aan bod. Dit heeft een aantal nadelen. Ten eerste, het aanleren van een nieuwe programmeertaal kost veel tijd en oefening. Deze tijd is niet beschikbaar, dus het behaalde eindniveau wat betreft VHDL is relatief laag. Daarbij is het lastig te toetsen in hoeverre men het begrepen heeft. Een cijfer koppelen aan de prestatie is daarom niet eenvoudig. Ten tweede, leerlingen moeten zelfstandig aan het werk met VHDL en moeten zelf (gedeeltelijk) uitzoeken hoe het allemaal werkt. In de praktijk blijkt dat het begeleiden van deze groep leerlingen voor de docent een stuk zwaarder is dan normaal; er is geen boek dat ze stap voor stap door de stof heen helpt, maar de belangrijkste informatiebron is de docent. Ten derde, er is relatief weinig tijd en er wordt verwacht van de leerlingen dat ze zelf met de stof aan de slag gaan. Hiervoor is een sterke motivatie nodig vanuit de leerlingen. Als die er niet (iedere les) is, dan zal de lesmodule niet goed uit de verf komen; met alleen uitleg en presentaties van de docent blijft het geheel waarschijnlijk onduidelijk.

Desondanks is de module met succes gegeven in de testklas. Hoewel de tijdsdruk hoog was, is alle stof goed aan bod gekomen en hebben de leerlingen aangegeven dat ze het begrepen hebben en interessant vonden. De opzet van de module en het ontwikkelde spel is goed. Deze lessen laten zien hoe de techniek echt in elkaar zit, in plaats van dat er ‘speelgoedomgevingen’ worden aangeboden aan leerlingen om in te werken. De leerlingen werken nu met een echte simulator en echte hardware die gebruikt worden in de industrie en er wordt getoond hoe een chip daadwerkelijk in elkaar zit. Er worden geen speelse modellen gebruikt van schakelingen, maar gewoon de techniek zoals die is. Dit hebben de leerlingen als zeer interessant en verhelderend ervaren. Daarbij laat dit heel duidelijk zien welke relaties er zijn tussen verschillende vakgebieden. Bijvoorbeeld, de relatie is getoond tussen abstracte wiskunde en een concrete implementatie ervan in hardware. Daarbij geeft deze lesmodule mogelijkheden voor verdere verdieping in de stof, welke in paragraaf 6.1 besproken zullen worden.

6.1 Schaalbaarheid

Hoewel dit project zich richtte op een lessenserie van acht tot tien lessen, is het mogelijk om de inhoud dusdanig aan te passen, zodat er meer of juist minder lessen nodig zijn. In deze paragraaf zullen de mogelijkheden worden verkend.

6.1.1 Grotere lesmodule voor een kwartiel

Het is eenvoudig om meer toe te voegen aan dit lespakket. Stel dat een kwartiel à 10 weken en 20 lessen beschikbaar is, dan kunnen bijvoorbeeld de volgende extra onderwerpen aan bod komen:

- Voor leerlingen is de meest interessante toevoeging wellicht om met Quartus te werken, zodat de synthese duidelijk wordt. Daardoor wordt de relatie tussen de VHDL-constructies en de gegenereerde hardware inzichtelijker. De leerlingen kunnen bijvoorbeeld een les besteden aan Quartus zelf met wat speelgoed-VHDL-code en vervolgens hun eigen modules analyseren hoe het wordt gemaakt voor een FPGA. Het maken van de modules kost wat meer tijd, maar ze begrijpen beter wat er gebeurt en wat de VHDL betekent.
- Daarbij kan de FPGA ook aan bod komen. Nu krijgt de FPGA geen aandacht, maar wanneer de synthese nader wordt bekeken, kan ook de hardwaretechnologie behandeld worden, van FPGA tot ASIC.
- Leerlingen moeten nu kiezen tussen het maken van een module óf in VHDL, óf in software voor de ZPU. Wanneer de leerlingen beide moeten doen, bijvoorbeeld dat men eerst een stuk VHDL schrijft en er vervolgens in de software met die hardware moet communiceren, dan zien zij nog beter hoe hardware en software samenwerken.

