Tutorial Schematic Entry met Quartus Prime Lite 20.1.1 en Intel ModelSim Starter 2020.1

DE HAAGSE HOGESCHOOL

J.E.J. op den Brouw
De Haagse Hogeschool
Opleiding Elektrotechniek
26 april 2022
J.E.J.opdenBrouw@hhs.nl

Inhoudsopgave

Voor suggesties en/of opmerkingen over deze tutorial kan je je wenden tot J. op den Brouw, kamer

D1.047, of je kunt email versturen naar J.E.J.opdenBrouw@hhs.nl.

Lijst van figuren

Lijst van tabellen

Listings

1. Inleiding

Vroeger was het de gewoonte om schakelingen op te bouwen uit losse componenten zoals transistoren, weerstanden en condensatoren. Naarmate de schakelingen complexer werden nam echter de kans op slechte verbindingen toe en de betrouwbaarheid af. Daarom is men er toe over gegaan meerdere componenten op één siliciumchip te integreren; voor digitale schakelingen begon dit met poorten en flipflops (Small Scale Integration), ging verder met tellers, decoders, multiplexers et cetera (Medium Scale Integration), en het einde is met de geavanceerde microprocessors en geheugens nog niet in zicht (Very Large Scale Integration). Het inwendige van zo'n geéntegreerde schakeling is echter niet te veranderen; de functionaliteit ligt vast. Een nieuwe generatie van componenten, de zogenaamde configureerbare logica, biedt de mogelijkheid zelf een schakeling te ontwikkelen. Deze componenten bevatten een groot aantal basisschakelingen (poorten, flipflops en losse verbindingen) die door de gebruiker willekeurig met elkaar en met in- en uitgangspennen kunnen worden verbonden. Bovendien zijn er recepten om bijvoorbeeld in één klap een gehele 16-bit teller te configureren (bibliotheekelementen). De taak van een ontwerper verschuift dus van solderen naar beschrijven.

Hoe gaat dit nu in zijn werk? Het eigenlijke beschrijven gebeurt met behulp van een PC. Dit is dus de onmisbare schakel in het geheel. De ontwerper bedenkt eerst een schakeling op papier. Als het probleem niet in één keer te overzien is, worden er deelontwerpen gemaakt die onderling verbonden zijn. Elk deelontwerp bevat een schakeling of, als het probleem nog niet te overzien is, weer deelontwerpen. We noemen dit principe hiërarchisch ontwerpen.

Nadat alle deelontwerpen zijn bedacht wordt overgegaan tot het invoeren van de deelschakelingen. Hier hebben we een verscheidenheid aan keuzes. We kunnen schakelingen invoeren als een schema met behulp van een tekenpakket, maar ook met behulp van een speciale taal die beschrijft wat de schakeling moet doen, bijvoorbeeld VHDL. Daarnaast zijn er nog invoermogelijkheden via toestandsmachines, booleaanse functies en waarheidstabellen.

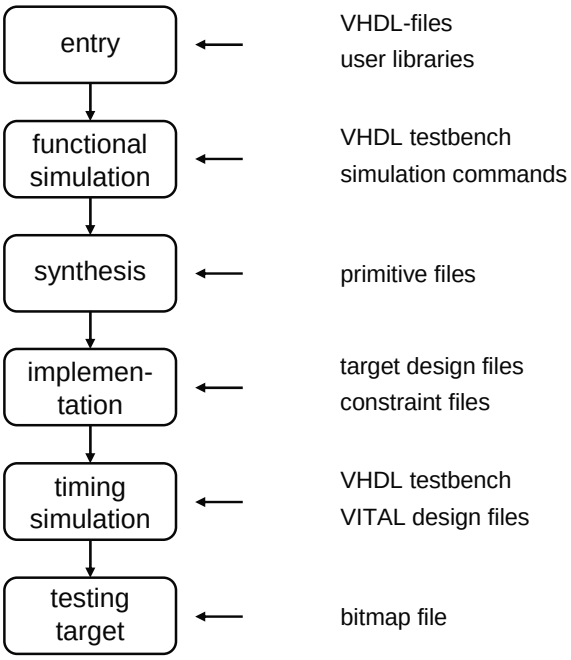
Wanneer alle deelontwerpen gemaakt zijn, moeten ze worden omgezet naar een bestand met gegevens die in de component geladen moet worden. Dit proces heet synthetiseren. Als tijdens dit omzetten een fout wordt geconstateerd, bijvoorbeeld twee uitgangen aan elkaar, wordt dit gemeld aan de ontwerper en moet de fout hersteld worden. Treden er geen fouten op dan levert de software een bruikbaar configuratiebestand op. LET OP: dit betekent nog niet dat het ontwerp precies doet wat het moet doen! Er kan nog best een functionele fout in het ontwerp zitten. Denk hierbij aan een programmeertaal. De compiler vindt geen syntax-fouten maar dat geeft geen garantie dat het programma doet wat het moet doen.

De laatste stap is het daadwerkelijk configureren van de component. Daarvoor is een hardwareprogrammer nodig. Die zorgt ervoor dat het configuratiebestand in de configureerbare component wordt gestopt.

Het ontwerptraject (een af te leggen weg van handelingen) wordt nu als volgt:

- bedenken van de schakeling,
- invoeren van het ontwerp met poorten,
- functionele simulatie
- omzetten van de poortschakeling naar een voor de configureerbare component geschikt bestand, eventuele fouten moeten eerst aangepast worden,
- eventueel timing simulatie
- testen van de schakeling.

In figuur 1.1 is het ontwerptraject nog eens schematisch aangegeven.



Figuur 1.1: *Het ontwerptraject.*

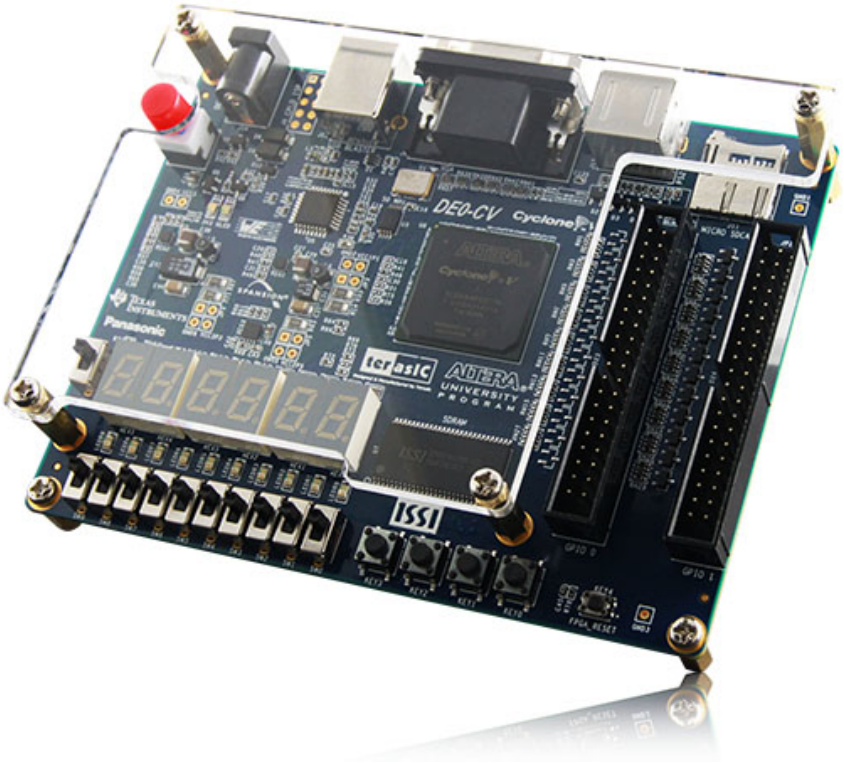
Het bedenken van de schakeling is een creatief proces. Ervaring en goede kennis van digitale systemen helpt je hierbij op weg. Het opzetten van een goede simulatie hoort bij de vaardigheden van de ontwerper. De rest wordt eigenlijk door de tools van Quartus afgehandeld. Het is voor de ontwerper niet meer interessant om op laag niveau de hardware te bekijken. Je moet er dan wel zeker van zijn dat je de hardware met de juiste regels beschreven hebt: combinatoriek en flankgevoelige geheugenelementen. Latches zijn uit den boze.

2. Practicumomgeving

In dit hoofdstuk wordt de practicumomgeving toegelicht.

2.1 Ontwikkelbord

Het practicum maakt gebruik van een ontwikkelbordjes, de DE0-CV. Op dit bordje is een FPGA van Intel geplaatst. Een foto is te zien in figuur 2.1.



Figuur 2.1: *Het DE0-CV-ontwikkelbord.*

Daarnaast zijn ook nog schakelaars, leds en 7-segment displays aanwezig. Zie hoofdstuk 3 voor meer informatie.

Naast het bordje wordt een softwarepakket van de fabrikant gebruikt genaamd Quartus Prime Lite. In de volgende paragraaf wordt een korte beschrijving gegeven van de software. Je hoeft niet alles in één keer te kennen. Verderop in deze handleiding is een tutorial opgenomen die je stap voor stap door het ontwerptraject loodst.

Om de component te configureren is een (hardware-)programmer nodig. Deze is op het ontwikkelbord geplaatst. Er is alleen een USB-kabel nodig voor een verbinding tussen een PC en het ontwikkelbord.

Quartus Prime Lite
Quartus Prime Lite is een alles-in-één pakket voor het ontwikkelen van digitale schakelingen en het configureren van (Intel) componenten. Het bestaat uit een viertal delen:

- het invoergedeelte - d.m.v. schema's, VHDL, toestandsdiagrammen;
- synthesizer - dit deel vertaalt de invoer naar een netlist;
- implemention - genereert een bit-file die je in de component kunt laden;
- programmer - dit deel configureert de component via de USB-interface.

ModelSim
ModelSim is een VHDL-simulator die direct vanuit de broncode simuleert. Er zijn in principe geen tussenstappen nodig zoals synthese. Je kan echter ook de uitvoer van de synthese simuleren. Hierdoor krijg je inzicht in de vertragingstijden. Met ModelSim is het mogelijk abstracte beschrijvingen van digitale schakelingen te simuleren. Deze schakelingen zijn niet synthetiseerbaar.

ModelSim kan als stand-alone pakket gebruikt worden. Wij zullen ModelSim gebruiken als onderdeel van Quartus en ModelSim starten vanuit Quartus.

2.2 Software-versies en Lite-editie

Het Quartus-pakket komt in twee smaken. Er is een volledige betaalde versie waarbij een licentie-server nodig is en er is een zogenaamde *Lite Edition*. De eerstgenoemde is de meest krachtige versie: alle *devices* van Intel zijn hiermee te configureren. Daarnaast heb je hier nog optie-pakketten voor DSP-ontwikkeling en digitale filters. De Lite Edition is gratis, heeft geen licentie-server nodig, maar kan niet synthetiseren voor alle beschikbare IC's. Er zijn geen optie-pakketten beschikbaar.

Van ModelSim bestaan ook twee versies: de volledige, betaalde Intel-versie en de zogenoemde *Intel Starter Edition*. De typering "Intel" geeft aan dat het specifiek ontwikkeld is om met Quartus samen te werken. De betaalde versie heeft geen beperkingen, de Intel Starter Edition kan maximaal 10000 VHDL-coderegels simuleren en verwerkt de code langzamer dan de betaalde versie.

De Lite Edition van Quartus (met ModelSim geïntegreerd) kan gevonden worden op

https://fpgasoftware.intel.com/20.1.1/?edition=lite&platform=windows#tabs-1

De software draait op Windows™ én Linux™, er is geen macOS-versie. Op het practicum wordt Windows gebruikt.

Let erop dat het practicum wordt uitgevoerd met versie 20.1.1. resp. versie 10.1d.

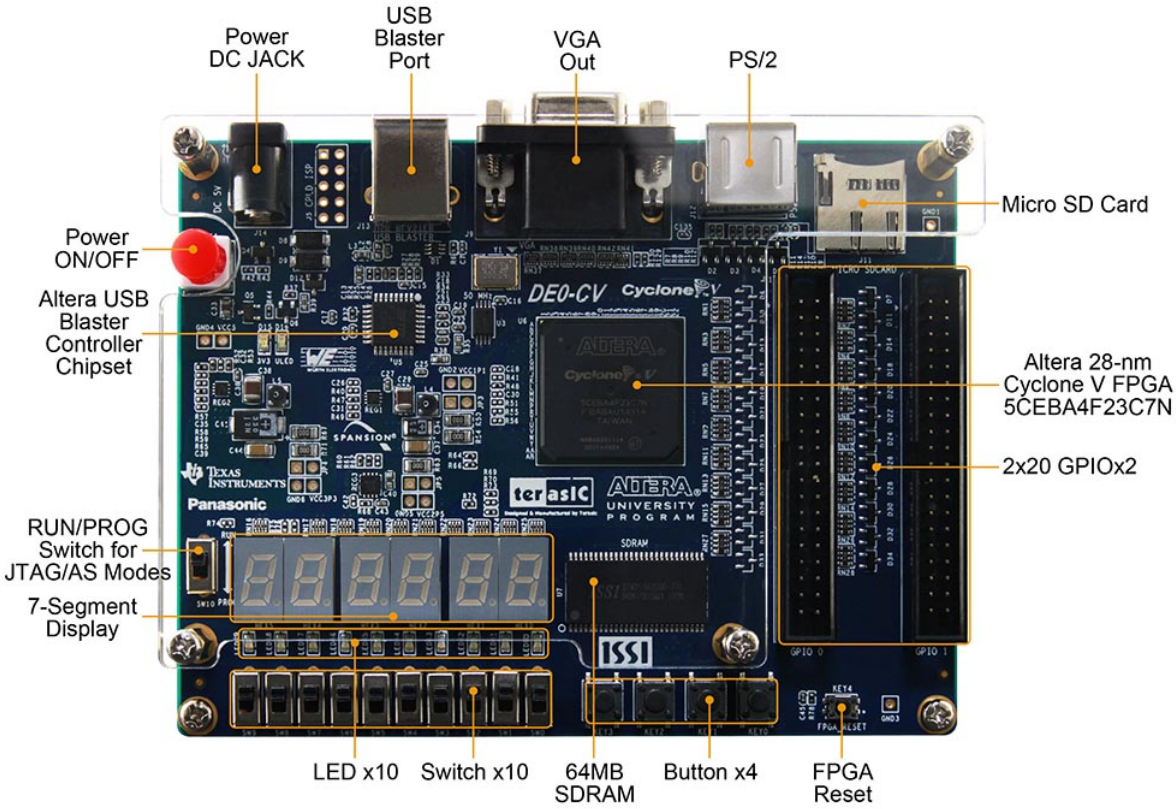
Noot: aangeraden wordt om versie 20.1.1 te installeren. Hogere versies gebruiken een andere simulator en daar is een (gratis) licentie voor nodig. Het opvragen en installeren van deze licentie is erg omslachtig.

3. Ontwikkelbordje

In dit hoofdstuk wordt het ontwikkelbordje nader beschreven. Eerst wordt een blokschema getoond en worden de diverse onderdelen kort beschreven. Daarna volgt een paragraaf over de configureerbare component, de Intel Cyclone V, met specificaties van het gebruikte type.

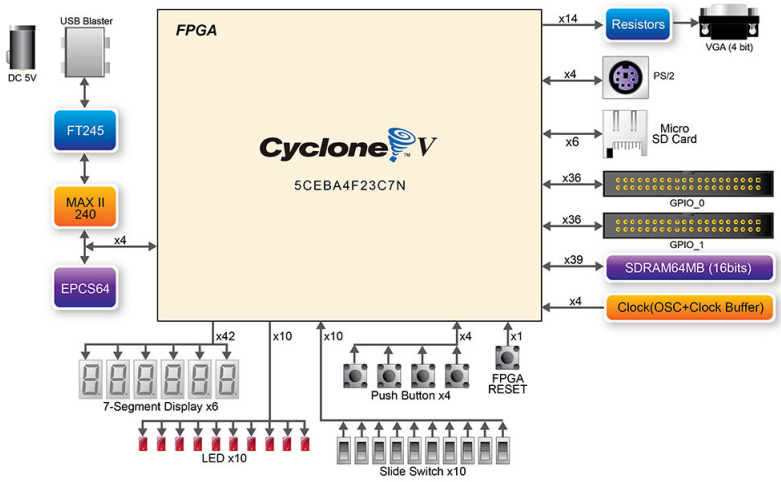
3.1 Blokschema

Het ontwikkelbord is ontwikkeld door het bedrijf Terasic in samenwerking met Intel, de fabrikant van de Cyclone V. Een gedetailleerde beschrijving is niet nodig; we geven slechts een blokschema van de onderdelen. Alles wat je nodig hebt tijdens het practicum staat hier beschreven. In figuur 3.1 is een foto afgebeeld met de diverse onderdelen.



