UNIVERSITEIT ANTWERPEN

Academiejaar 2020-2021

Faculteit Toegepaste Ingenieurswetenschappen

6-Embedded systems

Module VHDL debugging I^2C

Begeleider: Koen Lostrie

Universiteit Antwerpen

Inleiding

Het vak 6-Embedded systems wordt opgedeeld in 3 modules, die elk door verschillende begeleiders wordt opgevolgd. Deze module wordt begeleid door Koen Lostrie (koen.lostrie@uantwerpen.be).

In deze module is het de bedoeling dat gegeven VHDL code foutvrij gemaakt wordt. VHDL is een afkorting voor VLSI HDL, wat op zijn beurt staat voor *very large scale integration hardware description language*. Zoals ook in de studiegids vermeld staat, wordt een voorkennis van deze taal verondersteld.

Ter herinnering: VHDL is GEEN softwaretaal, maar een hardware beschrijvingstaal! Er is geen CPU waar deze code op zal "draaien", maar via synthesetools wordt het VHDL bestand omgezet naar een netwerk van logische poorten (EN-, OF-, EXOF-, NIET-poorten, flip flops,…): een zogenaamde netlist.

Doel van deze module

Het allerbelangrijkste leerdoel van deze module is debuggen. Hoe en met welke tools kan je best VHDL code debuggen? Er zullen drie belangrijke hulpmiddelen bekeken worden:

1) Simulatie: vooraleer de VHDL code omgezet wordt naar een netlist, kan de werking van de beschreven schakeling gecontroleerd worden aan de hand van een computersimulatie.
2) Logic analyzer/oscilloscoop: nadat de schakeling effectief geïmplementeerd werd in een programmeerbare logische component (FPGA in dit geval), kunnen in- en uitgangen gemeten worden aan de hand van een logic analyzer. In het labo zijn hiervoor LabNation SmartScopes beschikbaar.
3) ILA (Internal Logic Analyzer): om na implementatie ook nog interne signalen te kunnen bekijken, kan er tijdens het ontwerp extra hardware in de schakeling ingevoegd worden. Dit kan geautomatiseerd gebeuren aan de hand van de Xilinx development tools.

Het is dan ook zeer belangrijk dat deze drie hulpmiddelen uitvoerig gebruikt worden en ook beschreven worden in het portfolio (zie verderop: "Resultaten – portfolio").

De opdrachten behandelen de I^2C bus in detail, waarbij bit per bit gekeken wordt hoe dit protocol juist in mekaar zit. Hierbij zal een I^2C component verbonden worden met deze bus om vervolgens een eigen communicatie op te zetten hiermee. De werking van deze component (die voor iedere student verschillend kan zijn) is van ondergeschikt belang. Op de eerste plaats gaat het om de manier van debuggen van VHDL code en op de tweede plaats om het begrijpen van de gedetailleerde werking (op de fysieke laag van het OSI-model) van "een" protocol, in dit geval het I^2C protocol.

Voorbereiding

Het is ten zeerste aan te raden om vóór aanvang van de eerste zitting van deze module de vereiste softwaretools reeds te installeren, vermits dit zeer lang kan duren en de tijd voor de opdrachten beperkt is.

Xilinx Vivado kan je gratis downloaden via https://www.xilinx.com/downloads. De gratis Webpack editie ondersteunt gelukkig ook onze hardware, dus daar kan je gebruik van maken.

Resultaten – portfolio

Op het einde van de drie modules moet per module een portfolio ingeleverd worden. Dit wil dus zeggen dat er drie finale portfolio's ingeleverd moeten worden na de volledige laboreeks (9 zittingen). De exacte deadline voor inlevering zal op Blackboard verschijnen.

Voor deze module is het portfolio individueel (iedereen geeft zijn eigen portfolio af) en wordt er na iedere labozitting een tijdelijk portfolio verwacht, uiterlijk één week na de betreffende zitting.