6.1.2 Profielwerkstuk

Voor een profielwerkstuk, van 80 slu per persoon, is de stof ook zeer geschikt. Er is veel tijd beschikbaar en de bedoeling van het profielwerkstuk is dat men zelfstandig werk verzet. Aangezien VHDL en assembly leren voornamelijk veel tijd kost, kunnen leerlingen hier ook goed zelfstandig mee aan de slag.

Het schrijven van meerdere modules is echter niet zo geschikt. Wanneer er één is geïmplementeerd, dan is het maken van de rest gewoon meer van hetzelfde. Aangezien er een duidelijk verband is tussen verschillende vakgebieden, kunnen leerlingen voor hun werkstuk naast het implementeren van hardware of software, zich ook verdiepen in de natuurkunde achter het spel of elektrische schakeling, de scheikunde van de transistor of de wiskunde van het model en matrices.

Een andere mogelijkheid is dat de leerlingen zich bijvoorbeeld verdiepen in een specifiek onderdeel, zoals de ZPU. Zonder dat ze VHDL schrijven, weten ze wel dat er hardware is die via registers aangestuurd moet worden en het schrijven voor een stackprocessor is ook een uitdaging. Het project zou dan kunnen zijn om de volledige firmware te ontwerpen en te bouwen.

In het verlengde van de eerste presentatie, kan er ook beter gekeken worden naar het verschillen tussen een ASIC en GPP. Leerlingen kunnen zich verdiepen in wanneer welk type chip beter is geschikt en kunnen, als dit spel losgelaten wordt, ook kijken wat er in hardware in het dagelijks leven is ingebouwd. Kortom, het ligt eraan welke kant een leerling interessant vindt, maar er is genoeg uit te pluizen.

6.1.3 Lespakket voor één dag

VHDL is niet geschikt om in een dag(deel) even aan te leren. Wanneer leerlingen slechts enkele uren bezig willen zijn, dan kunnen ze een keus maken tussen hardware en software.

Schakelingen zijn al bekend, maar er is nooit echt serieus mee gewerkt. Een kleine opdracht zou kunnen zijn dat ze een circuit ontwerpen, dit bouwen en in een FPGA programmeren. Het bouwen kan met VHDL, wanneer bijvoorbeeld slechts alleen de logische expressies worden gebruikt. Het ontworpen circuit is dan direct te vertalen naar VHDL en te synthetiseren voor een FPGA. Ze zien dan dat er een chip wordt geprogrammeerd met hun circuit die vervolgens bijvoorbeeld leds aan kan sturen.

Een andere mogelijkheid is dat ze alleen kijken naar de stackprocessor. Het principe van deze processor is eenvoudig en de syntaxis van de bijbehorende assembly is snel uit te leggen. Het komt dan voornamelijk aan op het creatief omgaan met de stack en de beperkte instructieset. De leerlingen zien dan op een laag niveau hoe de eenvoudige processor wel veel kan doen. Er is dan niet zozeer meer een relatie met hardware ontwerpen, maar wel met wat meer techniek in een computer.

A

Lesplannen

Les 1: VHDL: introductie

Voorkennis: geen

Onderwerp: introductie praktische opdracht

Doelstelling: introductie geven, spel bedenken, model maken: leerlingen moeten in staat zijn om met matrixvermenigvuldigingswebsite te werken

Hulpmiddelen: presentatie
website voor matrixvermenigvuldiging

Onderwijsfunctie	Docent	Leerlingen	Tijd
Activeren voorkennis	controleren of enquête is ingevuld	groepen van 2 ll maken	0:00
Leerdoelen motivatie	introductie praktische opdracht eindcijfer op basis van modules voor spel		0:05
Presenteren kennis	intro hardware brainstorm spel	wat weten ze al?	0:08 0:15
Oefenen		zelf over formules laten denken	0:25
Presenteren kennis/werkwijzen	matrices introduceren		0:30
Oefenen en feedback krijgen		met website spelen	0:40
Huiswerk		met de website de juiste formules voor het model opstellen	0:50

Les 2: VHDL: matrices/model

Voorkennis: introductie onderwerp en matrices, ervaring met de matrixvermenigvuldigingswebsite

Onderwerp: model en architectuur

Doelstelling: de stap van wiskundig model naar hardware implemenatie moet duidelijk zijn