Figuur 3.1: *Het DE0-CV-ontwikkelbord met benoeming van periferie.*

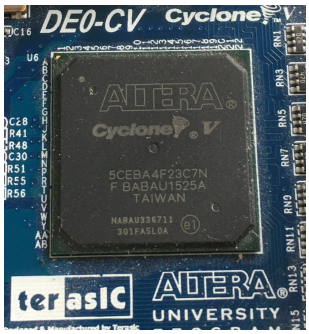
In figuur 3.2 is een blokschema weergegeven. Bij elk onderdeel staat vermeld met hoeveel signalen het onderdeel verbonden is met de Cyclone V. De voedingslijnen zijn niet meegenomen.



Figuur 3.2: *Blokschema ontwikkelbord.*

Cyclone V

Het hart van het ontwikkelbord wordt gevormd door de Cyclone V 5CEBA4F23C7N (zie figuur 3.3). Dit is een configureerbaar IC waarin een digitale schakeling kan worden geplaatst.



Figuur 3.3: *Foto Cyclone V.*

Push buttons

Het ontwikkelbord heeft vijf drukknoppen genaamd KEY3 t/m KEY0 en FPGA_RESET. Deze drukknoppen geven een laag logisch niveau (0) af als de knop is ingedrukt en een hoog logisch niveau (1) als de knop niet is ingedrukt. Verder zijn deze vijf knoppen ontdenderd en zijn te gebruiken als kloksignaal. FPGA_RESET wordt doorgaans gebruikt om de FPGA te resetten bij kloksynchrone schakelingen.

Slide switches

Het ontwikkelbord heeft tien schuifschakelaars genaamd SW9 t/m SW0. Een schakelaar geeft een laag logisch niveau (0) af als de schakelaar naar beneden is geschoven (het dichtst bij de rand van de print) en een hoog logisch niveau (1) als de schakelaar naar boven is geschoven. Deze schakelaars zijn niet ontdenderd en zijn alleen voor niveaugevoelige ingangen bedoeld.

Leds

Het ontwikkelbord heeft tien groene leds LEDG9 t/m LEDG0. Een led brandt als een hoog logisch niveau (1) wordt aangeboden en is uit als een laag logisch niveau (0) wordt aangeboden.

7-segmenten displays
Het ontwikkelbord heeft zes 7-segmenten displays waarmee getallen van verschillend formaat kunnen worden gemaakt. Elk display bestaat uit zeven leds waarmee een cijfer kan worden gevormd. Let op: de Decimal Point (DP) is niet verbonden met de FPGA. Een led brandt als een laag logisch niveau (0) wordt aangeboden en is uit als een hoog logisch niveau (1) wordt aangeboden.

Clock
Het ontwikkelbord heeft één 50 MHz klokoscillator aan boord. Het kloksignaal kan dienen als (direct) kloksignaal voor het klokken van flipflops of als invoer voor een Phase Locked Loop (PLL).

Een overzicht van de pinaansluitingen en de layout van de 7-segment displays is te vinden in bijlage A.

Overige aansluitingen
Het ontwikkelbord bevat verder nog een 64 MB SDRAM, een VGA-uitgang, een SD-card interface, een PS/2-interface en een expansion header met 64 I/O-lijnen en 8 kloklijnen. Dit wordt verder niet besproken.

3.2 Intel Cyclone V

De Cyclone V FPGA (Field Programmable Gate Array)[1] is opgebouwd uit zogenaamde logische elementen (Logic Elements, LE). Met deze elementen kan je een digitale schakeling bouwen. Het gebruikte type heeft er 18480 aan boord. Om een indruk te geven wat dat inhoudt: een volledige 32-bits processor (NIOS/IIf) gebruikt zo'n 867 elementen. Je kan er dus 20 processoren in kwijt.

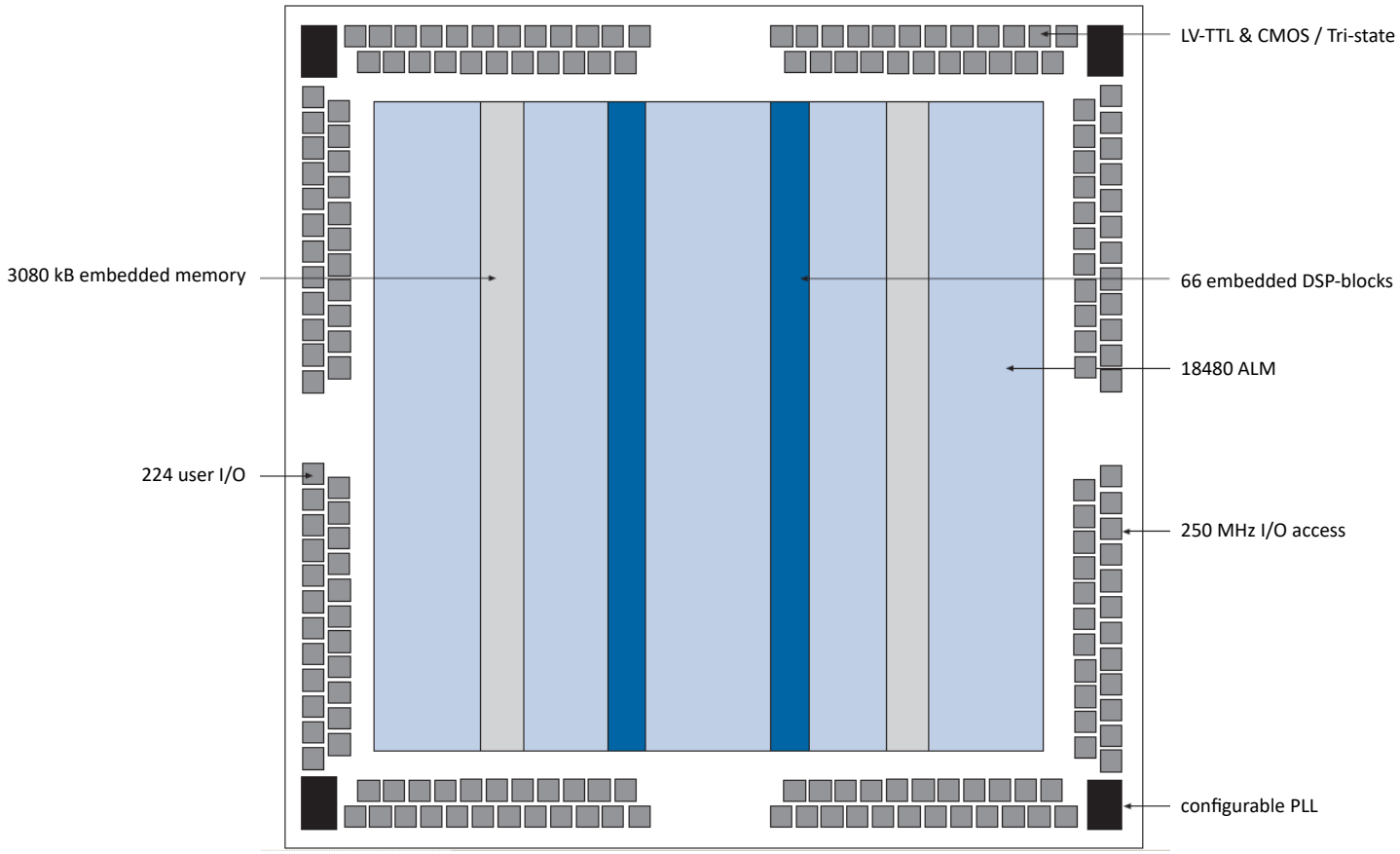
In tabel 3.1 zijn wat gegevens te vinden.

Tabel 3.1: *Enige gegevens van de 5CEBA4F23C7N.*

Aansluitpinnen (user I/O)	484 (224)
Logische elementen (ALM)	18480
Geheugenelementen	73920 (vier per ALM)
RAM-bits	3153920
DSP-blokken	66
Phase Locked Loops	4
Global Clock Networks	20
$t_{PD(pin-to-pin)}$	7 ns
f_{MAX} (I/O, stand alone)	250 MHz

Figuur 3.4 geeft een Floor Plan van de bij de workshop gebruikte 5CEBA4F23C7N. Daarnaast heeft elke Cyclone RAM en vermenigvuldigers aan boord. De vermenigvuldigers zijn sneller dan wanneer ze met LE's worden opgebouwd. Je kan ze bijvoorbeeld gebruiken bij digitale signaalbewerking.

[1] Het is wel raar dat een FPGA configureerbaar heet, terwijl de naam suggereert dat ze programmeerbaar zijn.



Figuur 3.4: *Floor Plan van de Cyclone V.*

Adaptive Logic Module

Elke Adaptive Logic Module (ALM) bestaat uit een aantal look-up table (LUT) en vier D-flipflops. De ALM kan een combinatorische of sequentiële functie vervullen. Een LUT kan elke combinatorische schakeling van een aantal variabelen nabootsen, de D-flipflop kan één bit onthouden. Indien je schakeling te groot is om in één ALM te stoppen wordt dat door de software (synthese, mapper) verdeeld over meerdere ALM's.

Routing en interconnect

Elke ALM kan maar een klein deel van schakeling bevatten. Een schakeling zal dus uit meerdere ALM's bestaan. Tussen de ALM's is dus informatie-uitwisseling nodig. Dat gebeurt door de routing en interconnect. Realiseer je dat zo'n 50% van het chipoppervlak alleen maar routing is! De vertragingstijd van combinatorische schakelingen komt voor 2/3 voor rekening van de routing! Binnen een ALM's is snelle interconnect mogelijk.

I/O Banks

De I/O banks (deze chip heeft er acht) zijn verantwoordelijk voor verbindingen tussen de buitenwereld en het interne gedeelte van de chip. Het levert de externe signalen netjes af aan de routing en signalen van routing worden netjes aan de buitenwereld afgeleverd. Tot de mogelijkheden horen: LV-TTL, LV-CMOS input, tri-state ouput, programmable slew rate.

4. Tutorial Schematic Entry

In deze tutorial gaan we ons bezighouden met het invoeren, simuleren en implementeren van digitale schakelingen met schematische invoer. Andere stappen zoals het aanmaken van een project en pintoewijzing worden in een andere tutorial behandeld.

Zoals bekend kan elke digitale schakeling worden opgebouwd met de bekende poorten zoals AND, OR en NOT. Daarnaast zijn er veel andere poorten zoals de EXOR, NAND en NOR. Tijdens deze tutorial leer je hoe je een schema dat is opgebouwd uit poorten moet invoeren. Daarnaast maak je kennis met hiërarchisch ontwerpen: het schema van een deelschakeling kan je gebruiken als een "poort" in een ander schema. Hierdoor kan je snel grotere schakelingen maken. Het ontwerpen van digitale schakelingen met poorten is een traject apart en komt niet in deze tutorial aan bod.

De tutorial behandelt slechts een klein gedeelte van alle mogelijkheden die in Quartus en ModelSim voor handen zijn. Je zal zelf meer functies moeten onderzoeken die je voor de opdrachten nodig hebt.

We zullen de volgende stappen doorlopen:

- project openen;
- schema-invoer d.m.v. Schematic Editor;
- simuleren van de schakeling op functioneel niveau;
- compilatie (synthese en implementatie) van de poortschakeling naar een configuratiebestand;
- downloaden van de configuratiebestand in de Cyclone V.

Deze tutorial zal je stap voor stap door de diverse onderdelen van Quartus en ModelSim leiden waarna je zelf een aantal opdrachten moet uitwerken.

NB: gebruik geen spaties, leestekens of "vreemde"tekens in mapnamen en bestandsnamen! Bestandsnamen altijd beginnen met een letter!

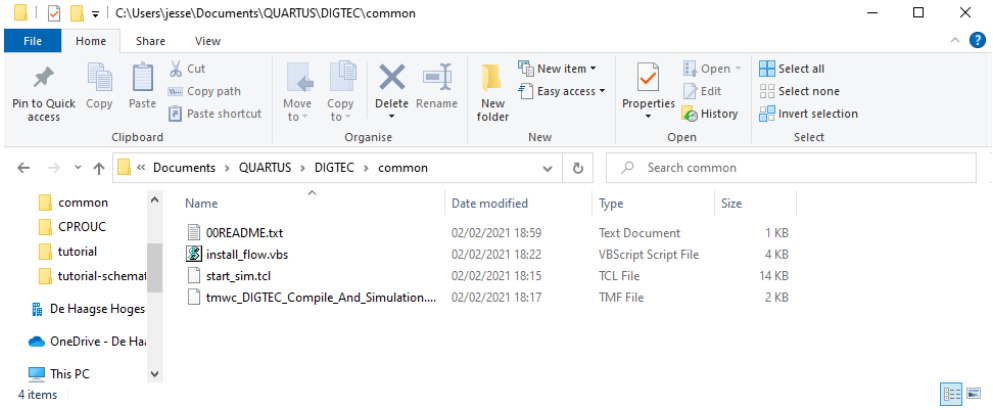
Noot: het doel van deze tutorial is het leren omgaan met de software (zogenaamde tools), niet het leren van digitale techniek. Over de werking van de schakeling die wordt ingevoerd (wat doet het) wordt geen uitleg gegeven.

4.1 Installatie project Tutorial

Voordat we echt kunnen beginnen, moeten we eerst de projectomgeving inrichten. Hiervoor moet je toegang hebben tot de BlackBoard Course DIGTEC. De bestanden kunnen ook gevonden

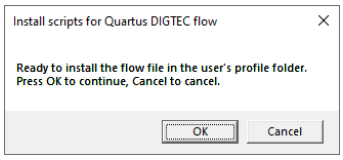
worden op https://ds.opdenbrouw.nl/digtec.html. Quartus is *niet* geïnstalleerd op de schoolcomputers. Je moet je eigen laptop gebruiken. Volg de onderstaande stappen om de het project te installeren.

1. Maak op je laptop een nieuwe map met de naam `QUARTUS` aan. Dat kan op de C:-schijf in de map `Documents`.
2. Maak in de map `QUARTUS` een map aan met de naam `DIGTEC`.
3. Download van BlackBoard[2] het bestand `digtec_common_tutorial.zip` en plaats dit bestand in de map `DIGTEC`.
4. Pak het bestand uit in deze map. Let op: bij uitpakken maakt Windows automatisch de map `digtec_common_tutorial` aan! Dat is niet de bedoeling!
5. Je vindt nu twee mappen: `tutorial` en `common`. Navigeer naar de map `common`. Hierin vindt je een bestand met de naam `install_flow.vbs`. Zie figuur 4.1.



Figuur 4.1: *De inhoud van de map* `common`.

6. Dubbelklik op het bestand `install_flow.vbs`. Er volgt nu een scherm. Zie figuur 4.2. Klik op **OK** om de flow te installeren.



Figuur 4.2: *Installeren flow.*

7. De flow wordt nu geïnstalleerd. Hierna volgt een melding. Klik op **OK** om de installatie af te sluiten.

Noot: het is ook mogelijk om de bestanden handmatig in een andere map te installeren, Zie de bijlage B.

4.2 Project en naamgeving

Een project bestaat uit niets anders dan een verzameling bestanden. De belangrijkste bestanden hebben de volgende extensies:

[2] Een alternatief is https://ds.opdenbrouw.nl/digtec/digtec_common_tutorial.zip

Tabel 4.1: *Betekenis bestandsnaamextensies.*

Extensie	Volledige naam	Betekenis
.qpf	Quartus Project File	Projectbestand met naam en algemene informatie
.qsf	Quartus Settings File	Instellingen bij een bepaalde *revisie*, meestal is er maar één revisie
.bdf	Block Design File	Schema-bestand, zoals poorten, en componenten waardoor hiërachiëen mogelijk zijn.
.bsf	Block Schematic File	Een component ("poort") gemaakt van een bdf-bestand of vhd-bestand
.do	ModelSim Command File	Bestand met commando's voor de simulator
.vhd	VHDL File	Bestand met VHDL-code
.v	Verilog File	Bestand met Verilog-code, wordt tijdens het practicum niet gebruikt
.sof	SRAM Object File	Bestand met "code" voor de FPGA, informatie die in de FPGA geladen moet worden

Quartus maakt in het simulatie- en synthesetraject nog veel meer bestanden met extensies aan, die zijn niet van belang. ModelSim-testbenches beginnen met `tb_`.

Een project kan automatisch in Quartus worden geopend door de dubbelklikken op het betreffende .qpf-bestand.

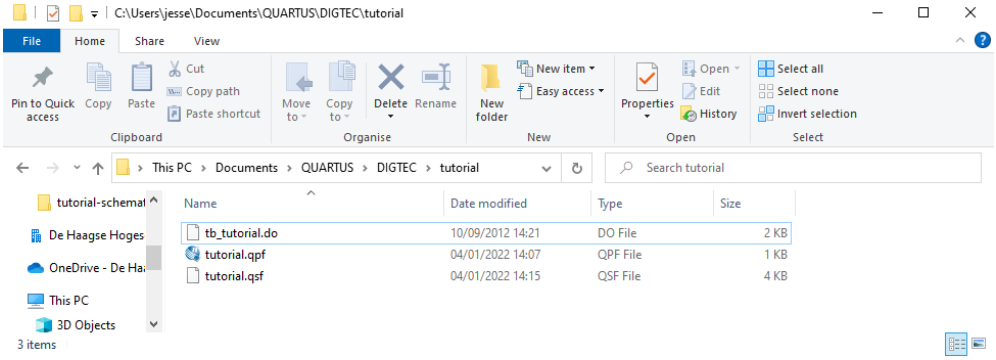
Niet op een .bdf-bestand dubbelklikken, want dan is er geen projectomgeving beschikbaar en kan er niet gesimuleerd of gesynthetiseerd worden.

