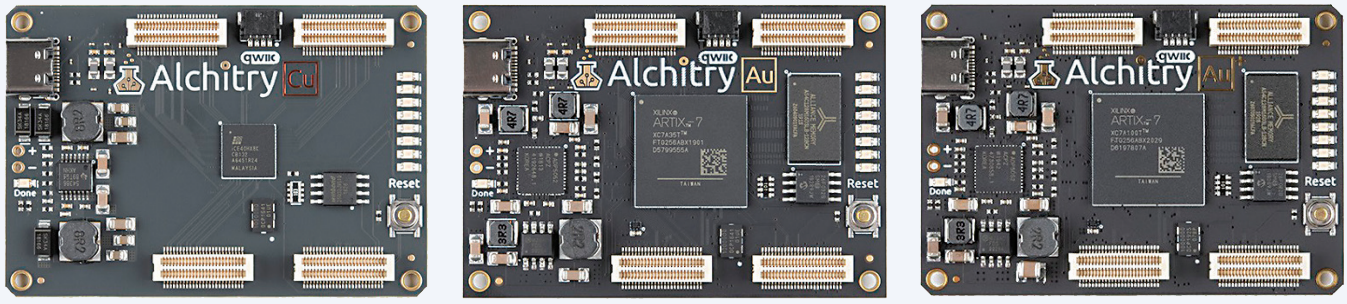
In een **open-source processor**

voorbeeldhoofdstuk: vergelijking van Lattice- en Xilinx-FPGA's



De Alchitry FPGA-ontwikkelboards die in dit artikel worden besproken Van links naar rechts: Cu – Au – Au+.

Monte Dalrymple (VS)

In deze aflevering van Elektor Books presenteren we een hoofdstuk uit het boek Inside an Open-Source Processor van Monte Dalrymple, uitgegeven door Elektor. FPGA-technologie in combinatie met (V)HDL is in trek bij elektronici, en met RISC-V is nu ook de open source-weg naar professionele toepassingen geopend. In deze bijdrage vergelijken we de resultaten van een voorbeeldapplicatie die draait op de Alchitry FPGA-boards van Lattice en Xilinx die bij Elektor verkrijgbaar zijn. Maar eerst een paar tips!

Opmerking van de redactie: *dit artikel is een deel uit het boek Inside an Open-Source Processor – an Introduction to RISC-V, dat we enigszins hebben bewerkt om te voldoen aan de redactionele standaarden en paginaopmaak van Elektor Magazine. Aangezien het een gedeelte is van een grotere publicatie, verwijzen sommige aanduidingen in dit artikel naar passages elders in het origineel. Auteur en uitgever hebben hun best gedaan om dat te voorkomen, en zijn graag bereid om eventuele vragen te beantwoorden. Contactgegevens staan in het kader* ***Vragen of opmerkingen?****.*

In het verleden was het opzetten van een project in een FPGA-toolreeks een lastige klus, maar dankzij de ontwikkeling van de technologie is dit inmiddels veel eenvoudiger geworden. In het geval van het voorbeeld in dit hoofdstuk (een 24-uursklok die elders in het boek wordt beschreven) is het meeste werk al gedaan.
Bedenk wel dat grote projecten, of projecten waarbij maximaal gebruik wordt gemaakt van de mogelijkheden van de FPGA, onvermijdelijk veel complexer zullen zijn.Bij zowel Lattice als Xilinx hoeven, afgezien van het kiezen van het juiste FPGA-doel voor het project, slechts twee of drie bestanden in de tools te worden gekoppeld. Bij deze vereenvoudigde aanpak moeten alle projectbestanden in dezelfde directory staan. Zoals eerder vermeld, wordt de lezer sterk aangeraden de Alchrity-tutorials voor de installatie van de FPGA-software en het opzetten van een project door te

nemen, omdat in dit artikel alleen de belangrjkste details worden behandeld.