Hulpmiddelen: presentatie
website voor matrixvermenigvuldiging

Onderwijsfunctie	Docent	Leerlingen	Tijd
Activeren voorkennis	website erbij pakken, juiste volgorde vermenigvuldigingen bepalen	aangeven welke stappen in welke volgorde gedaan moeten worden (=huiswerk)	0:00
Nabespreken/ samenvatten	samenvatten stappen tot complete formules, OpenGL gebruikt ook deze techniek!		0:05
Presenteren kennis	creatief proces: bepalen architectuur op basis van formules	wat is het verschil tussen beide implementaties?	0:10
Oefenen		nadenken hoe een deling en sinus te implementeren	0:20
Presenteren kennis	LUT laten zien = trucje = creatief		0:25
Presenteren kennis	op basis van spelidee en model, complete architectuur bepalen	hardwareblokken identificeren	0:30
Nabespreken/ samenvatten	samenvatting idee–model–architectuur		0:40
Huiswerk		kennis over bits/bytes/logische operaties opfrissen	0:50

Les 3: VHDL: introductie hardwaretaal VHDL

Voorkennis: het proces van het maken van een chiparchitectuur is bekeken en op een hoog niveau zijn hardwarestructuren behandeld

Onderwerp: een nieuwe taal: VHDL

Doelstelling: na de les hebben de leerlingen VHDL geschreven in ModelSim

Hulpmiddelen: presentatie
ModelSim
VHDL Quick Reference Card
VHDL introductieopdracht met bijbehorende bestanden

Onderwijsfunctie	Docent	Leerlingen	Tijd
Activeren voorkennis	architectuur van chip laten zien, samenvatten wat we hebben gedaan		0:00
Oefenen en feedback krijgen	wijzen naar QRC en opdrachten	opdracht 1 en 2 maken	0:02
Presenteren kennis	klassikaal relatie tussen VHDL en hardware laten zien		0:25
Oefenen en feedback krijgen		opdracht 3, 4 en 5 maken	0:30
Presenteren kennis	klok en geheugen uitleggen, waar nodig		0:45
Huiswerk		opdrachten afmaken	0:50

Les 8: VHDL: afronding

Voorkennis: VHDL en hardwareontwerp

Onderwerp: VHDL, leuk, maar hoe nu verder?

Doelstelling: na de les hebben de leerlingen kennis van hoe een chip wordt gerealiseerd in silicium

Hulpmiddelen: presentatie

Onderwijsfunctie	Docent	Leerlingen	Tijd
Oefenen en feedback krijgen	FPGA-build maken van modules	afronden modules	0:00
Presenteren kennis	presentatie chiptechnologie, CMOS, halfgeleider		0:35
Huiswerk		evaluatie-enquête invullen	0:50

B

Overzicht klas 5 vwo

leerling	pakket				prestaties		
	profiel	natuurkunde[a]	scheikunde[a]	wiskunde B/D[a]	SE[b]	informatica[c]	VHDL[d]
A	N&T	ja	ja	ja	6,3	7,1	7,1
B	E&M	nee	nee	ja	7,0	7,5	7,4
C	N&T	ja	ja	ja	6,3	6,4	6,7
D	E&M	nee	nee	ja	6,6	7,3	6,9
E	N&T	ja	ja	ja	7,2	8,5	7,5
F	E&M	nee	nee	nee	6,2	6,0	6,8
G	E&M	nee	nee	nee	7,4	7,4	7,3
H	N&T	ja	ja	ja	6,7	7,0	7,0
I	E&M	nee	nee	nee	6,8	7,3	6,6
J	N&T	ja	ja	ja	6,8	7,4	6,5
K	E&M	nee	nee	ja	6,6	6,3	6,8
L	E&M	nee	nee	ja	6,7	7,2	7,3
M	E&M	nee	nee	nee	6,7	7,5	6,8
gemiddelde					6,72	7,15	6,97

[a] leerling heeft dit vak in pakket

[b] gemiddelde cijfer voor schoolexamen voor alle vakken inclusief 4V

[c] gemiddelde cijfer voor informatica dit schooljaar

[d] behaalde cijfer voor praktische opdracht over VHDL

Tabel B.1: Overzicht leerlingen klas 5 vwo

C

Resultaten enquête voorkennis

In tabel C.1 staan de resultaten betreffende de voorkennisenquête in de 5 vwo klas van 14 leerlingen.