Gebruik geen spaties, leestekens of "vreemde" tekens in bestandsnamen en mapnamen!

Gebruik geen minteken in bestandsnamen, underscores zijn wel toegestaan.

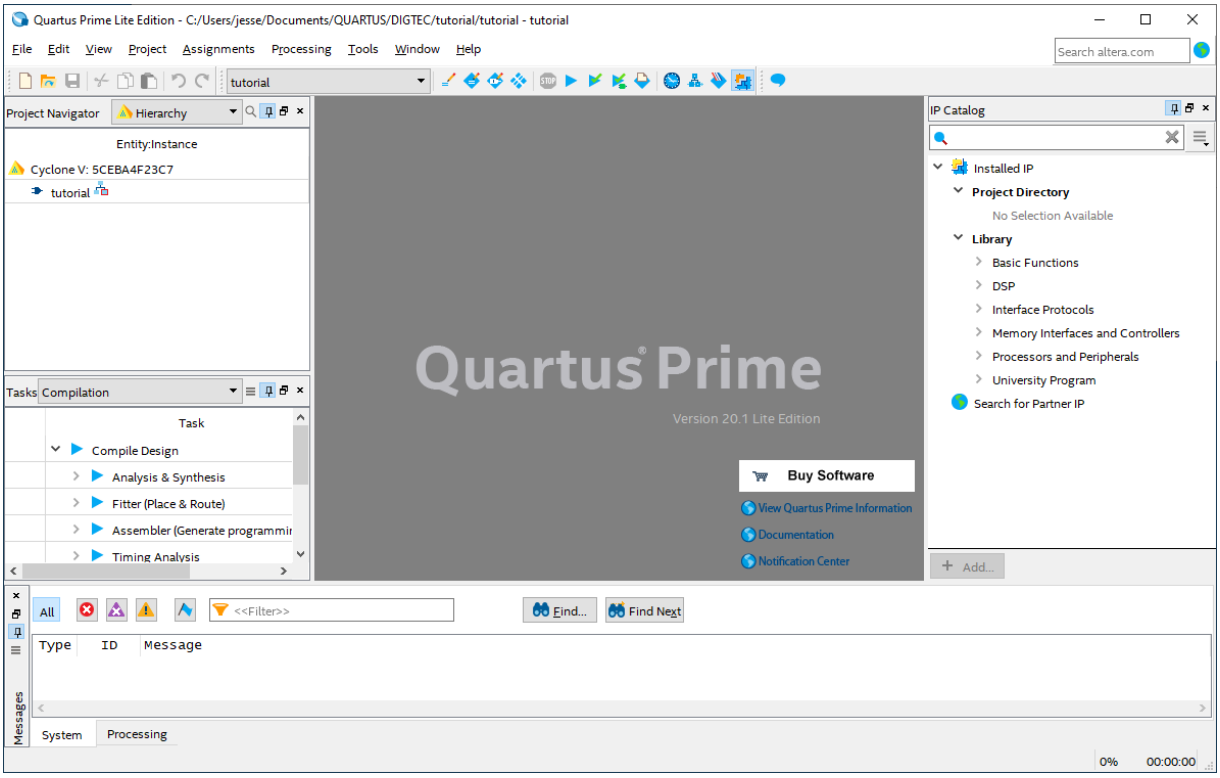
4.3 Project Tutorial starten

We starten Quartus op via Explorer. Navigeer naar `tutorial`-map en dubbelklik op het bestand `tutorial.qpf`. Zie figuur 4.3.



Figuur 4.3: *Inhoud van de map* `tutorial`.

De *Project Manager* wordt gestart (figuur 4.4). De grootte en indeling van het scherm kan iets afwijken van de figuur. Probeer dit aan te passen volgens de figuur.



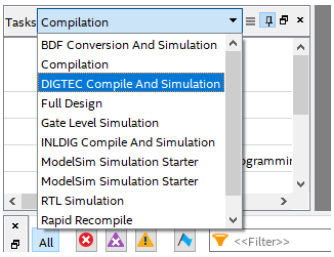
Figuur 4.4: *Openingsscherm Project Manager*

De *Project Manager* bestaat uit menu's, knoppenbalk en en een aantal vensters. Linksboven is de *Project Navigator* te zien. Hierin worden alle (broncode-)bestanden en de onderlinge afhankelijkheden zichtbaar gemaakt, zoals hiërarchieën en testbenches. Daaronder is de *Tasks*-venster te zien. Hierin worden de mogelijke taken bij een geselecteerd bestand weergegeven, zoals synthetiseren of implementeren. In het midden is het *edit*-venster te zien waar bestanden bewerkt

en bekeken kunnen worden. Links is het venster te zien waarmee IP-cores kunnen worden geselecteerd (we gebruiken dit venster niet). Onderaan is het *Message*-venster te zien waar uitvoer van de diverse opdrachten wordt afgedrukt.

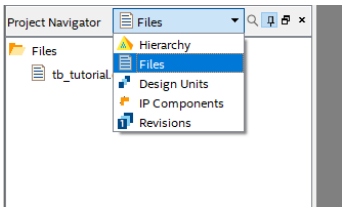
Om extra ruimte in het edit-venster te krijgen, kan je het beste de IP Catalog uitschakelen.

Voor de juiste werking moet eerst de *flow* worden ingesteld. In het venster Tasks kan je bij het onderdeel Flow kiezen voor een flow. Kies hier de flow DIGTEC Compile And Simulation. Zie figuur 4.5.



Figuur 4.5: *Selecteren van de DIGTEC-flow.*

Kies in het venster *Project Navigator* voor het tabblad Files. Hier zie je een overzicht van de bestanden die tot het project behoren. Het project bestaat uit slechts één bestand genaamd `tb_tutorial.do`. Zie figuur 4.6.

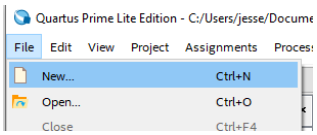


Figuur 4.6: *Het `Files`-tabblad.*

4.4 Aanmaken schemabestand

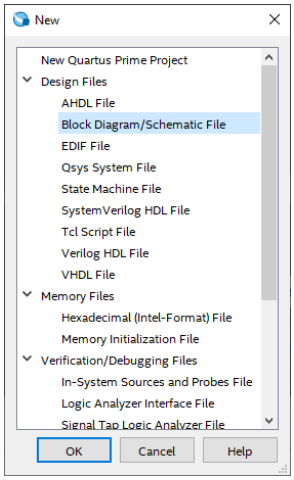
We gaan nu het eerste schema invoeren. Dit schema gaat later gebruikt worden als “poort” in een ander schema.

Als eerste zullen we een schemabestand in het project aanmaken. Klik in de Project Manager op **File→New** (figuur 4.7). Als alternatief kan de toetscombinatie **Ctrl+N** gebruikt worden.



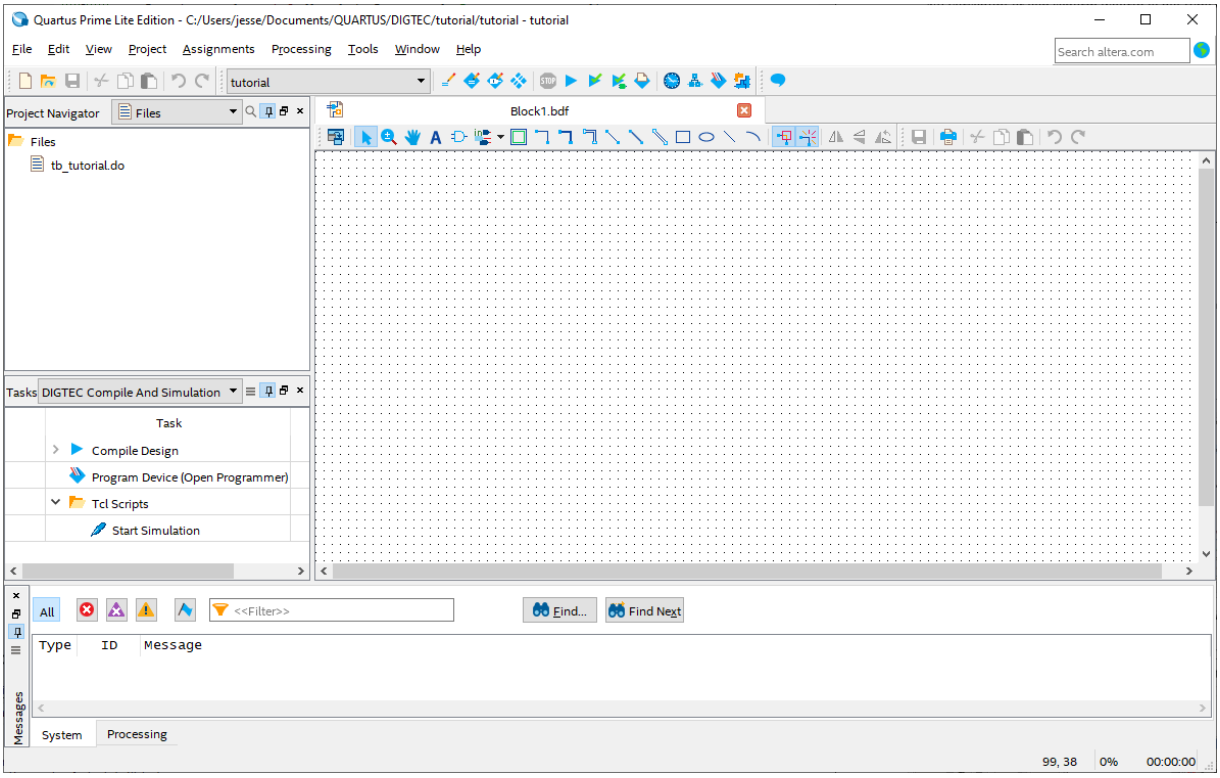
Figuur 4.7: *Aanmaken nieuw bestand.*

Nu verschijnt er een scherm waarin je het type van het nieuwe bestand kan kiezen (figuur 4.8). Kies voor Block Diagram/Schematic File.



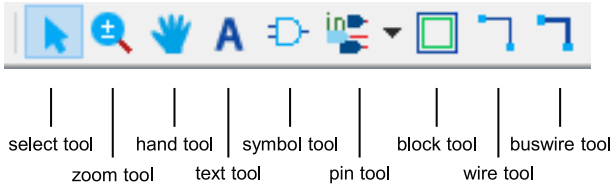
Figuur 4.8: *Keuze bestandstype.*

De Project Manager opent nu een leeg bestand. Dit is te zien in figuur 4.9. Merk op dat het bestand de tijdelijk naam Block1.bdf heeft. Het sterretje achter de naam geeft aan dat het bestand gewijzigd en nog niet opgeslagen is. Bij het opslaan van het bestand moet alsnog een andere naam worden ingevoerd.



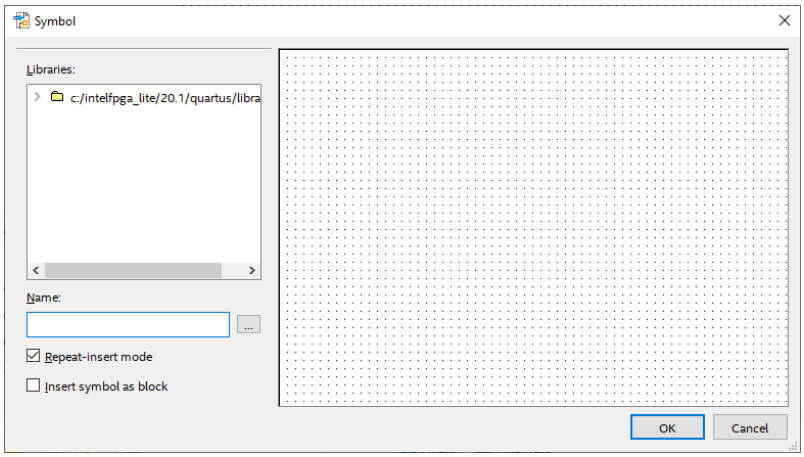
Figuur 4.9: *Overzicht Quartus IDE na aanmaken nieuw BDF-bestand.*

Bovenaan het edit-venster staat een rij knoppen. De twee belangrijkste zijn de poortinvoerknop (symbol tool, ziet eruit als een AND-poort) en de pin-invoerknop (pin tool). Zie figuur 4.10.



Figuur 4.10: *Overzicht van de knoppen.*

Klik nu op het AND-poortje. Er wordt een dialoogvenster geopend. Zie figuur 4.11.

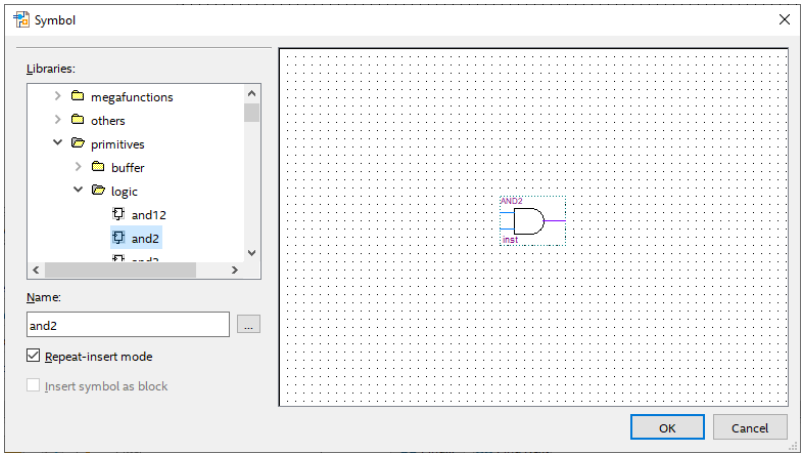


Figuur 4.11: *Selectie component.*

Linksboven staan de *libraries* (bibliotheken) vermeld. Hierin zijn de poorten opgenomen die we gaan gebruiken.

Noot: het kan zijn dat de het beeld in figuur 4.11 iets afwijkt. Dat is in de regel geen probleem.

Open nu de library tot op het niveau van logic en selecteer de AND2-poort. Zie figuur 4.12.



Figuur 4.12: *Selectie AND2-poort.*

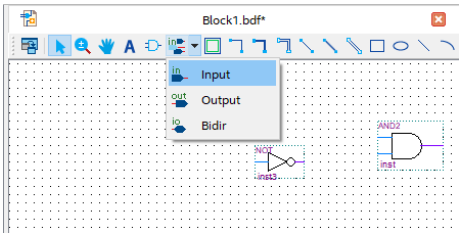
Klik op **OK**. De AND2-poort "hangt" nu aan de cursor. Plaats de poort ergens in het midden van het edit-venster.

Noot: na het plaatsen “hangt” er opnieuw een AND2-poort aan de cursor. Je kan er dus nog één plaatsen. Dit wordt de *Repeat-insert Mode* genoemd. Druk op de **ESC**-toets om dit af te breken.

Noot: in figuur 4.12 zie je ook andere bibliotheken. Gebruik alleen symbolen uit de logic-bibliotheek.

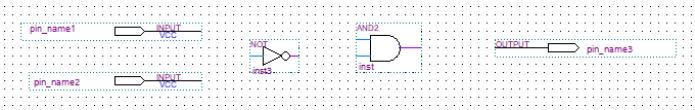
Voeg nu op gelijke wijze een NOT-poort in (ook inverter genoemd). Deze poort zit iets lager in dezelfde kolom.

Nu moeten de ingangs- en uitgangspoorten nog worden geplaatst. Dit kan op twee manieren: via de symbol tool (maar dan onder het niveau pin) of via de pin tool. Klik hiervoor op het symbool van pin tool pictogram en selecteer Input. Zie figuur 4.13.



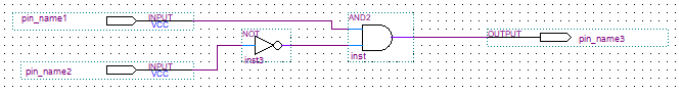
Figuur 4.13: *Selectie input-poort.*

Plaats nu twee input-pinnen in het edit-venster. Selecteer via de pin tool nu Output en plaats een output-pin. Zie figuur 4.14.



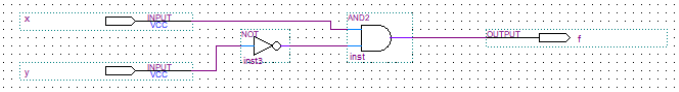
Figuur 4.14: *Alle geplaatste componenten.*

Nu moeten de diverse pinnen en poorten aan elkaar verbonden worden. Dat gaat vrij makkelijk. Plaats de cursor boven het eindpunt van de bovenste input-pin. Druk op de linker muisknop en houdt deze ingedrukt. Sleep nu naar *vlak voor* de bovenste ingang van de AND2-poort en laat de muisknop los. Er verschijnt nu een blauwe lijn. Maak nu een kleine lijn haaks naar beneden en vervolgens een lijn naar AND2-poort. Klik dan even in het edit-veld zodat de blauwe lijn gedeselecteerd wordt. De lijn wordt nu paars. Verbind vervolgens ook de andere componenten zoals aangegeven in figuur 4.15.