Tips voor Lattice (Alchitry Cu)

Voor de FPGA die op deze print is gemonteerd, is de Lattice iCEcube2-software nodig. Dit is een uitgerijpt product zonder veel toeters en bellen, maar u moet rekening houden met het volgende:

1. Zorg ervoor dat het project is ingesteld met de juiste apparaatopties. Voor de Alchitry Cu wordt gebruikgemaakt van de HX8K uit de iCE40-familie, in CB132-behuizing. Alle I/O-banken gebruiken een voeding van 3,3 V.
2. Alleen het bestand *yrv_alchitry.v* moet als ontwerpbestand worden ingevoerd, omdat alle andere ontwerpbestanden automatisch in de juiste volgorde worden opgenomen.
3. Kies voor Lattice LSE Synthesis als synthesetool. Tijdens de logicasynthese zijn geen constraint-bestanden nodig.
4. Nadat de synthese is voltooid, moeten het pin-constraint bestand *yrv_alchitry.pcf* en het timing-constraint bestand *timing.sdc* worden gekoppeld voordat de rest van de tools wordt uitgevoerd. In de verschillende logbestanden staan enkele waarschuwingen, maar geen enkele daarvan zou belangrijk genoeg mogen zijn om actie te ondernemen.

Tips voor Xilinx (Alchitry Au en Au+)

Voor de FPGA's die op deze printen worden gebruikt is Xilinx Vivado-software nodig. Dit is een zeer complex product, en voor dit project wordt slechts een kleine subset van de mogelijkheden hiervan gebruikt. Een paar dingen om rekening mee te houden:

1. Alleen het bestand *yrv_alchitry.v* moet als ontwerpbestand worden ingevoerd.
2. Alleen het constraint-bestand *yrv_alchitry.xdc* is vereist. Dit bestand bevat de pin-constraints, timing-constraints en de specificaties van de programmeerspanning.
3. De juiste apparaatoptie is xc7a35tftg256-1 voor het Au-board en xc7a100tftg256-1 voor het Au+-board.

In de logbestanden voor deze chips staan nog meer waarschuwingen, maar nogmaals: geen daarvan zou belangrijk genoeg mogen zijn om actie te ondernemen.

FPGA-resultaten

Dit voorbeeldproject is uitgevoerd op alle drie de doel-FPGA-boards. De verschillende resultaten worden in de volgende paragrafen behandeld. Op dit punt is het echter interessant om een aantal resultaten naast elkaar te zetten. **Tabel 1** toont de gerapporteerde kloksnelheid voor de 100MHz-deler.

De 100MHz-deler is ontworpen voor maximale snelheid, met slechts één logicaniveau tussen de flip-flops. Dit betekent dat dit resultaat de mogelijkheden van de technologie zou moeten representeren.

Tabel 1. FPGA-resultaten.

100MHz-deler	Alchrity Cu	Alchrity Au	Alchrity Au+
Snelheid (bij synthese)	313,9 MHz	114,1 MHz	114,2 MHz
Snelheid (bij plaatsing)	542,3 MHz	-	-
Snelheid (definitief)	625,4 MHz	111,9 MHz	111,9 MHz

Tabel 2. FPGA-resultaten.

FPGA-attribuut	Alchrity Cu	Alchrity Au	Alchrity Au+
Logica-elementen	7.680	20.800	63.400
Gebruikte LE's	4.646	2.189	2.185
Gebruikte LE's (%)	60,6%	10,5%	3,5 %
Flip-flops	1331	1377	1377
Snelheid (bij synthese)	11,8 MHz	28,1 MHz	27,7 MHz
Snelheid (bij plaatsing)	25,8 MHz	-	-
Snelheid (definitief)	30,8 MHz	34,4 MHz	34,2 MHz

De gerapporteerde snelheid voor de Lattice-chip (als die klopt) is indrukwekkend. Het interessantste is dat de gerapporteerde snelheid beter en vermoedelijk nauwkeuriger wordt naarmate de implementatie vordert van logische synthese via plaatsing van logica tot het uiteindelijke gerouteerde resultaat. Hoewel het prima is als de uiteindelijke prestatie beter is dan de oorspronkelijke raming, is een afwijking van een factor twee niet ideaal, omdat dit ertoe kan leiden dat een project ten onrechte wordt afgeblazen omdat het onhaalbaar lijkt te zijn.