Nr.	Vraag	1: Het onderwerp is onbekend	2: Ik heb er wel eens van gehoord	3: Het moet weer even opgefrist worden	4: Het onderwerp is bekend	Gemiddelde
1	Een computer rekent met bits en bytes. Weet je hoe je binaire, decimale en hexadecimale getallen in elkaar kunt omzetten?	0	2	11	1	2,93
2	Een chip is gemaakt van een halfgeleider, bijvoorbeeld silicium. Weet je hoe een transistor in een halfgeleider werkt?	7	5	2	0	1,64
3	Als een chip wordt gemaakt, dan worden er altijd groottes genoemd. De laatste Intel Pentium Core2 processor is gemaakt met een 45 nm-proces. Weet je wat dit betekent?	8	3	1	2	1,79
4	Een processor in de pc loopt op een klok van enkele GHz'en. Weet je hoe die snelheid wordt bepaald en wat de beperkende factor is om niet gewoon meteen naar 10 GHz te gaan?	4	6	1	3	2,21
5	Ken je de logische operaties als *and* (en), *or* (of), *xor* (exclusieve of), *not* (niet)?	0	1	6	7	3,43
6	Van die paar logische operaties kun je heel veel ingewikkelde dingen maken in een chip, zoals geheugen, een opteller, een mp3-speler en een grafische processor voor een spelcomputer. In hoeverre kun je met je kennis over bits en de logische operaties bedenken hoe dergelijke chips in elkaar zitten?	4	6	4	0	2,00
7	Zoals je software programmeert met Java, C++ of Delphi, kun je hardware programmeren met VHDL of Verilog. Weet je hoe je met VHDL of Verilog hardware kan maken?	13	1	0	0	1,07
8	Apparaten kunnen met elkaar communiceren over een draad. Hierin zitten vaak maar enkele aders (draadjes), vaak maar twee. Weet je hoe het protocol van de (digitale) communicatie werkt tussen bijvoorbeeld pc-muis, pc–printer, pc–beeldscherm, pc–pc via een netwerkkabel?	2	7	4	1	2,29
9	Een processor in een pc kan via een bus communiceren met het geheugen en de videokaart. Weet je hoe een bus werkt?	0	1	11	2	3,07
10	Een processor moet je programmeren. Je hebt de programmeertaal Java al gezien. Om je geschreven programma uit te voeren, moest je de code *compilen*. Wist je dat de computer *assembly* (machinecode) maakt?	3	3	6	2	2,50
11	Weet je hoe assembly werkt?	9	5	0	0	1,36
12	De processor voert een programma uit. Ken je de stappen van de processor: instruction fetch, decode, execute, writeback?	7	5	2	0	1,64
13	In de wiskunde wordt veel gebruik gemaakt van matrices en vectoren. In hoeverre ken je de basisoperaties als matrix-matrix- en matrix-vectorvermenigvuldiging?	12	2	0	0	1,14

Tabel C.1: Resultaten enquête voorkennis 5 vwo

D

Resultaten evaluatie-enquête

1.–15. Zie tabel D.1

16. Geef tenminste één punt dat verbeterd kan worden aan deze lessenserie.

- meer tijd
- Het ging te snel.
- Er werd naar mijn mening te veel van ons verwacht. Het was allemaal toch iets te complex.
- De manier van hoe je programmeert in VHDL, wat de wave nou echt inhoud etc.
- Iets meer theorie in de vorm van uitleg. Ook had er veel meer tijd voor moeten worden uitgetrokken. Ik had echt te weinig tijd.
- meer uitleg in begin
- Opdrachten of procedures volgen. Zodat je eerst een betere orientatie kreeg of wat je moest doen. Ik wist niet welke stappen ik moest volgen. Om tot een goed werkend iets te komen.
- Wellicht hadden we beter een iets langere periode bezig kunnen zijn met dit onderwerp. Dan hadden we wellicht wat meer kennis gehad en dan hadden we zelf ook iets fatsoenlijks kunnen creëren.
- Het ging een beetje snel en i kdenk dat er iets teveel werd verwacht.
- Misschien wat meer uitleg hoe je precies een stuk VHDL code kan schrijven.
- meer tijd, iets meer uitleg over het programmeren
- er was te weinig tijd voor, hier zou je een hele periode aan moeten werken, dan kun je je er nog beter inverdiepen en begrijpen.