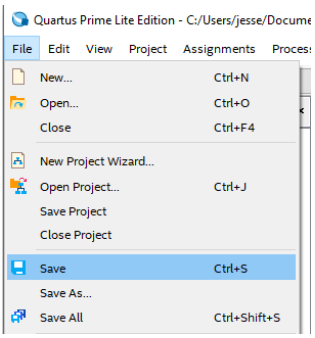
Figuur 4.15: *Alle geplaatste componenten met verbindingen.*

Als laatste stap moeten de ingangen en uitgangen nog een nieuwe naam krijgen. Dubbelklik precies op de naam pin_name1. De naam wordt nu geselecteerd. Verander dit in x (kleine letter). Vervang nu ook pin_name2 door y en pin_name3 door f. Het geheel is te zien in figuur 4.16.



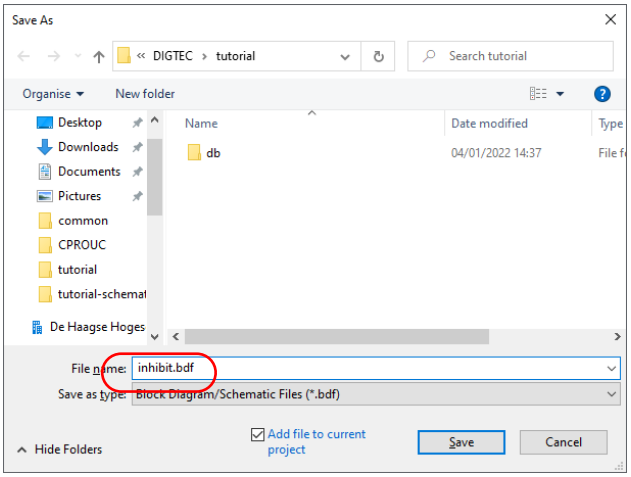
Figuur 4.16: *Schema met nieuwe pinnamen.*

Nu het schema is ingevoerd, moet het bestand opgeslagen worden. Klik in de Project Manager op **File**→**Save** (zie figuur 4.17).



Figuur 4.17: *Bestand opslaan.*

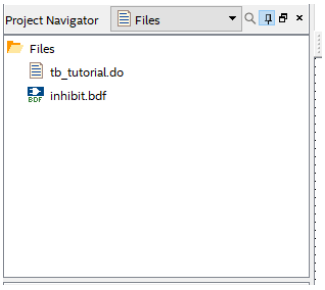
Er wordt een scherm geopend waarin de juiste map en de bestandsnaam ingevuld kunnen worden. Zie figuur 4.18. Er wordt een bestandsnaam voorgesteld, wijzig dit in inhibit.bdf. Let op het vinkje bij Add file to current project.



Figuur 4.18: *Opgeven bestandsnaam BDF-bestand.*

Noot: je mag geen spaties, leestekens of "vreemde" tekens in de bestandsnaam opnemen!
Noot: let goed op de map waarin het bestand opgeslagen wordt. Quartus wil nog wel eens de map van een eerder geopend project presenteren.
Noot: bestandsnaam *niet* met een cijfer beginnen!

Het bestand is nu terug te vinden in de Project Navigator onder het tabblad Files. Zie figuur 4.19.

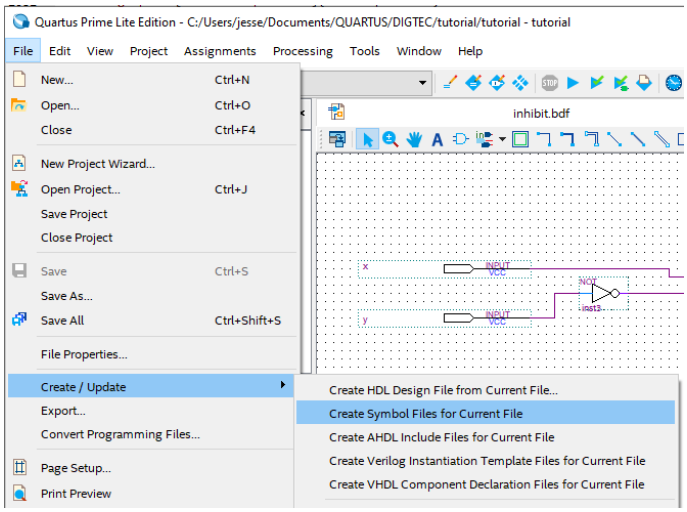


Figuur 4.19: *Het opgeslagen bestand staat in de lijst van bestanden.*

4.5 Aanmaken van symbool van het huidige bestand

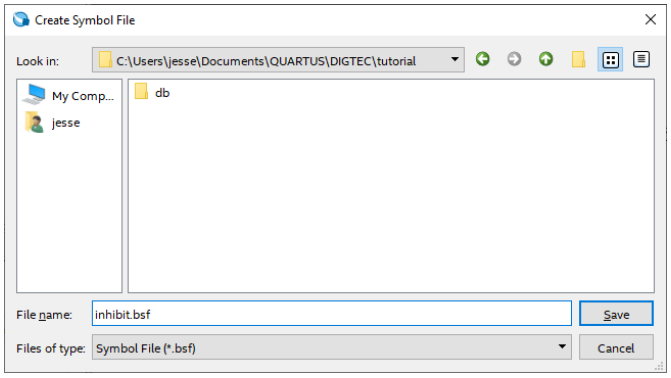
We gaan het schema dat net is opgeslagen in een ander schema gebruiken. Hiervoor moet eerst een *symbol* worden aangemaakt van het schema. Dit symbol komt dan terug in de *library*.

Open het bestand `inhibit.bdf` en, als het bestand geopend is, selecteer in het edit-venster het tabblad van het bestand. Selecteer via het menu **File→Create/Update→Create Symbol Files for Current File**. Zie figuur 4.20.



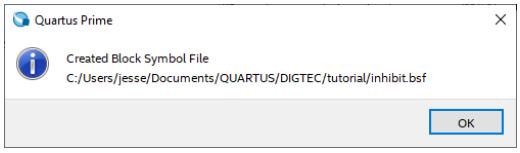
Figuur 4.20: *Aanmaken nieuw symbool.*

Er wordt een dialoogvenster geopend waarin een bestandsnaam kan worden ingevoerd. De voorgestelde naam inhibit.bsf is goed. Klik op **OK** om het bestand op te slaan. Zie figuur 4.21.



Figuur 4.21: *Bestandsnaam nieuw symbool.*

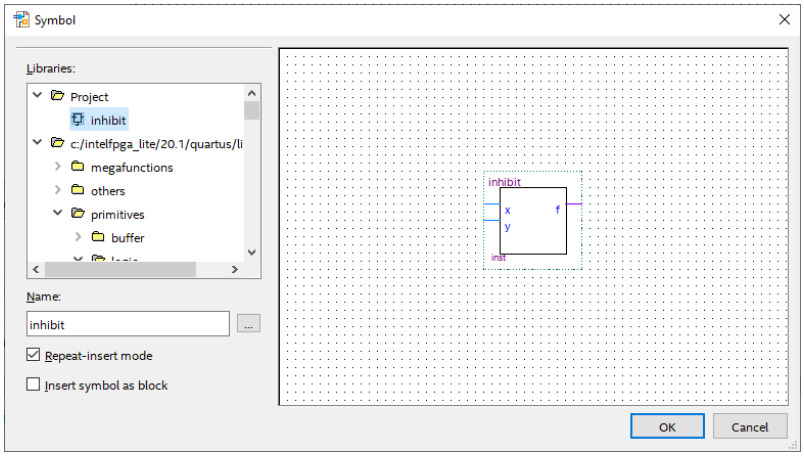
Quartus komt met de melding dat het bestand is aangemaakt. Zie figuur 4.22.



Figuur 4.22: *Het bestand is opgeslagen.*

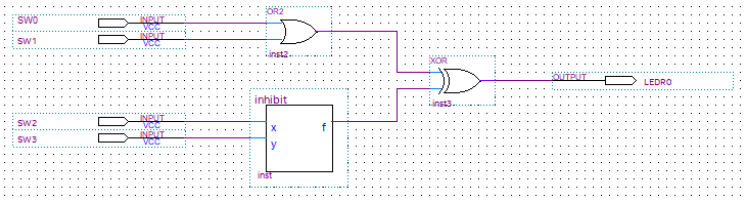
4.6 Aanmaken van een tweede schemabestand

We maken nu op dezelfde wijze een nieuw schemabestand aan (zie figuren 4.7 en 4.8). Plaats daarin de net aangemaakt “poort”. Deze kan je vinden onder onder de library Project. Zie figuur 4.23.



Figuur 4.23: *Selectie nieuwe component.*

Plaats deze “poort” ergens in het edit-venster. Maak het schema verder af zoals in aangegeven in figuur 4.24. Let goed op de poorten; er moeten een OR2- en een XOR-poort geplaatst worden.



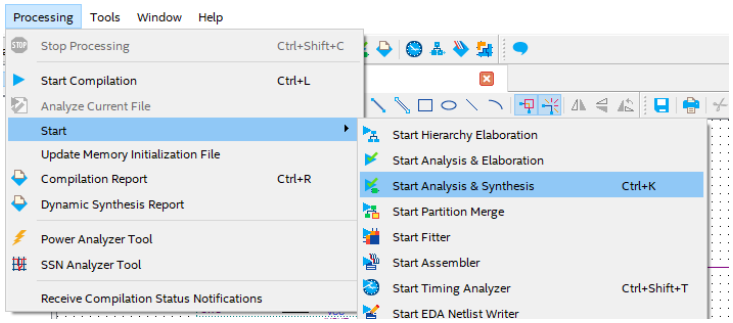
Figuur 4.24: *Het bestand is opgeslagen.*

Let goed op de namen van de ingangen en uitgang (hoofdletters!).

Sla het bestand op onder de naam tutorial.bdf. De naam van het bestand komt nu in de lijst van bestanden bij de Project Navigator.

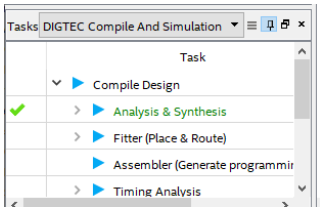
4.7 Eerste synthese

Om te controleren of het schema *syntactisch correct* is en er hardware voor kan worden gegenereerd, starten we de synthesizer. Klik in de Project Manager op **Processing**→**Start**→**Start Analysis & Synthesis** of gebruik de sneltoetscombinatie **Ctrl+K**. Zie figuur 4.25.



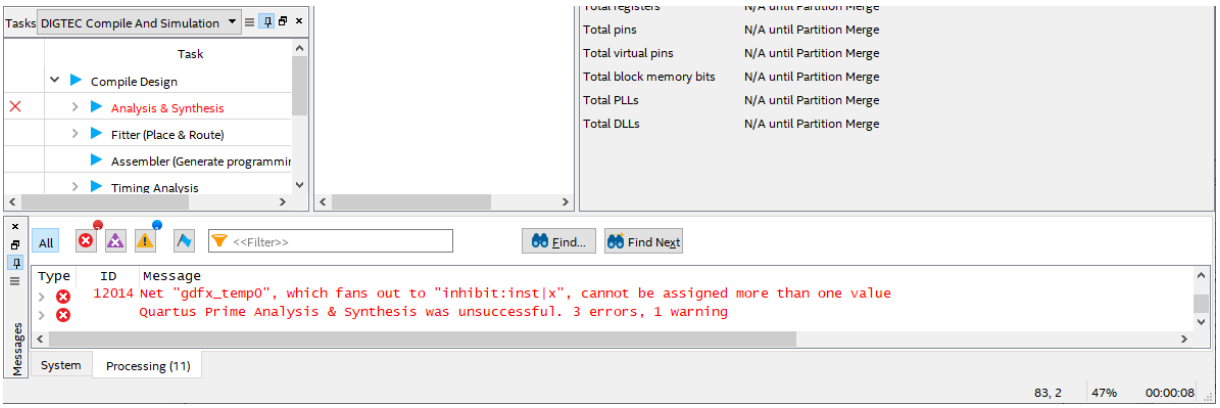
Figuur 4.25: *Starten Analysis & Synthesis vanuit het menu.*

Als deze stap gelukt is zie je onder Tasks een vinkje verschijnen (figuur 4.26).



Figuur 4.26: *De analyse is gelukt.*

Mocht je een fout hebben gemaakt, bijvoorbeeld twee ingangen aan elkaar verbonden, dan zal de synthesizer dat melden, zie figuur 4.27. Merk op dat de fout in de schakeling tutorial zit.



Figuur 4.27: *De analyse is mislukt.*

4.8 Simulatie poortschakeling

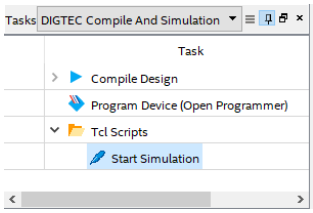
Nu de schema's ingevoerd zijn, moet het geheel gesimuleerd te worden. Dit droogzwemmen is bedoeld om te kunnen verifiëren of de schakeling, en dus de schema's, werkt volgens de specificaties. Quartus Prime Lite gebruikt hiervoor de externe simulator ModelSim van firma Mentor Graphics. We gebruiken de simulator om aan te tonen dat onze schakeling functioneel correct is. Het kan namelijk best zijn dat de schakeling gesynthetiseerd kan worden, maar dat de schakeling niet doet wat het zou moeten doen. Bij deze simulatie worden geen vertragingstijden meegenomen.

Voor simulatie is een zogenaamd scriptbestand nodig. Hierin staan opdrachten voor de simulator. Je kan deze opdrachten ook interactief op een commandoregel invoeren, maar vaak wil je

de simulatie een paar keer opnieuw draaien. Dan is het steeds invoeren van de (zelfde reeks) commando's een tijdrovende zaak. In het scriptbestand staat een aantal opdrachten die de simulator gebruikt om de ingangen mee aan te sturen. Dit zijn de zogenoemde *stimuli*. De simulator gebruikt deze waarden om de schakeling door te rekenen.

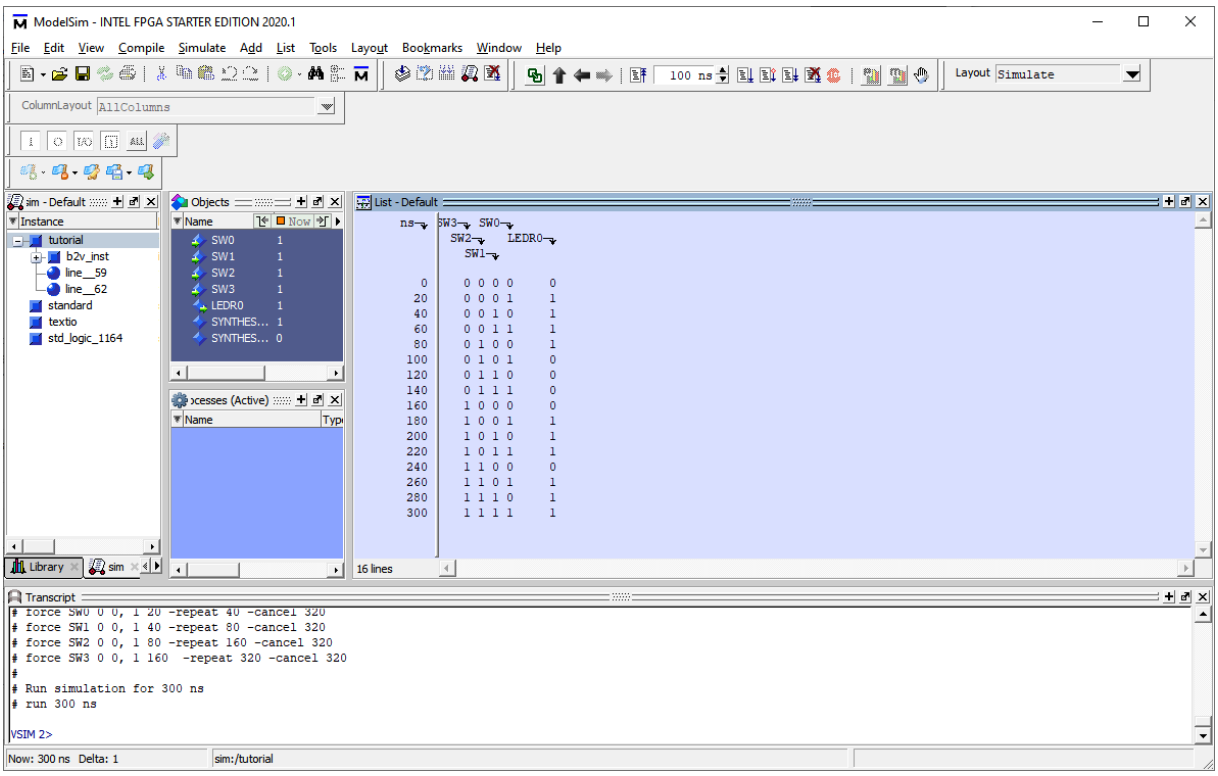
Noot: de commando's staan het het bestand tb_tutorial.do dat al is aangemaakt. Je kan de inhoud van het bestand bekijken, maar laat de inhoud ongewijzigd anders kan de simulatie mislukken.