De Xilinx-tools geven geen aparte snelheidsschatting bij de plaatsing van logica, maar de gerapporteerde snelheid is bij logicasynthese en bij het eindresultaat vrijwel identiek. Vanuit het oogpunt van een ontwerper verdient dit de voorkeur.

Tabel 2 geeft een overzicht van de voor het project benodigde resources en de maximale klokfrequentie voor de CPU zelf. Deze resultaten omvatten geen van de 64bit-tellers die deel uitmaken van de *RISC-V Privileged Architecture*.

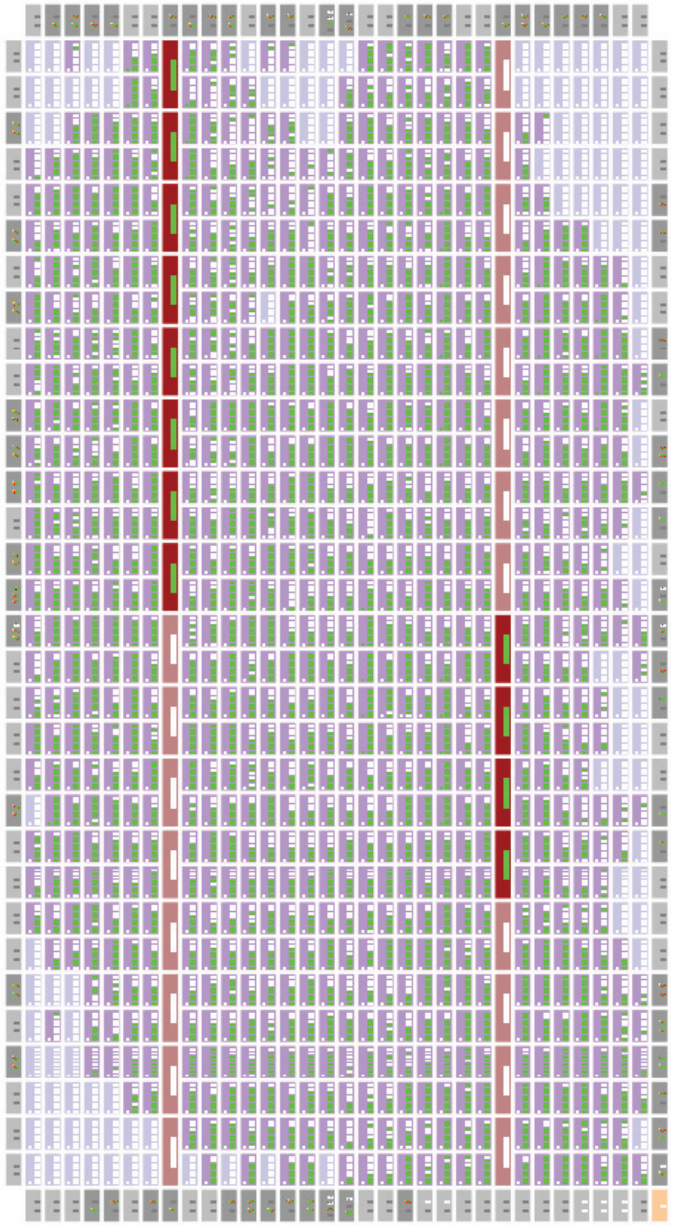
Listing 1. Gedeelte van het Lattice-rapport.

```
Total Logic Cells: 4646/7680
       Combinational Logic Cells: 3315    out of   7680    43.1641%
       Sequential Logic Cells:    1331    out of   7680    17.3307%
       Logic Tiles:               847     out of   960     88.2292%
   Registers:
       Logic Registers:           1331    out of   7680    17.3307%
       IO Registers:              0       out of   1280    0
   Block RAMs:                    12      out of   32      37.5%
   Warm Boots:                    0       out of   1       0%
   Pins:
       Input Pins:                36      out of   95      37.8947%
       Output Pins:               50      out of   95      52.6316%
       InOut Pins:                0       out of   95      0%
   Global Buffers:                2       out of   8       25%
   PLLs:                          0       out of   2       0%
```