17. Geef tenminste één punt dat je erg goed vond aan deze lessenserie.

- de opbouw
- Nieuw onderwerp dat belangrijk is voor informatica.

vraag nr.		1: oneens	2: beetje mee oneens	3: beetje mee eens	4: eens	gemiddelde
1	Ik vond het interessant om nu eens echt het binnenste van een chip te zien.	0	1	9	2	3,08
2	Ik begrijp nu (beter) hoe een processor werkt.	0	3	7	2	2,92
3	Nu ik weet hoe hardware gemaakt wordt, kan ik me een voorstelling maken van hoe computers/-telefoons/PDA's/etc. ongeveer van binnen werken.	0	1	8	3	3,17
4	Ik vond het niveau van de lesstof te hoog, ik kon het nauwelijks begrijpen.	0	1	5	6	3,42
5	Ik vond het tempo in de les te hoog.	0	0	8	4	3,33
6	Ik heb veel geleerd. Er was genoeg lesstof om me in te verdiepen; vervelen was echt niet nodig.	0	1	8	3	3,17
7	Het aantal lessen was te weinig, ik had graag meer lessen willen hebben en zo meer kunnen leren van dit onderwerp.	0	2	3	7	3,42
8	Het onderdeel over matrices en het model van het spel vond ik interessant.	1	5	5	1	2,50
9	Het onderdeel over VHDL vond ik interessant.	0	8	4	0	2,33
10	De organisatie was goed; de lessen hadden een goede opbouw en waren goed gestructureerd.	0	3	6	3	3,00
11	De opdrachten waren duidelijk, ik wist goed wat er van me verwacht werd.	1	5	6	0	2,42
12	Het spel met de ruimteschepen vond ik leuk, ik wilde er graag mee spelen en aan werken.	0	3	7	2	2,92
13	VHDL staat niet in het boek, maar ik vind dat VHDL eigenlijk wel thuishoort in het informaticaprogramma; informatica is niet compleet zonder een chip gezien te hebben!	1	1	7	3	3,00
14	Deze paar lessen hebben een wereld voor me geopend; ik heb nu meer interesse in hardware/chips dan voorheen.	1	5	6	0	2,42
15	Ik heb geleerd dat om een chip te kunnen maken je informatica/elektrotechniek/natuurkunde/scheikunde/wiskunde nodig hebt. Ik wist niet dat die vakken zoveel met elkaar te maken hadden.	1	1	5	5	3,17

Tabel D.1: Resultaten enquête evaluatie na hardwarelessenserie

- Presentaties waren duidelijk maar toch moeilijk te begrijpen.
- De laatste les omdat je hier nou echt een visueel beeld erbij kreeg en hoe de chip nou echt werkt.
- Ik wist wat mij te doen stond, maar vaak wist ik dat niet hoe ik het moest doen, maar dat kwam door mijn eigen schuld (gebrek aan kennis).
- dat je kunt zien wat je gemaakt hebt
- Ik vond het ondewerp wel goed. De presentatie's waren wel goed.
- Dat er voor een ontwikkelbord gezorgd was.
- Dat je meer vrije tijd had en zelf wel een grote inbreng.
- Uitleg over hoe een chip gemaakt wordt.
- nieuw onderdeel
- goede opbouw.

18. Heb je nog andere opmerkingen over dit onderwerp?

- Ik heb liever dat je opdrachten krijgt en procedures moet volgen. Zodat je bepaalde richtlijnen krijgt over wat je moet doen. Ik had echt geen flauw idee welke stappen en hoe ik iets moest aanpakken

E

Stappenplan docent

E.1 Stappenplan geven van de lesmodule

Om deze lessenserie te geven, moet de docent een aantal zaken voorbereiden. Hieronder wordt een overzicht gegeven van de benodigde te ondernemen stappen.

1. **Planning**: Afhankelijk van het aantal te verzorgen lessen, kunnen verschillende onderdelen van de lesmodule weggelaten worden of juist uitgebreid worden. In tabel E.1 wordt een suggestie gedaan welke onderwerpen besproken kunnen worden, gegeven het aantal beschikbare lessen.