Dubbelklik in het Tasks-venster op Start Simlation, zie figuur 4.28.



Figuur 4.28: *Starten van de simulatie.*

Tijdens het starten van ModelSim wordt een *splashscreen* getoond. De simulator wordt gestart (dit kost enige tijd) en er worden vijf vensters geopend (zie figuur 4.29).



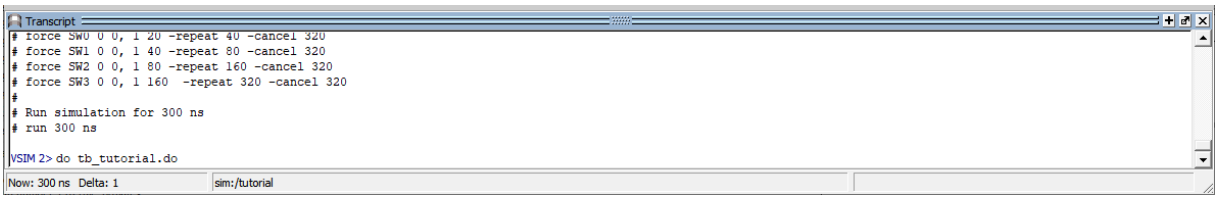
Figuur 4.29: *Het resultaat van de simulatie.*

Onderaan is de *Transcript Window*. Hier kan je ook losse commando's geven (voor gevorderden). Rechts is de *List Window*. De overige drie zijn op dit moment niet van belang.

In de List Window wordt de uitkomst van de simulatie weergegeven. Je ziet een aantal rijen. Een rij wordt voorafgegaan aan een getal (dat is een simulatietijd) en daarna vijf nullen of enen. De

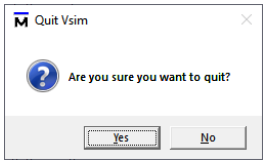
eerste vier stellen de waarden voor van de vier ingangen SW3 t/m SW0, de laatste de waarde van de uitgang LEDR0. Deze namen komen overeen met de schakelaars en de led van het ontwikkelbordje. Zie hoofdstuk 3.

Je kan het commando-script nogmaals uitvoeren door in de transcript window op de ↑-toets te drukken. Het laatst ingevoerde commando verschijnt dan. Druk op de **enter**-toets om dat commando uit te voeren. Zie figuur 4.30.



Figuur 4.30: *Het transcript-window.*

De simulatie is nu ten einde. Sluit de simulator af via het menu **File**→**Quit**. Er wordt een dialoogvenster geopend, zie figuur 4.31. Klik op **Yes**.

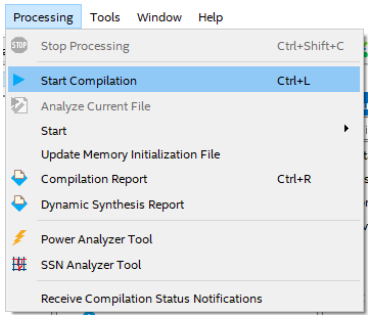


Figuur 4.31: *ModelSim afsluiten.*

4.9 Compilatie

Nu de simulatie uitgevoerd is, wordt de schakeling gecompileerd. Compileren valt uiteen in twee delen: synthese en implementatie. Synthese houdt in dat de schakeling wordt vertaald met als resultaat een *netlist*; een beschrijving van de digitale logica in primitieven. Denk hierbij aan poorten, flipflops, LUTs (LookUp Table, ROM) of speciale voorzieningen zoals Phase Locked Loops of klokbuffers. Deze primitieven zijn voor elk configureerbaar type weer anders. Implementatie houdt in dat de primitieven worden *gemapped* op de LE's.

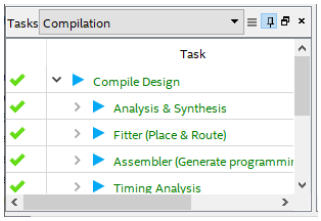
Start de compilatie via de menuoptie **Processing**→**Start Compilation** of gebruik de sneltoetscombinatie **Ctrl+L**. Zie figuur 4.32.



Figuur 4.32: *Starten van de compilatie.*

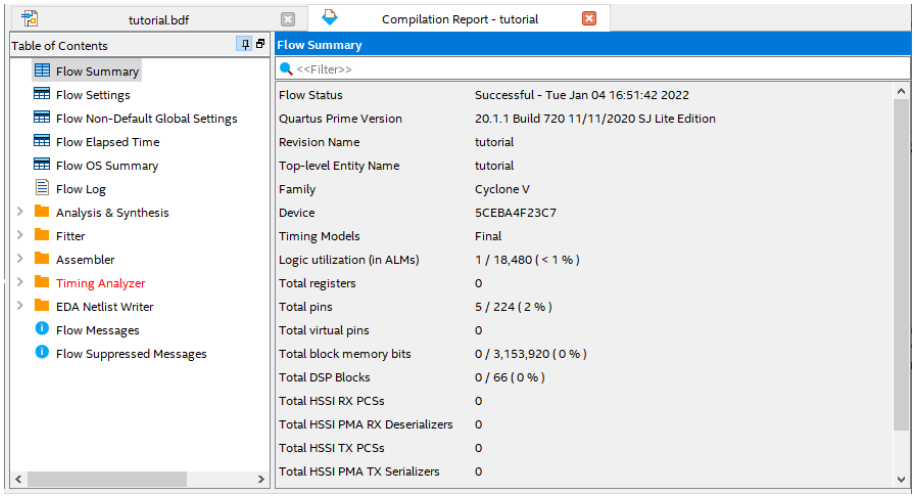
De compilatie wordt nu gestart. Dat kan afhankelijk van het ontwerp enige tijd duren. Je kan de voortgang in het **Tasks**-venster. Als de compilatie geen fouten oplevert krijg je een melding zoals

in figuur 4.33. In het algemeen zijn de waarschuwingen niet problematisch, maar kijk de meldigen toch even na.



Figuur 4.33: *De compilatie is gelukt.*

Je krijgt na compilatie een rapport met alle verrichte werkzaamheden. Een interessant onderdeel hiervan is het aantal gebruikte *Logic Elements*. Hieraan kan je zien hoe groot je ontwerp is. Zie figuur 4.34.



Figuur 4.34: *Overzicht van het resultaat van de compilatie.*

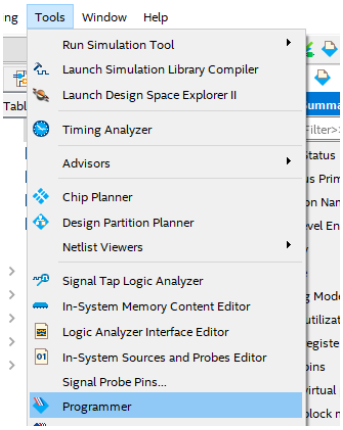
4.10 Configureren van de Cyclone V

De compilatiestap is nu afgerond. We gaan verder met het configureren van de Cyclone V. Dat doen we gaan door het configuratiebestand in deze component te laden. Dat gaat volgens het JTAG-protocol. Dit protocol stelt de gebruiker in staat een hele keten van componenten te configureren. Dit is erg handig omdat je zo updates in de componenten kan laden, ook al zijn ze al op een print gemonteerd.

Zorg ervoor dat de USB-kabel juist is aangesloten en het ontwikkelbord is aangezet. Als je dit vergeet, zal de software een foutmelding geven.

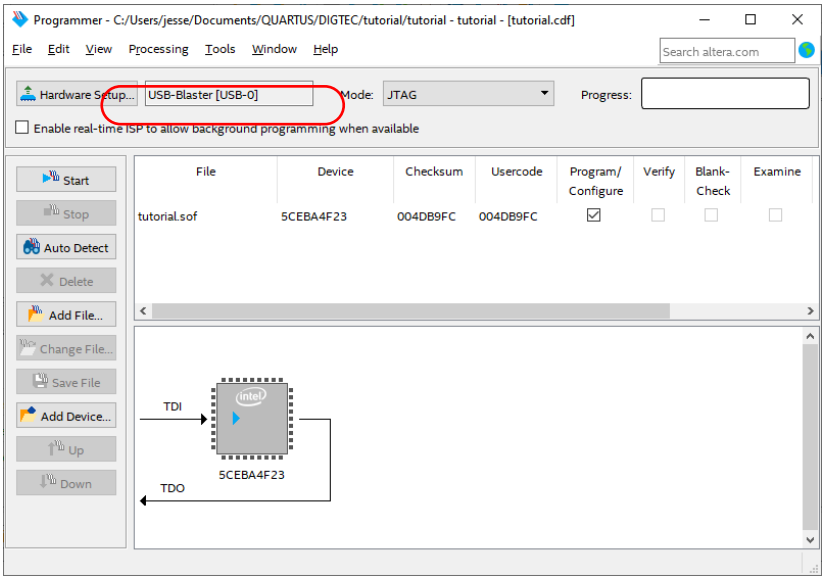
Raadpleeg de docent voordat je de apparatuur aansluit en aanzet!

Selecteer in de Project Manager de menuoptie **Tools**→**Programmer** (figuur 4.35).



Figuur 4.35: *Starten van de programmer.*

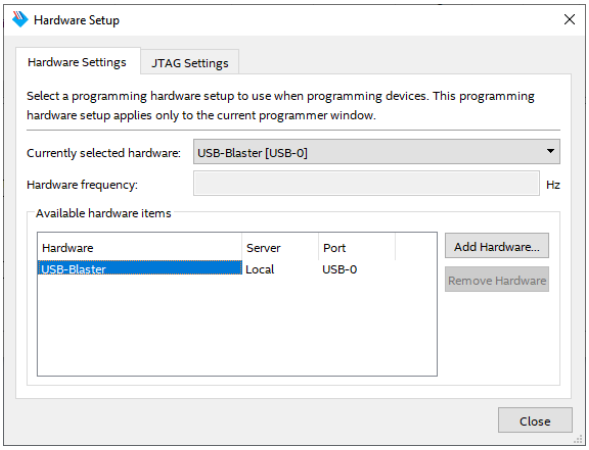
De programmer wordt gestart (figuur 4.36). Je kan zien of de programmer het ontwikkelbord heeft gevonden als in het veld naast Hardware Setup de regel USB-Blaster [USB-0] staat. Zie rode omcirkeling.



Figuur 4.36: *Overzicht van de Programmer IDE.*

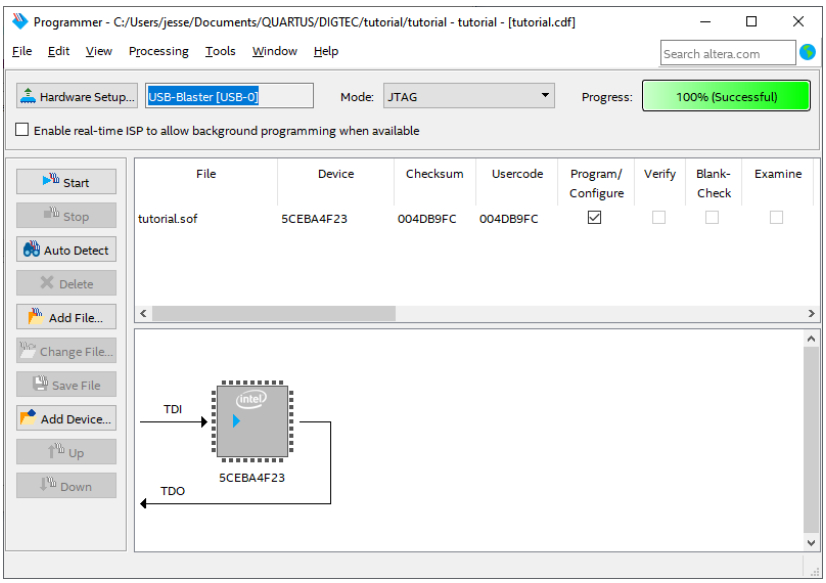
Als in het veld naast Hardware Setup de opmerking No Hardware staat, klik dan op de knop **Hardware Setup**. **Dubbelklik** in het geopende dialoogvenster op USB-Blaster in de lijst bij Available Hardware Items en klik daarna op de knop **Close**. Zie figuur 4.37.

Je ziet ook dat de programmer een bestand tutorial.sof heeft geselecteerd. Dit is het configuratiebestand dat in de Cyclone V geladen moet worden.



Figuur 4.37: *Selecteren van de USB-Blaster download-hardware.*

Druk op **Start** om de Cyclone V te configureren. Daarna kan je het ontwerp testen door middel van de schakelaars en de leds. Als het configureren is gelukt, zie je een venster zoals in figuur 4.38.



Figuur 4.38: *Configureren van de Cyclone V is gelukt.*

Test nu of het DE0-CV-bordje precies reageert zoals de simulatie. Denk eraan: een logische 0 is schuifschakelaar naar beneden, een logische 1 is schuifschakelaar omhoog. Een led gaat branden bij een logische 1 en is gedoofd bij een logische 0.

De tutorial is nu ten einde. Veel succes met het practicum.

5. Tutorial project aanmaken

In deze tutorial gaan we ons bezighouden met het aanmaken en gebruiken van een compleet project. We beginnen met het handmatig starten van Quartus (Quartus Prime Lite). Daarna starten we het aanmaken van een nieuw project. We selecteren vervolgens een geschikte map op de schijf (SSD, HDD), de gebruikte FPGA en de simulator (ModelSim Starter Edition). We voeren een schema in zoals gebruikelijk. We draaien de synthese één keer. Quartus kant daarna de ingangen en uitgangen van de schakeling. Vervolgens koppelen we de ingangen en uitgangen aan pinnen van de FPGA. We draaien dan een volledige compilatie (synthese en implementatie). Als laatste stap laden we het ontwerp in de FPGA.

We zullen de volgende stappen doorlopen:

- Quartus starten;
- opzetten van een nieuw project;
- schema-invoer d.m.v. Schematic Editor;
- Invoeren van een commando-script voor simulatie;
- simuleren van de schakeling op functioneel niveau;
- synthese van de poortschakeling;
- toekennen van ingangen en uitgangen aan pinnen van de FPGA;
- compilatie (synthese en implementatie) van de poortschakeling naar een configuratiebestand;
- downloaden van de configuratiebestand in de Cyclone V.

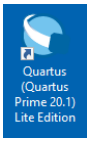
Deze tutorial zal je stap voor stap door de diverse onderdelen van Quartus en ModelSim leiden waarna je zelf een aantal opdrachten moet uitwerken.

NB: gebruik geen spaties, leestekens of ”vreemde”tekens in mapnamen en bestandsnamen! Bestandsnamen altijd beginnen met een letter!

Noot: het doel van deze tutorial is het leren omgaan met de software (zogenaamde tools), niet het leren van digitale techniek. Over de werking van de schakeling die wordt ingevoerd (wat doet het) wordt geen uitleg gegeven.

5.1 Quartus starten

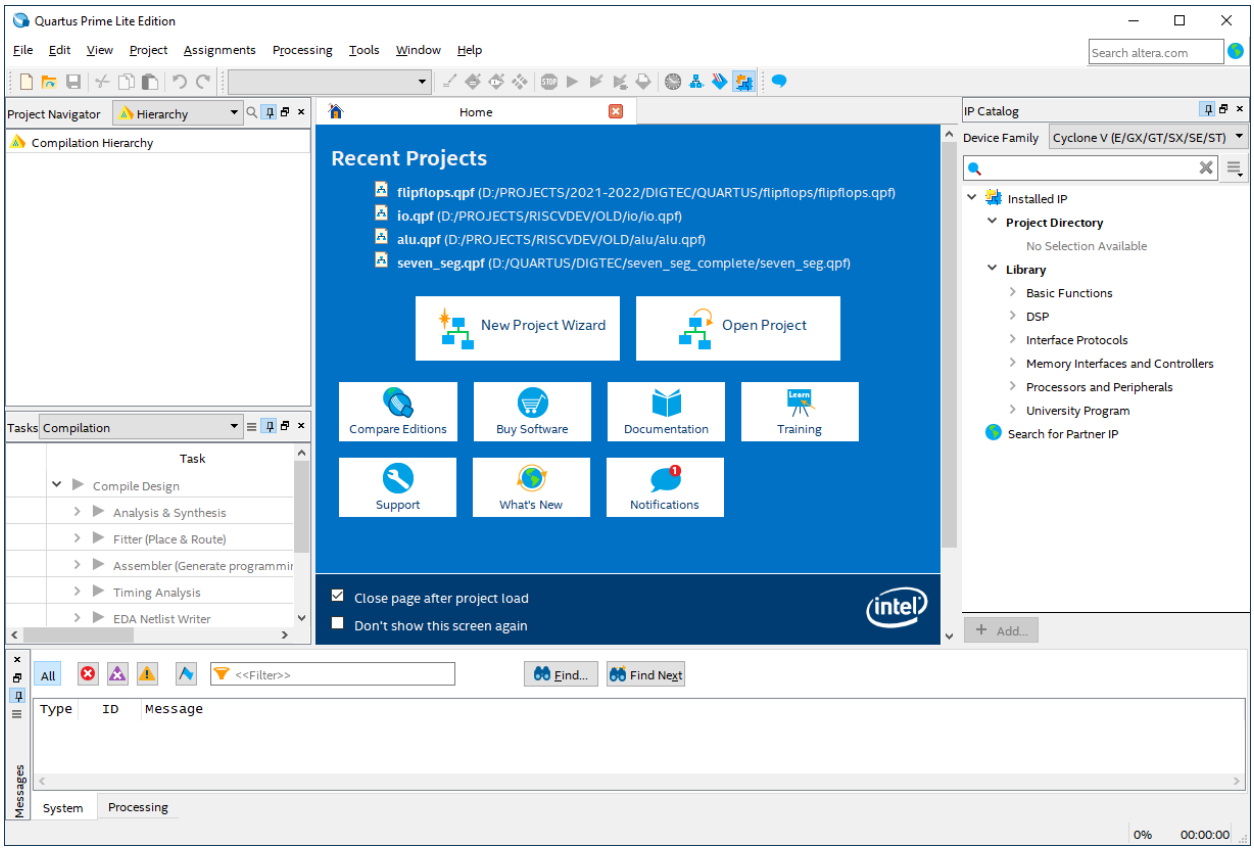
Start Quartus door in de Bureaublad op het Quartus-pictogram te dubbelklikken. Zie figuur 5.1. Je kan Quartus ook starten door in de Windows-zoekbalk het woord Quartus te tikken.