De tweede rij van deze tabel laat zien waarom het vergelijken van FPGA-resultaten lastig kan zijn. Uit de tabel blijkt dat de Xilinx-implementatie minder dan de helft van de resources verbruikt dan die welke vereist zijn bij de Lattice-implementatie. Maar een Xilinx logica-element bevat aanzienlijk meer logica dan een Lattice logica-element. Bovendien maakt het Xilinx-ontwerp gebruik van de speciale DSP-blokken die de Lattice-chip niet heeft.
Als we kijken naar het percentage gebruikte logica-elementen zien we het verschil in omvang tussen de drie FPGA's. Iets meer dan de helft van de Lattice-chip is nodig voor het *yrv_mcu*-ontwerp, maar slechts ruwweg een tiende van de Xilinx-chip uit het middensegment en minder dan een twintigste van de grote Xilinx-chip. Dit is in overeenstemming met de kostprijs van de verschillende ontwikkelboards.
Zoals verwacht is het aantal flip-flops dat nodig is voor het ontwerp ongeveer hetzelfde bij de drie FPGA's; het exacte aantal is vooral een functie van het syntheseprogramma voor de logica. Tools voor logicasynthese repliceren flip-flops om de prestaties te verhogen of de routing te vereenvoudigen.
Uit de vergelijkingen van de snelheid blijkt dat alle drie de FPGA's ongeveer gelijk presteren, wat enigszins verrassend is aangezien de Lattice-FPGA niet de speciale DSP-blokken bevat die in de Xilinx-implementatie worden gebruikt.

Details Alchitry Cu

Listing 1 is rechtstreeks ontleend aan het Lattice-implementatierapport en toont de details van het resources-gebruik. **Figuur 1** is een deel van een screenshot van het Lattice floorplanner tool en toont de verdeling van de gebruikte logica-elementen over de chip. Deze verdeling is tamelijk gelijkmatig over de chip. ook al wordt in totaal slechts ongeveer 60% gebruikt.



Figuur 1. Lattice-floorplanner (Alchitry Cu).

Details Alchitry Au

Listing 2 is rechtstreeks ontleend aan het Xilinx-implementatierapport en toont de details van het resources-gebruik. Het volledige Xilinx-rapport is veel uitgebreider en behandelt alle specifieke hardware die beschikbaar is in de FPGA.
Figuur 2 komt uit een screenshot van het Xilinx floorplanner-tool en toont de verdeling van de gebruikte logica-elementen over de chip, samen met de verdeling van de specifieke logicablokken over