Dit portfolio:

- is een verzorgd PDF document (layout, taalgebruik,…).
- is meer dan een verzameling screenshots en code.
- bevat een overzicht van wat je gedaan hebt, welke fouten je ontdekt hebt, hoe je die fouten ontdekt hebt. Beschouw het als een verslag aan een leek (die wel iets kent van VHDL, maar de opdracht niet kent), die moet begrijpen wat er gedaan en bereikt werd.
- bevat een overzicht van de bestede tijd per onderdeel (bv. simulatie, begrijpen van de code, fout zoeken, schrijven van portfolio, enz.).
- bevat relevante screenshots van de drie debug-methodes (zie "Doel van deze module").

Verdediging portfolio

Na inlevering van het portfolio zal er nog een moment afgesproken worden waarop iedereen individueel zijn portfolio moet komen toelichten. Hierbij zullen extra vragen gesteld worden om te achterhalen of:

- het portfolio volledig eigen werk is (niet gekopieerd)
- de inhoud van het portfolio diepgaand genoeg gekend is: Waarom werkt dit niet? Waarom gebruik je niet dat? Wat zou er gebeuren indien…? enz.
- de gebruikte tools diepgaand genoeg gekend zijn: Hoe werkt ILA? Wanneer gebruik je welke debug-methode? enz.
- de I^2C bus diepgaand genoeg gekend is.

Opdrachten

1) Er wordt VHDL-code gegeven (op Blackboard) voor een I^2C master. Deze code zal echter fouten bevatten, waardoor het geheel gedebugd zal moeten worden.

   Dit MOET gebeuren aan de hand van elk van volgende werkwijzen:

   - via simulatie (Xilinx Vivado)
   - via oscilloscoop/logic analyzer (LabNation SmartScope)
   - via ingebouwde FPGA testlogica (Xilinx ILA)

   Neem van alle uitgevoerde werkwijzen voldoende screenshots waarop duidelijk te zien is wat het probleem is. Leg ook duidelijk uit wat er gewijzigd moest worden (in de VHDL code) om het probleem op te lossen.

   Merk op dat de basiswerking van het I^2C protocol gekend moet zijn om deze VHDL code volledig te kunnen begrijpen. Anderzijds kan de VHDL code een hulpmiddel zijn om de basiswerking van het I^2C protocol te doorgronden.

2) In een tweede fase moet er een zinvolle communicatie via I^2C opgezet worden tussen de I^2C (PMOD) module en het FPGA bord. Met zinvolle communicatie wordt zowel lees- als schrijfopdrachten bedoeld naar enkele registers van de PMOD module. Als de PMOD module bijvoorbeeld een magnetisch kompas is, kan er naar het moderegister geschreven worden, de X-, Y- en Z-registers uitgelezen worden, enz. Afhankelijk van de gebruikte PMOD module, zal dit verschillen. Spendeer niet teveel tijd aan het “correct” gebruik van de PMOD module (kalibratie, dataverwerking, enz.), maar zorg ervoor dat je minstens kan lezen en schrijven naar één of meerdere registers. In I^2C termen wil dit dus zeggen: **minstens een read, write en burst read**.

   Hiervoor zal je online documentatie moeten opzoeken over je specifieke PMOD/I^2C module. Lees deze goed, zodat je weet welke registers je kan lezen en/of schrijven én wat het I^2C slave adres is van je module. Vergeet ook niet dat de meeste modules nog **externe pull-up weerstanden** nodig hebben op de SDA en SCL lijnen!

3) In een laatste (extra, uitbreidings-) fase kan nog een zogenaamde "bridge" gemaakt worden tussen I^2C en UART, zodat de PMOD module via I^2C communiceert met het FPGA bord en het FPGA bord communiceert met de PC/laptop via UART. In het voorbeeld van het kompas zou de PC dan via UART commando's moeten kunnen sturen (door de FPGA vertaald naar I^2C commando's) naar het PMOD kompas. Ook hier is de ruwe data het allerbelangrijkste, niet het correcte gebruik van de PMOD module. Als je via UART een I^2C register kan (burst) lezen en schrijven, is de opdracht al geslaagd.