Figuur 5.1: *Het pictogram van Quartus Prime Lite.*

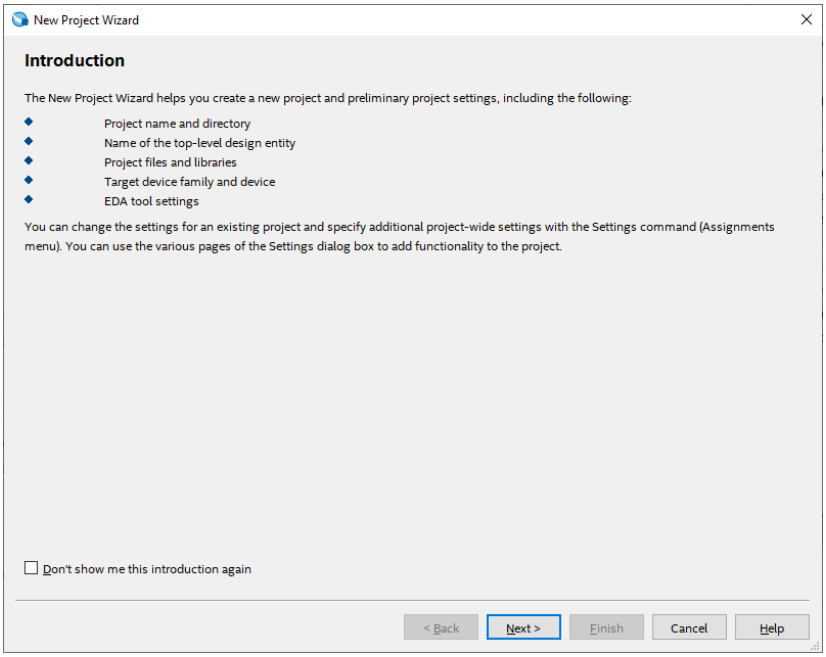
5.2 Project aanmaken

Quartus start nu op met het beginscherm, zie figuur 5.2. In het midden van het beginscherm worden de gebruikte projecten weergegeven (dit kan leeg zijn) en worden diverse opties gepresenteerd. Eén van die opties is New Project Wizard. We gaan deze wizard gebruiken.



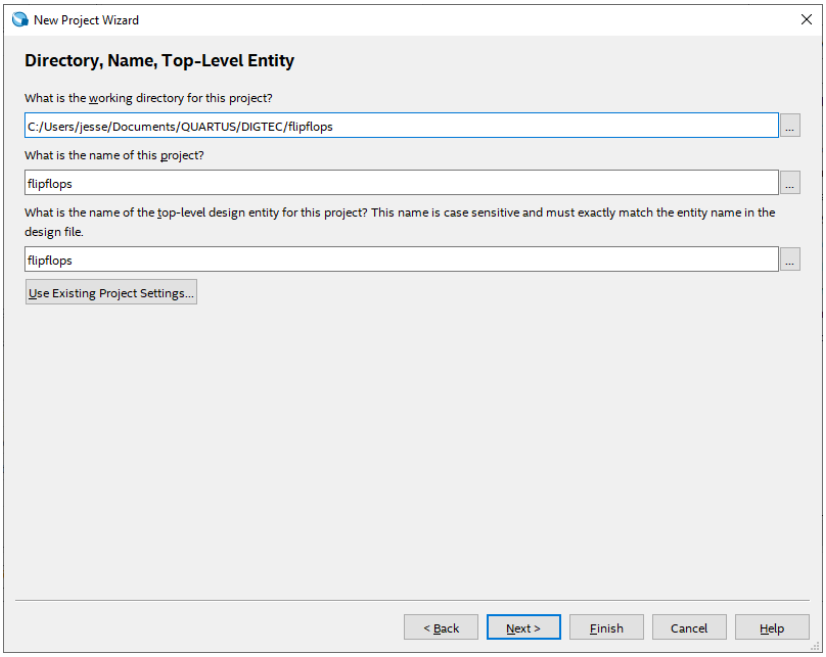
Figuur 5.2: *Het opstartscherm van Quartus.*

Klik in Quartus op de knop **New Project Wizard**. Er wordt een nieuw scherm gestart, zie figuur 5.3. Hierin wordt algemene informatie over de te volgen handelingen. Klik onderaan het scherm op **Next**.



Figuur 5.3: *Het opstartscherm van de wizard.*

Er wordt een nieuw scherm getoond, zie figuur 5.4. De eerste stap is het selecteren van de project-map op de schijf (SSD, HDD). Vul de volledige naam van de map in. Eventueel kan je met de knop met drie puntjes een map selecteren. De mapnaam mag geen spaties of vreemde tekens bevatten. Underscores zijn wel toegestaan



Figuur 5.4: *Het selecteren van de project-map, de naam van het project en de top-level entity.*

In hetzelfde scherm moeten vervolgens de naam en de top-level entity worden opgegeven. Vul hiervoor flipflops in.

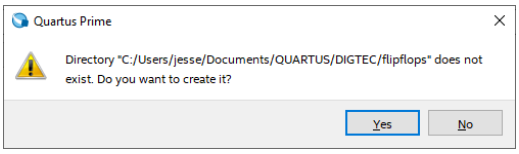
Let op!

Gebruik geen spaties, leestekens of “vreemde” tekens in bestandsnamen en mapnamen!

Gebruik geen minteken in bestandsnamen, underscores zijn wel toegestaan.

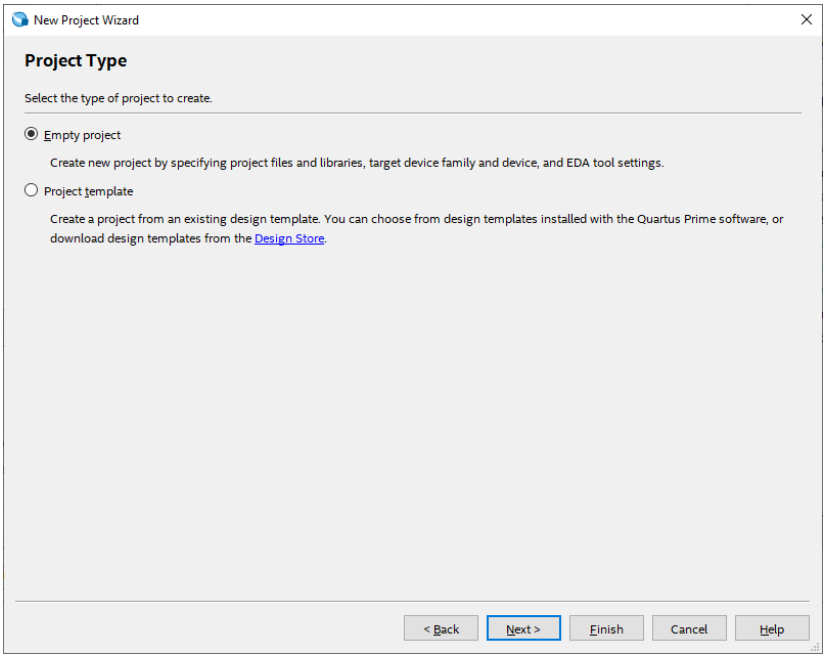
De projectnaam en top-level entity mogen geen spaties bevatten en moeten beginnen met een letter.

Als het goed is, bestaat de map nog niet. Je krijgt dan een melding zoals in figuur 5.5. Klik op **Yes** om de map op de schijf aan te maken.



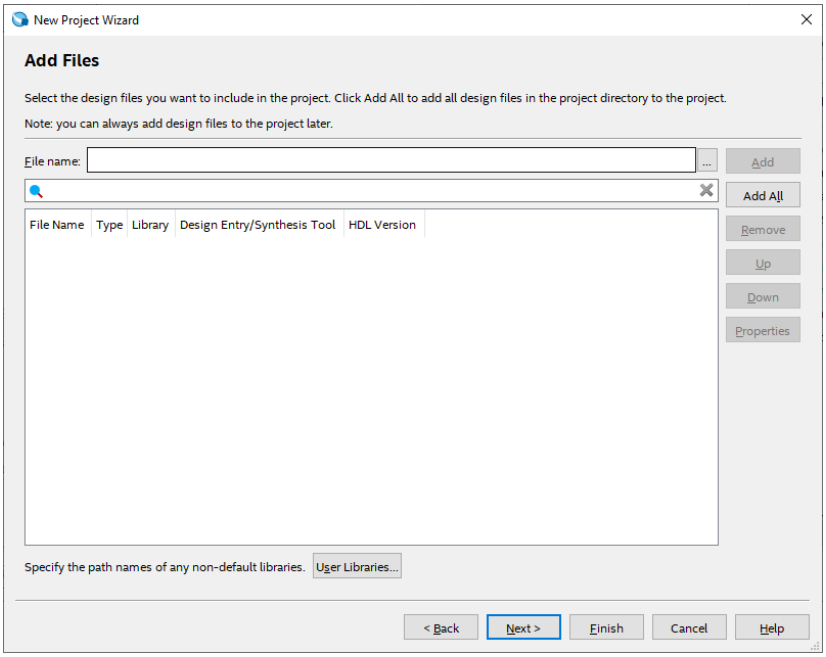
Figuur 5.5: *De map bestaat nog niet.*

In het volgende scherm (figuur 5.6) kiezen we voor een leeg project. Deze opties is standaard geselecteerd. Klik op **Next**.



Figuur 5.6: *Projecttype selecteren.*

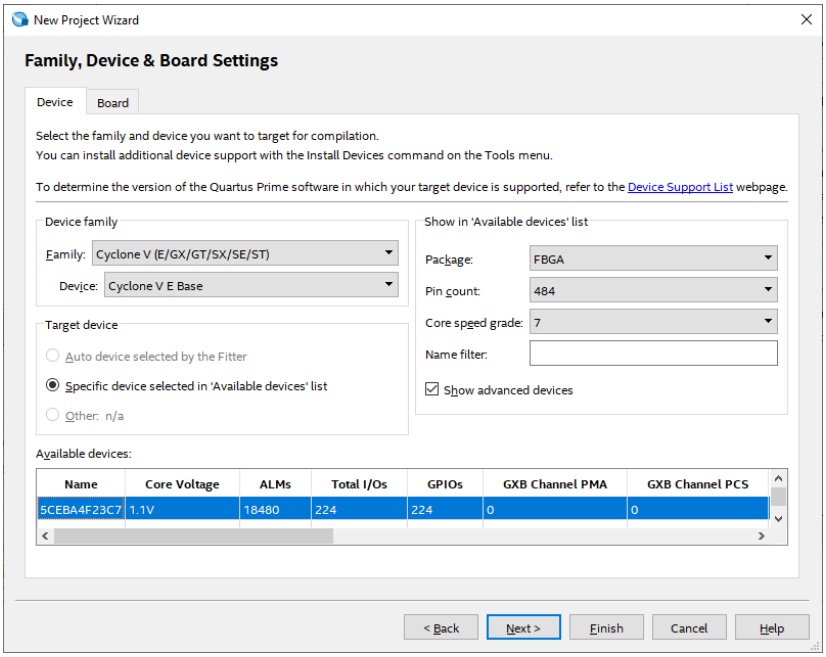
We kunnen nu kiezen om bestaande bestanden toe te voegen, zie figuur 5.7. Voorlopig hebben we nog geen bestanden dus klik op **Next**. We zullen in een later stadium alsnog bestanden toevoegen.



Figuur 5.7: *Bestanden toevoegen.*

Nu moeten we de juiste FPGA voor het project selecteren. De FPGA is het type 5CEBA4F23C7. Dit moet zeer zorgvuldig gebeuren:

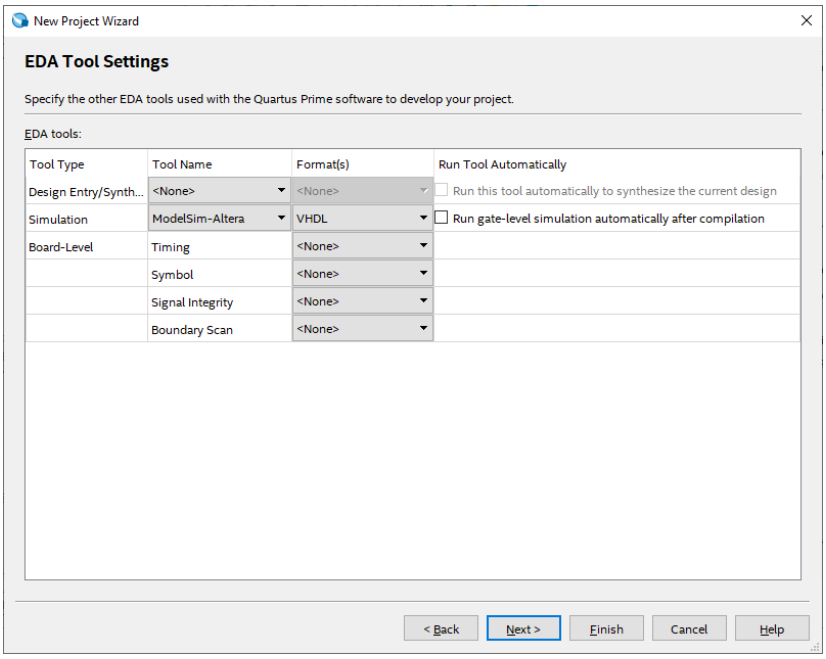
1. Selecteer bij Family de optie Cyclone V (E/GX/GT/SX/SE/ST)
2. Selecteer bij Device de optie Cyclone V E Base
3. Selecteer bij Package de optie FBGA
4. Selecteer bij Pin count de optie 484
5. Selecteer bij Core speed grade de optie 7



Figuur 5.8: *FPGA selecteren.*

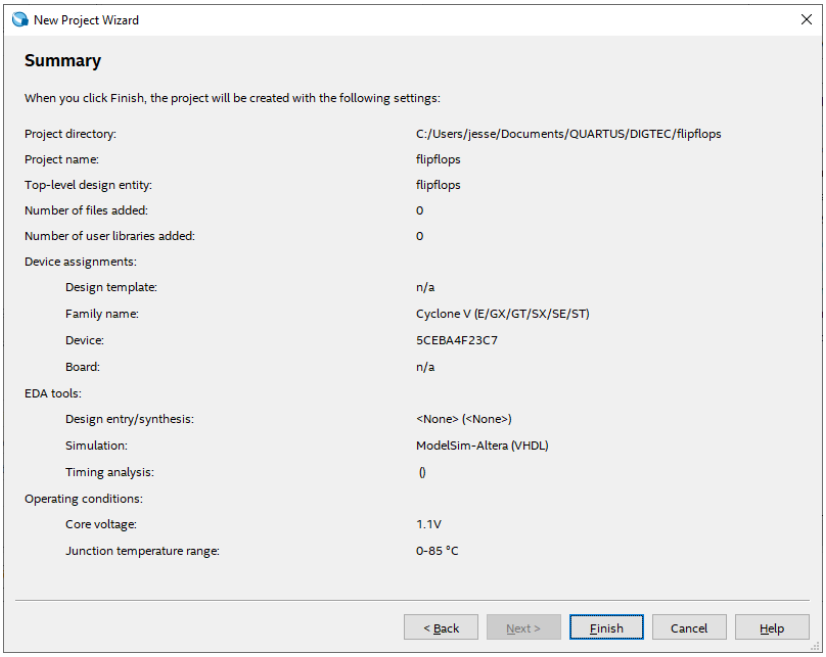
In het veld Available devices onderaan het scherm moet je het type 5CEBA4F23C7 selecteren. Alle gegeven van de gebruikte FPGA zijn nu ingevoerd. Klik op **Next**.

In het volgende scherm (figuur 5.9) moeten we de simulator instellen. Wij gebruiken ModelSim Altera Starter Edition. Kies bij Simulation voor ModelSim-Altera en voor VHDL[3] en klik op **Next**.