2. **Verdiepen in lesstof**: Vervolgens moet de docent kennis hebben van de lesstof. Hiervoor is het van belang om in ieder geval hoofdstukken 3 en 4 bestudeert te hebben. De docent moet de door de leerlingen te schrijven modules kennen. Het is in mindere mate van belang dat de docent alle code van het spel kent.

3. **Uitvoeren synthese-flow**: De docent de synthese-flow van het spel met de modules voor de leerlingen uit kunnen voeren. Hiervoor wordt Quartus gebruikt. Er is een Quartus projectbestand beschikbaar met alle benodigde instellingen. Deze is te vinden in de map `fpga/tools/quartus`. Tevens staat in dezelfde map een gesynthetiseerde demo-versie van het spel met alle functionaliteit. Gedetailleerde instructies voor synthese een starten van het spel zijn te vinden in paragraaf E.3.

4. **Verdiepen/aanpassen in lesmateriaal**: Voor de lessen is al het lesmateriaal aanwezig: de presentaties, notities bij de slides van de presentatie, opdrachten voor leerlingen en VHDL-code dat de leerlingen als framework kunnen gebruiken. Deze documenten zijn te vinden in de map `doc/les*`. De lesstof moet

beschikbare tijd	onderwerpen
< 7 lessen	te weinig tijd om wat te bereiken
8–10 lessen	zoals voorgesteld, zonder de ZPU
10–13 lessen	zoals voorgesteld, met ZPU
> 13 lessen	zoals voorgesteld, met ZPU en (nog te ontwikkelen) stof over FPGA-synthese

Tabel E.1: Suggestie te behandelen onderwerpen

wellicht worden uitgebreid of ingekort, afhankelijk van de beschikbare tijd en de voorkennis van de leerlingen. Eventueel kan de enquête van paragraaf 4.2 gehouden worden om inzicht te krijgen in de aanwezige voorkennis.

5. **Voorbereiding lesmodule**: Op school moet op iedere computer ModelSim worden geïnstalleerd. Tevens moet er een computer zijn (van school of van de docent zelf) waarop de synthese kan worden uitgevoerd. Daarnaast moet de hardware beschikbaar zijn: het FPGA-bord, een beeldscherm en toetsenbord. Het is sterk aan te raden om alle PDF-bestanden met opdrachten voor de leerlingen uit te printen en uit te delen in de klas.

6. **Geven van lesmodule**: Voornamelijk wanneer leerlingen zelf met VHDL aan de slag gaan, moet de docent ze stimuleren om veel op de FPGA te testen en te experimenteren met de gegeven VHDL-constructies. Leerlingen hebben de neiging om het hele probleem in een keer aan te pakken; de docent moet er voor zorgen dat ze dit stap voor stap opbouwen en testen. Zie de evaluatie in paragraaf 5.2.

E.2 Stappenplan uitvoeren simulatie

Voor alle modules is er één VHDL-bestand beschikbaar waarin de simulatieomgeving staat. Daarnaast is er een `run.do`-bestand die de simulatie start. De leerlingen moeten beide bestanden toevoegen aan een nieuw project in ModelSim en `run.do` uitvoeren.

Het spel zelf is in zijn geheel nauwelijks te simuleren. Dit komt voornamelijk door het relatief langzame beeldscherm; voordat er een frame wordt getekend, zijn er al vele milliseconden verstreken, die in ModelSim uren in beslag kunnen nemen. Voor de meeste onderdelen van het spel zijn aparte testbenches gemaakt, zodat alles snel en goed getest kan worden.

In de map `fpga/tools/modelsim` staat een projectbestand `space.mpf`. Deze bevat alle benodigde bronbestanden, testbenches en `.do`-bestanden om de benodigde simulaties uit te kunnen voeren.

E.3 Stappenplan uitvoeren synthese

1. Start Quartus.

2. Open het project `space.qpf` met de top-level `top.qsf` uit de map `fpga/tools/quartus`.

3. De synthese gebruikt de module-bestanden uit de map `fpga/modules/build/ingeleverd`. Overschrijf deze bestanden met het ingeleverde werk van de leerlingen.

4. Voer de volledige flow uit. Het programmeerbestand voor de FPGA dat door Quartus wordt gemaakt, is `fpga/tools/quartus/top.sof`.