Figuur 5.9: *Selecteren van de ModelSim simulator.*

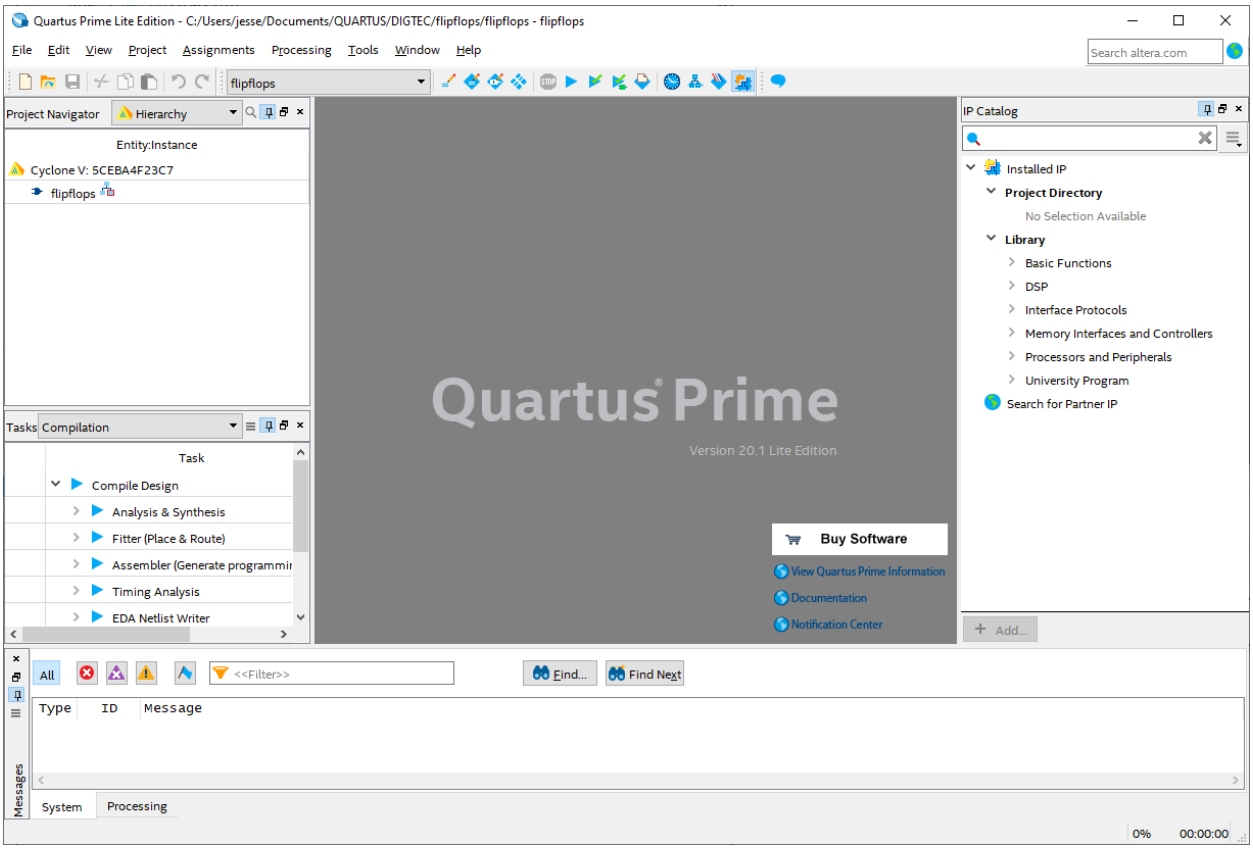
Er wordt nu een opsomming van de instellingen gegeven (figuur 5.10). Klik op **Finish**.



Figuur 5.10: *Opsomming van de wizard.*

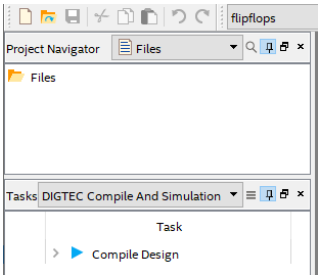
[3] Wij voeren schema's in, de simulator werkt met VHDL. Via de *flow* DIGTEC Compile and Simulation wordt een schema automatisch vertaald zodat de simulator gestart kan worden.

De wizard is nu afgesloten en Quartus laat het beginscherm zien. We hebben een leeg project (figuur 5.11). Links boven is gekozen FPGA en de hiërarchie te zien.



Figuur 5.11: *Het project is aangemaakt.*

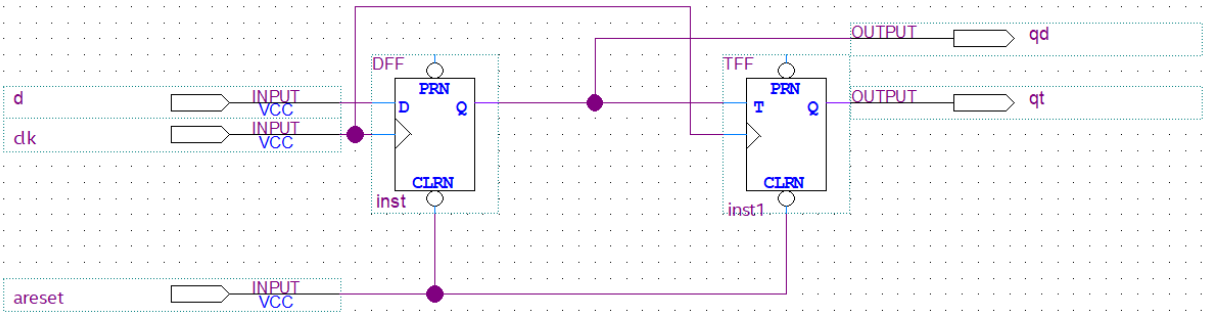
Kies links in de Project Navigator voor het tabblad Files en in Tasks voor het tabblad DIGTEC Compile and Simulation. Zie figuur 5.12.



Figuur 5.12: *Kiezen voor bestanden and flow.*

5.3 Aanmaken schemabestand

We gaan nu een schema invoeren. Dit gebeurt op dezelfde wijze als hoofdstuk 4. Het schema bestaat uit een D-flipflop en een T-flipflop. Het schema heeft drie ingangen, d, clk en areset en twee uitgangen, qd en qt. Het hele schema is te zien in figuur 5.13. Merk op dat ingang PRN niet is aangesloten. Verder is te zien dat de asynchrone reset areset actief laag is.



Figuur 5.13: *Schema met flipflops.*

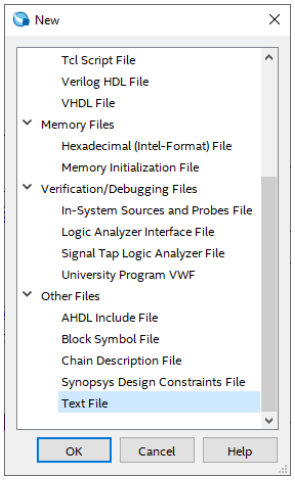
Voer dit schema in en sla het bestand op onder de naam flipflops.bdf.

5.4 Aanmaken simulatie-script

Voordat we de schakeling op het bordje kunnen testen, moeten we het eerst simuleren. Dit droogzwemmen is bedoeld om te kunnen verifiëren of de schakeling, en dus het schema, werkt volgens de specificaties. Quartus Prime Lite gebruikt hiervoor de externe simulator ModelSim van firma Mentor Graphics. We gebruiken de simulator om aan te tonen dat onze schakeling functioneel correct is. Het kan namelijk best zijn dat de schakeling gesynthetiseerd kan worden, maar dat de schakeling niet doet wat het zou moeten doen. Bij deze simulatie worden geen vertragingstijden meegenomen.

Voor simulatie is een zogenaamd scriptbestand nodig. Hierin staan opdrachten voor de simulator. Je kan deze opdrachten ook interactief op een commandoregel invoeren, maar vaak wil je de simulatie een paar keer opnieuw draaien. Dan is het steeds invoeren van de (zelfde reeks) commando's een tijdrovende zaak. In het scriptbestand staat een aantal opdrachten die de simulator gebruikt om de ingangen mee aan te sturen. Dit zijn de zogenoemde *stimuli*. De simulator gebruikt deze waarden om de schakeling door te rekenen.

We beginnen met het aanmaken van een nieuw bestand. Selecteer via het menu **File→New** of gebruik de sneltoetscombinatie **Ctrl+N**. In het geopende venster, kies voor Text File. Zie figuur 5.14.



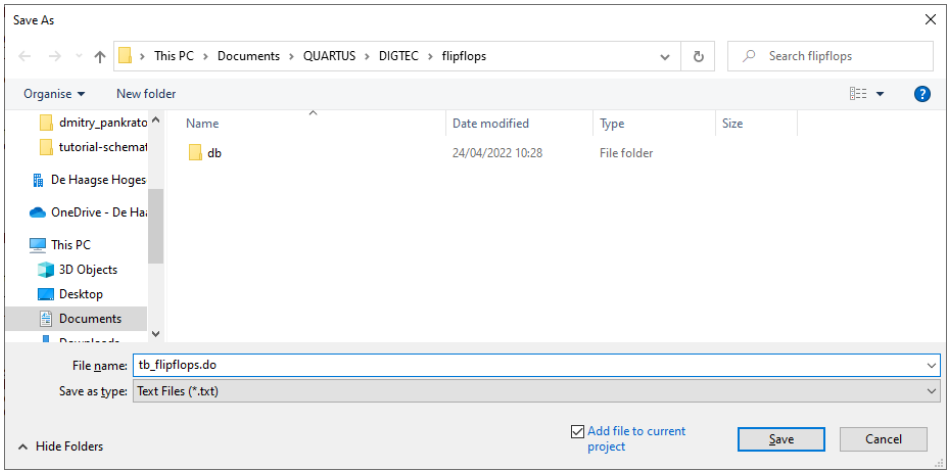
Figuur 5.14: *Aanmaken van een tekstbestand.*

Voer nu de inhoud van het simulatie-script in. Het script is te zien in listing 5.1.

```
# Filename:     tb_flipflops.do
# Filetype:     Modelsim Script File
# Date:         11 jan 2022
# Update:       -
# Description:  Script File For Automatic Simulation
# Author:       <jouw naam>
# State:        Lab
# Error:        -
# Version:      1.0
# Copyright:    (c)2022, De Haagse Hogeschool

# Transcript on, we see output
transcript on

# Recreate working directory
if {[file exists rtl_work]} {
    vdel -lib rtl_work -all
}
vlib rtl_work
vmap work rtl_work

foreach vhd_file [ glob *.vhd ] {
    puts "Compiling: $vhd_file"
    vcom -2008 -work work $vhd_file
}

# Start the simulator
vsim -t 1ns -L rtl_work -L work -voptargs="+acc" flipflops

# Log all signals in the design
add log -r *

# Add all toplevel signals
# Add a number of signals of the simulated design
add wave -label clk clk
add wave -label areset areset
add wave -label d d
add wave -label qd qd
add wave -label qt qt

# Open wave window
view wave
noview Signals
noview Structure
noview list

# Set stimuli
force clk 0 0, 1 10 -repeat 20
force areset 0 0, 1 5
force d 0 0, 1 25, 0 80, 1 135, 0 172, 1 178

# Run simulation for 300 ns
run 300 ns
```

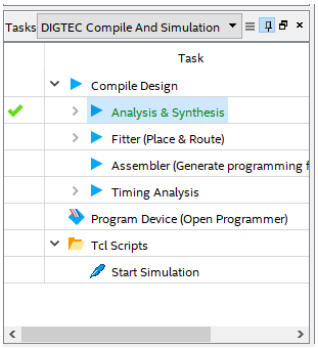
Listing 5.1: *De inhoud van het simulatie-script*

Sla het bestand op onder de naam tb_flipflops.do. **Let op de extensie! Die moet .do zijn!**



Figuur 5.15: *Opslaan van het simulatie-script.*

Het bestand wordt nu zichtbaar in de Project Navigator aan de linkerkant. Voer vervolgens alleen de synthese uit in het Tasks-venster. Zie figuur 5.16.



Figuur 5.16: *Analyse en synthese van het schema.*

We kunnen nu de simulatie starten. Klik in Tasks op Start Simulation. Zie figuur 5.16.

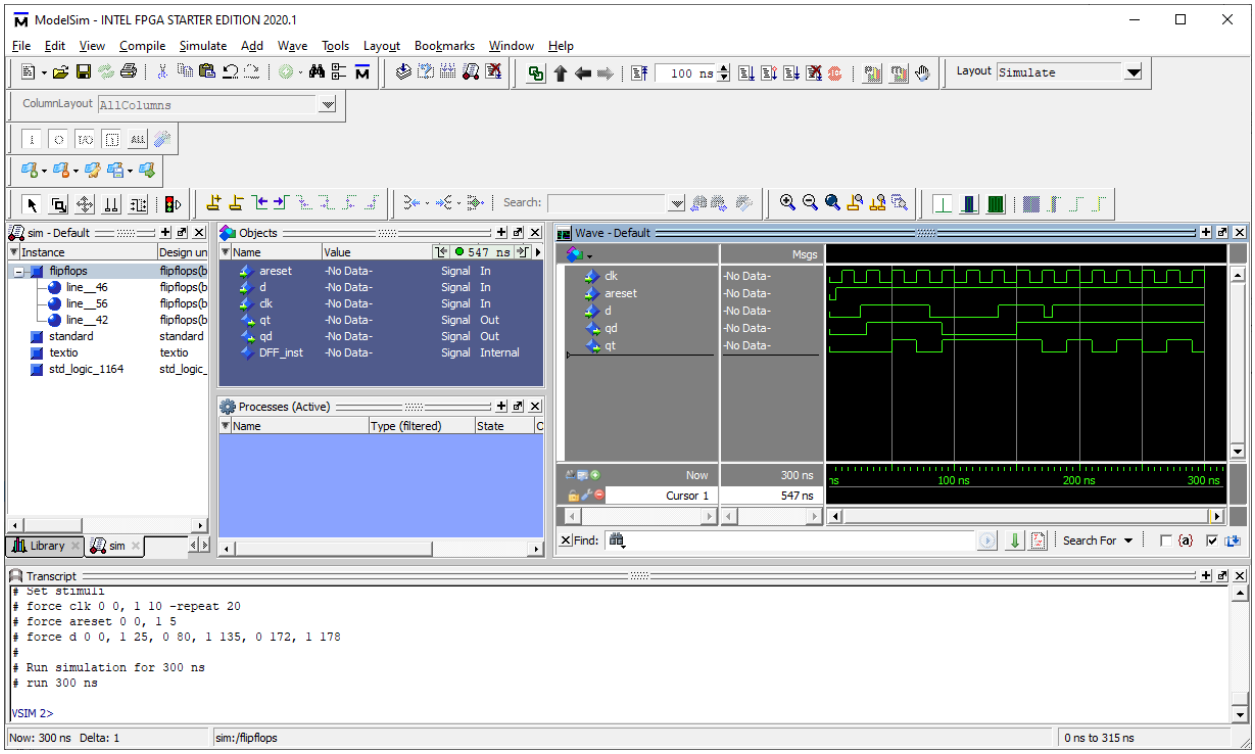
De simulator start nu (dat kan even duren) en draait de commando's uit het simulatie-script. Als alles goed verloopt (denk aan invoerfouten) wordt het scherm in figuur 5.17 zichtbaar.

Als de simulatie voortijdig stopt, is dit meestal te wijden aan een invoerfout. In het transcript-scherm onderaan kan de fout gevonden worden. In het bestand tb_flipflops.do moet de fout hersteld worden. Vergeet daarna niet om het bestand op te slaan.

Om de stimuli aan te passen moet je de force-commando's aanpassen. De algemene gedaante van het force-commando is:

```
force <signaal> waarde tijd, waarde tijd, ...
```

waarbij waarde 0 of 1 is en tijd de absolute tijd in simulatie-eenheden is (in dit geval 1 ns).

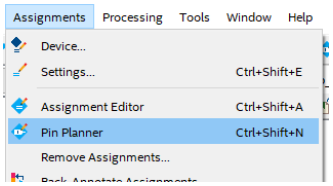


Figuur 5.17: *De simulator na het draaien van het script.*

5.5 Koppelen van de ingangen en uitgangen aan pinnen

Als we het schema op het bordje willen testen, moeten we eerst de ingangen en uitgangen koppelen aan pinnen van de FPGA. De schakelaars, drukknoppen en leds zijn per stuk gekoppeld aan exact één pin. Dit wordt gedaan met de Pin Planner.

Start de Pin Planner via het menu **Assignments→Pin Planner** of gebruik sneltoetscombinatie **Ctrl+Shift+N**. Zie figuur 5.18.

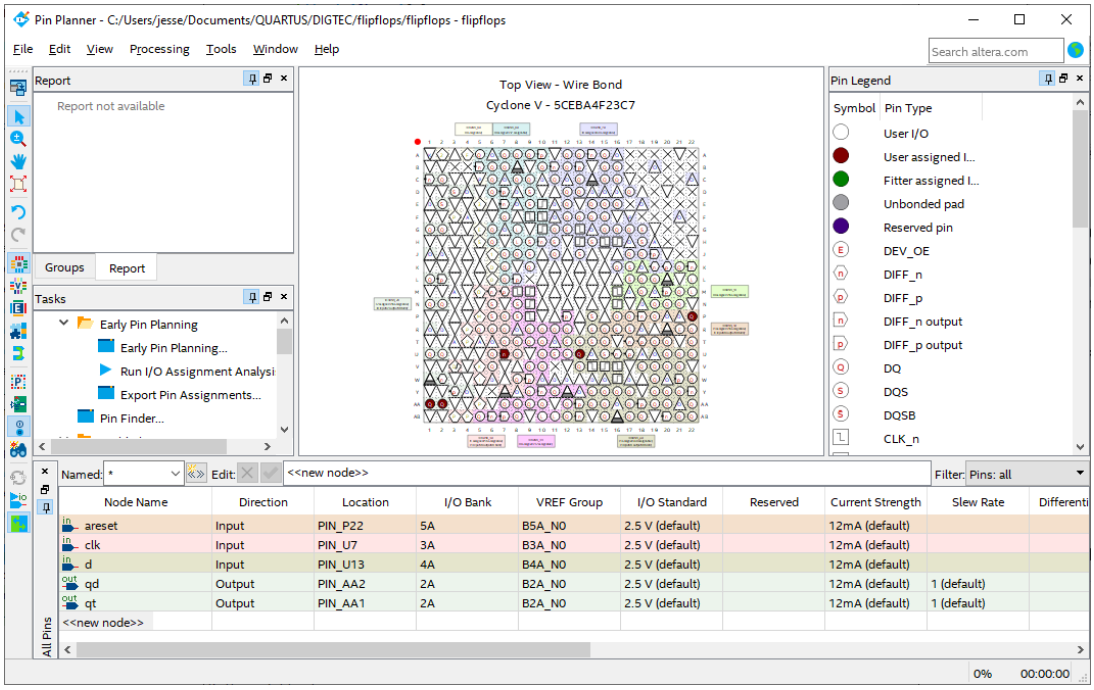


Figuur 5.18: *Starten van de Pin Planner.*

De Pin Planner wordt gestart in een nieuw venster. We voeren de pinnamen in zoals te zien is in tabel 5.1. Zie ook figuur 5.19.

Tabel 5.1: *Koppelingen ingangen aan pinnen en schakelaars*

Ingang	Pinnaam	Bordje
clk	U7	KEY0
areset	P22	FPGA_RESET
d	U13	SW0
qd	AA2	LEDR0
qt	AA1	LEDR1



Figuur 5.19: *Pin Planner ingevuld.*

Sluit de Pin Planner af met menu **File**→**Close** (er is geen Save-mogelijkheid).

In bijlage A is een tabel te vinden met veel gebruikte pinnen.

5.6 Compilatie en configuratie

De compilatie en configuratie gaat exact hetzelfde zoals in hoofdstuk 4 beschreven is. Zie aldaar. Als het ontwerp geladen is in het bordje, kunnen we het testen. Let erop dat ingangen clk en areset verbonden zijn met drukknoppen die actief laag zijn: indrukken van een drukknop geeft een logische 0 aan de ingang, niet indrukken geeft een logische 1. Ingang d is gekoppeld aan een schuifschakelaar. De uitgangen qd en qt zijn verbonden met ledjes.

5.7 Extra opdracht

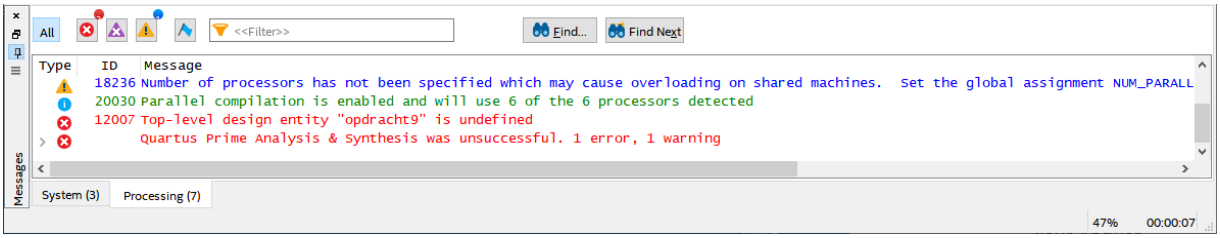
Als extra opdracht kan je het aantal flipflops in het schema uitbreiden. Er zijn meer typen flip-flops te vinden: D- en T-flipflops met enable, J-K- en S-R-flipflops en een D-latch. Zorg ervoor dat het simulatie-script met de nieuwe signalen wordt uitgebreid en ken nieuwe pinnen toe aan de nieuwe signalen.

6. Tips, tricks & troubleshoot

In dit hoofdstuk wordt een aantal veel voorkomende problemen toegelicht en hoe je ze kunt verhelpen. Daarnaast natuurlijk de tips & tricks.

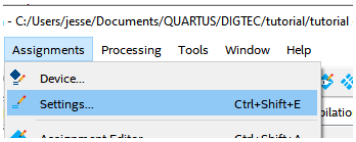
6.1 Foutmelding Top-level undefined

Onderstaande foutmelding geeft aan (zie figuur 6.1) dat de ingestelde *top-level design entity* niet gedefinieerd is. De kwalificatie *top-level* slaat op de allerhoogste design entity (denk aan VHDL-entity) en die kan niet gevonden worden of is niet gedefinieerd.



Figuur 6.1: *De top-level entity is niet gevonden.*

Dit gebeurt meestal als er geen BDF-bestand is met de naam van het project (bijvoorbeeld tutorial.bdf). Gelukkig kan in Quartus een andere entity als top-level worden ingesteld. Selecteer via het menu **Assignments**→**Settings** of gebruik de sneltoets **Ctrl+Shift+E**. Zie figuur 6.2.



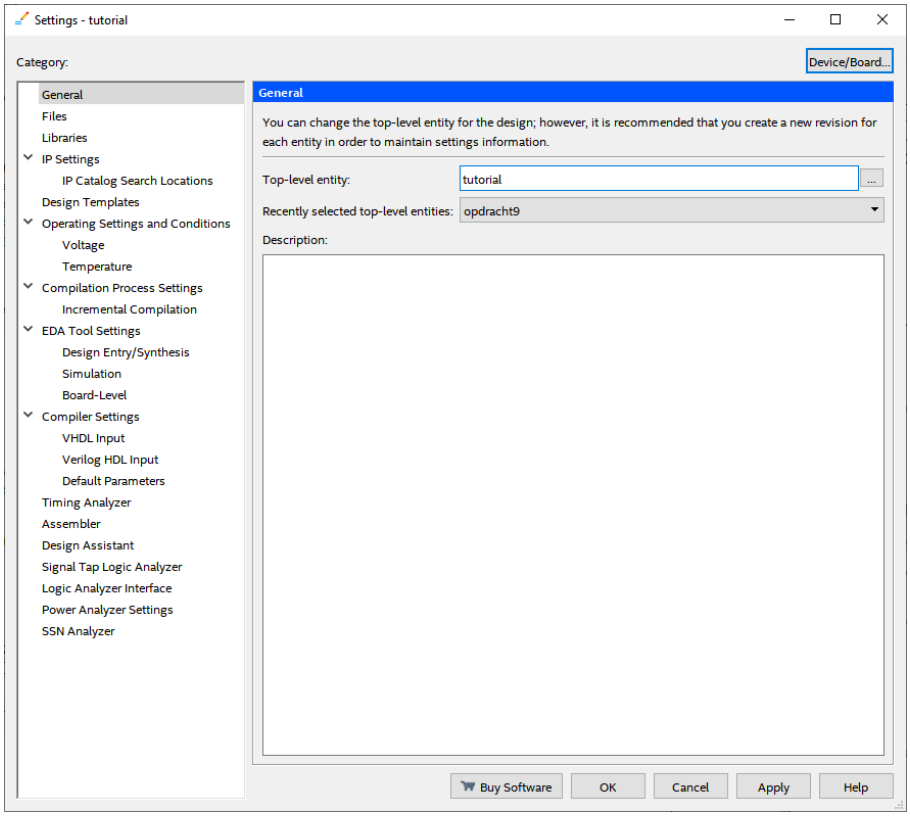
Figuur 6.2: *Openen van de Settings.*

Klik in het nieuw geopende dialoog links boven op General.

Rechts kan een ander top-level gekozen worden door op de knop met de drie puntjes te klikken achter het veld met Top level entity: waarna in een klein venster de nieuwe naam gekozen kan worden. Klik dan op **Ok**. Het kleine venster wordt afgesloten. Klik in de dialoog eerst op knop **Apply** en daarna op knop **Ok**. Zie figuur 6.3. De dialoog wordt afgesloten.

Het kan ook zijn dat er geen top-level zichtbaar is. Vul dan op de regel achter Top-level entity de juiste naam in en sluit af.

In de Program Navigator is nu de nieuwe top-level entity te vinden.



Figuur 6.3: *Selectie van top-level entity.*

6.2 Bestandsnaam en entity-naam

Een *entity* is een hardware-eenheid en levert bij synthese dus hardware op. Een *bestandsnaam* is de naam van het bestand waarin de hardware beschreven of getekend is. De entity-naam is onafhankelijk van het gebruikte besturingssysteem, de bestandsnaam is wel afhankelijk.

In Quartus zijn schemabestanden met de extensie .bdf gekoppeld aan de entity-naam: de bestandsnaam zonder de extensie is gelijk ook de entity-naam.

Bij beschrijvingstalen als VHDL en Verilog ligt dat anders: de bestandsnaam hoeft niet hetzelfde te zijn als de entity-naam. In feite is het bestand een *container* met daarin de beschrijving van de hardware. ModelSim gebruikt de bestandsnaam om de beschrijving te compileren (bv. met vcom en vlog) maar gebruikt de entity-naam bij het starten van de simulatie (m.b.v. vsim).

Het is raadzaam om de bestandnaam (zonder extensie) hetzelfde te houden als de entity-naam. Daarmee voorkom je problemen.

Let op: entity-namen mogen *niet* beginnen met een cijfer of een leesteken. In feite gelden voor de entity-namen dezelfde regels als voor variabelen. Schemabestandsnamen mogen dus ook niet met een cijfer of een leesteken beginnen, VHDL-bestandsnamen wel.

6.3 Opruimen van een Quartus-project

Quartus heeft de neiging om tijdens compilatie ontzettend veel, vooral kleine bestanden aan te maken. Je kan heel veel van die bestanden en mappen gewoon verwijderen als je project is afgerond.

De mappen `db`, `incremental_db`, `output_files` en `simulation` kan je gewoon verwijderen. Let erop dat je geen eigen aangemaakte bestanden in de map `simulation` moet hebben staan.

Onderstaand script kan je draaien in een Windows-command box en verwijdert bijna elk bestand dat niet nodig is.

```
@echo off
rem
rem
echo.
echo This program will delete all unnessecary files from Quartus
   projects.
echo.
echo Please be VERY carefull! Press Ctrl-C to break.
echo.
pause
echo.

echo Removing all db directories...
FOR /D /r %%G in ("db*") DO rd /s/q %%G
echo Removing all incremental_db directories...
FOR /D /r %%G in ("incremental_db*") DO rd /s/q %%G
echo Removing all simulations directories...
FOR /D /r %%G in ("simulation*") DO rd /s/q %%G
echo Removing all output_files directories...
FOR /D /r %%G in ("output_files*") DO rd /s/q %%G

del /s *.done
del /s *.rpt
del /s *.sof
del /s *.pof
del /s *.summary
del /s *.jdi
del /s vsim.wlf
del /s *.bak
del /s transcript
del /s *assignment_defaults.qdf
del /s *.pin
del /s modelsim.ini
del /s *.qws
del /s *.smsg
del /s *.map
del /s *.cdf
del /s *.dpf
del /s PLLJ_PLLSPE_INFO.txt

echo.
echo Done.
echo.
pause
```

Listing 6.1: *Windows opruimscript*

A. Pinbenaming 5CEBA4F23C7N

In deze bijlage vind je de pinbenaming terug. Er staan ook opmerkingen bij.

Tabel A.1: *Pinbenamingen FPGA, deel 1.*

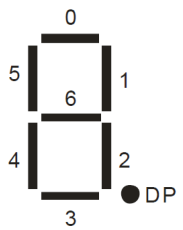
Type	Quartus-naam	Pinnaam	Opmerking
Clock	CLOCK_50	M9	Global Clock 1
Push Buttons	KEY[0]	U7	Ontdenderd, actief laag
	KEY[1]	W9	
	KEY[2]	M7	
	KEY[3]	M6	
	FPGA_RESET	P22	ook wel KEY[4]
Switches	SW[0]	U13	Niet ontdenderd, actief hoog
	SW[1]	V13	
	SW[2]	T13	
	SW[3]	T12	
	SW[4]	AA15	
	SW[5]	AB15	
	SW[6]	AA14	
	SW[7]	AA13	
	SW[8]	AB13	
	SW[9]	AB12	
Leds	LEDR[0]	AA2	Actief hoog
	LEDR[1]	AA1	
	LEDR[2]	W2	
	LEDR[3]	Y3	
	LEDR[4]	N2	
	LEDR[5]	N1	
	LEDR[6]	U2	
	LEDR[7]	U1	
	LEDR[8]	L2	
	LEDR[9]	L1	

Dit zijn de benamingen zoals ze in de documentatie van het DE0-bordje worden gebruikt. Je kan ook je eigen namen gebruiken. Quartus zal, wanneer van toepassing, het woord PIN_ voor de pinnaam zetten, dus J2 wordt dan PIN_J2.

Op de volgende pagina staan de pingegevens van de 7-segment displays. Tevens is de layout gegeven.

Tabel A.2: *Pinbenamingen FPGA, deel 2.*

Type	Quartus-naam	Pinnaam	Opmerking
7-segment	HEX00	U21	Alle actief laag
	HEX01	V21	
	HEX02	W22	
	HEX03	W21	
	HEX04	Y22	
	HEX05	Y21	
	HEX06	AA22	
	HEX10	AA20	
	HEX11	AB20	
	HEX12	AA19	
	HEX13	AA18	
	HEX14	AB18	
	HEX15	AA17	
	HEX16	U22	
	HEX20	Y19	
	HEX21	AB17	
	HEX22	AA10	
	HEX23	Y14	
	HEX24	V14	
	HEX25	AB22	
	HEX26	AB21	
	HEX30	Y16	
	HEX31	W16	
	HEX32	Y17	
	HEX33	V16	
	HEX34	U17	
	HEX35	V18	
	HEX36	V19	



Figuur A.1: *Layout 7-segment displays*

Noot: DP niet aangesloten.

B. DIGTEC-FLOW HANDMATIG INSTALLEREN

Tijdens het practicum wordt gebruik gemaakt van een eigen *flow*. In een flow staan de handelingen die door Quartus gedaan moeten worden om tot het gewenste resultaat te komen. Deze flow is nodig voor het uitvoeren van de practicumopdrachten.

Er is echter een probleem: ModelSim kan alleen maar VHDL-code (en Verilog) simuleren. Er is een script gemaakt (`start_sim.tcl`) dat alle .bdf-bestanden vertaalt naar VHDL-code en vervolgens de simulator start. Het script is zo geschreven dat het ook op Linux draait.

Om de flow op Windows te laten draaien moet je de volgende handelingen verrichten:

- Installeer de Quartus-software op een standaard plaats, bijvoorbeeld `c:\intelFPGA\`.
- Installeer de Modelsim-software[4] onder de Intel-root (met versienummer), meestal iets van `c:\intelFPGA\20.1\`
- Download het bestand `digtec_common_tutorial.zip` van BlackBoard of de website https://ds.opdenbrouw.nl/quartus/.
- Pak het bestand uit in een map, bijvoorbeeld `C:\Documents\QUARTUS\DIGTEC\`. Je krijgt dan twee mappen genaamd `common` en `tutorial`
- Navigeer naar de map `common` en dubbelklik op het bestand `install_flow.vbs`. Volg de aanwijzingen op het scherm. Zie ook paragraaf 4.1.

Mocht het niet lukken om de flow automatisch te installeren, dan kan je de flow ook handmatig installeren. De flow is opgeslagen onder de naam `tmwc_DIGTEC_Compile_And_Simulation.tmf` in de map `common` en moet geïnstalleerd worden in de *profile map* van de gebruiker, meestal iets van
`C:\Users\<gebruikersnaam>\` (natuurlijk zonder $<$ en $>$).

Om nu de flow onder Windows te gebruiken moet je de padnaam in het bovengenoemde bestand wijzigen in de juiste locatie. **Let op: geen spaties in de padnaam!**

```
... beginstuk weggelaten ...

    <task>
        <id>Start Simulation</id>
        <name>Start Simulation</name>
        <item_bitmap>tcl_command</item_bitmap>
        <status_ok_if>project_is_open</status_ok_if>
        <action type =
            "tcl_command">D:/QUARTUS/INLDIG/common/start_sim.tcl</
               action>
    </task>
</tasks>
```

[4] Vanaf versie 13.0 wordt ModelSim automatisch mee-geïnstalleerd.