E.4 Stappenplan opstarten FPGA en spel

1. Sluit alle apparatuur aan: het FPGA-bord (voor de stroom is er een adapter met verloopstekker voor Europees stopcontact beschikbaar), een VGA-monitor, een

PS/2-toetsenbord en een computer om het FPGA-bord te programmeren via een USB-kabel.

2. Schakel alle apparatuur in. De FPGA draait nu een standaard testprogramma, welke wat op het scherm toont en de ledjes laat knipperen.

3. Gebruik de Quartus Programmer (als onderdeel van Quartus en los te installeren met `bin/90_quartus_programmer.exe`) om de FPGA te programmeren. Programmeer de FPGA met `DE1_USB_API.sof` uit de map `fpga/tools/control_panel`.

4. Start `fpga/tools/control_panel/DE1_Control_Panel.exe` en maak verbinding met het FPGA-bord.

5. Upload de sprites (`fpga/modules/sprites/build/sram.bin`) naar het SRAM op het bord via het Control Panel. Zolang het FPGA-bord ingeschakeld blijft, zal de SRAM de sprites behouden, zelfs als de FPGA opnieuw wordt geprogrammeerd.

6. Programmeer de FPGA met het spel van `fpga/tools/quartus/top.sof`.

7. Na programmeren wordt het spel gereset en zal direct de achtergrondafbeelding en de ruimteschepen in startpositie te zien zijn op het scherm. Het spel begint wanneer op enter wordt gedrukt. Het spel kan worden gereset door KEY0 in te drukken.

Voor de demo van het volledig werkende spel moeten dezelfde stappen worden uitgevoerd, maar zijn de sprites en het spel te vinden in de map `fpga/demo`.

Bibliografie

[1] Examenprogramma informatica havo/vwo.

[2] The Cat's Eye Nebula: Dying Star Creates Fantasy-like Sculpture of Gas and Dust. http://hubblesite.org/gallery/album/nebula/pr2004027a/.

[3] Altera. Cyclone II FPGA Starter Development Kit. http://www.altera.com/-products/devkits/altera/kit-cyc2-2C20N.html.

[4] J.D. Bayliss en S. Strout. Games as a "flavor" of CS1. In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pages 500–504, New York, NY, USA, 2006. ACM. ISBN 1-59593-259-3. doi: http://doi.acm.org/10.1145/1121341.1121498.

[5] A. Eckerdal, M. Thuné, en A. Berglund. What does it take to learn 'programming thinking'? In *ICER '05: Proceedings of the first international workshop on Computing education research*, pages 135–142, New York, NY, USA, 2005. ACM. ISBN 1-59593-043-4. doi: http://doi.acm.org/10.1145/1089786.1089799.

[6] R. Franquinet, R. Leijtens, H. Reinders, en M. ter Wal. *Enigma*. www.enigma-online.nl, 2007.

[7] F.V.H. van Geel. Een overzicht van het Programmeeronderwijs op Middelbare scholen in Nederland. Onderzoek van onderwijs, Universiteit Twente, Juli 2008.

[8] N. Jacobson en S.K. Schaefer. Pair programming in CS1: overcoming objections to its adoption. *SIGCSE Bull.*, 40(2):93–96, 2008. ISSN 0097-8418. doi: http://doi.acm.org/10.1145/1383602.1383643.

[9] P.R. da Motta Junior en H.A. de Lima Junior. Teaching Computer Programming: A Game Driven Approach. In *Brazilian Symposium on Computer Games and Digital Entertainment*, 2006. ISBN: 85-7669-098-5.

[10] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, en J. Paterson. A survey of literature on the teaching of introductory programming. In *ITiCSE-WGR '07: Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 204–223, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1345443.1345441.

[11] V. Schmidt. Vakdossier 2007 informatica. Technical report, SLO, 2007.

[12] V. Schmidt. Handreiking. Technical report, SLO, 2007.

[13] K. Squire en H. Jenkins. Harnessing the Power of Games in Education. In *InSight*, volume 3, chapter 1, pages 5–33. Institute for the Advancement of Emerging Technologies in Education, 2003.

[14] L. Williams, E. Wiebe, K. Yang, M. Ferzli, en C. Miller. In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education*, 12:197–212, September 2002. doi: 10.1076/csed.12.3.197.8618.