Listing 2. Gedeelte van het Xilinx XCA35T-rapport.

```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              | 2189 |     0 |     20800 | 10.52 |
|   LUT as Logic          | 2189 |     0 |     20800 | 10.52 |
|   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
| Slice Registers         | 1377 |     0 |     41600 |  3.31 |
|   Register as Flip Flop | 1377 |     0 |     41600 |  3.31 |
|   Register as Latch     |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                |    3 |     0 |     16300 |  0.02 |
| F8 Muxes                |    0 |     0 |      8150 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

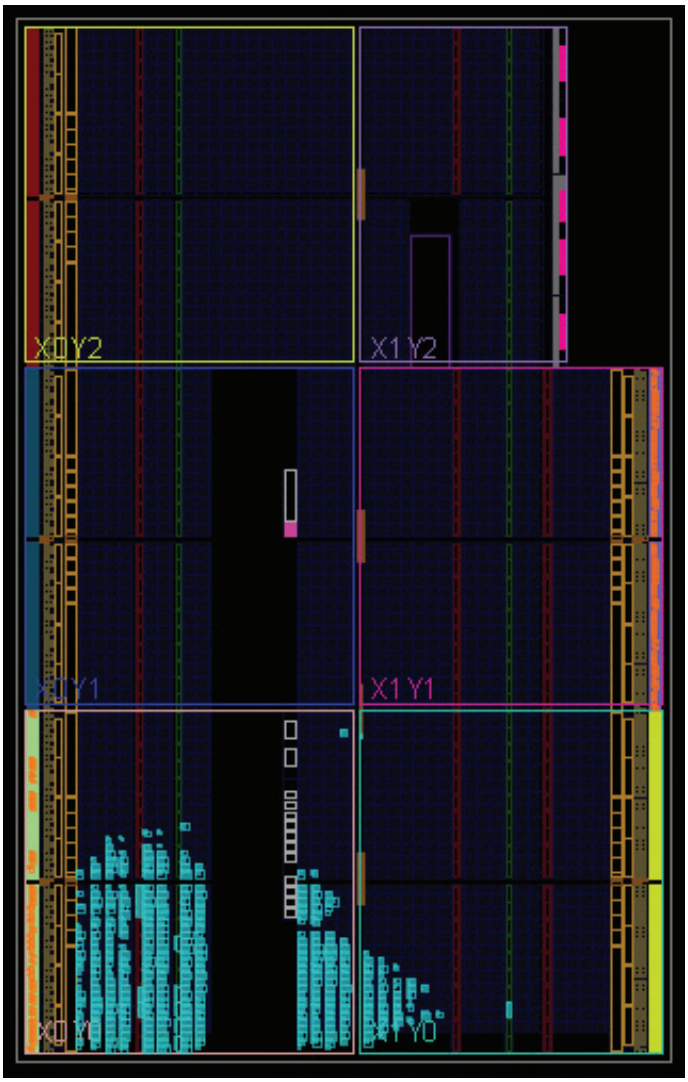
de chip. Het lijkt erop dat plaatsingstools altijd beginnen in de linkerbenedenhoek van een chip. Het is niet duidelijk waarom de logica in tweeën lijkt te zijn gesplitst.

Details Alchitry Au+

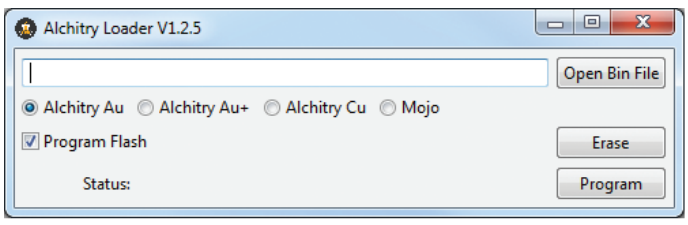
Listing 3 is ook weer ontleend aan het Xilinx-implementatierapport met details over het resources-gebruik. Merk op dat de cijfers voor gebruikte resources bijna gelijk zijn bij de beide Xilinx-chips.
Figuur 3 komt weer een deel van een screenshot van het Xilinx floorplanner-tool. Interessant is hier dat, hoewel deze FPGA driemaal zoveel logische elementen bevat, het totale gebruikte gebied vrijwel even groot lijkt als in de eerste Xilinx-implementatie.

Programmering van de hardware

Met het Alchitry Loader-programma kan de FPGA-bitstream eenvoudig naar een Alchitry-ontwikkelboard worden gedownload. Dit stand alone-programma wordt automatisch geïnstalleerd als onderdeel van de Alchitry Labs-software die bij Alchitry verkrijgbaar is. **Figuur 4** toont de gebruikersinterface van dit programma. Het programmeren van het flash-geheugen op het ontwikkelboard dat wordt gebruikt om de FPGA te laden is heel eenvoudig: specificeer het bitstream-bestand, selecteer het target-board en klik op de Program-knop.
Standaard zoekt het programma naar een bitstreambestand van het type .bin, dat standaard door de iCEcube2-software wordt gebruikt. Houd er bij het laden van het Xilinx-bitstreambestand rekening mee dat de Vivado-software een bitstreambestand van het type .bit genereert.

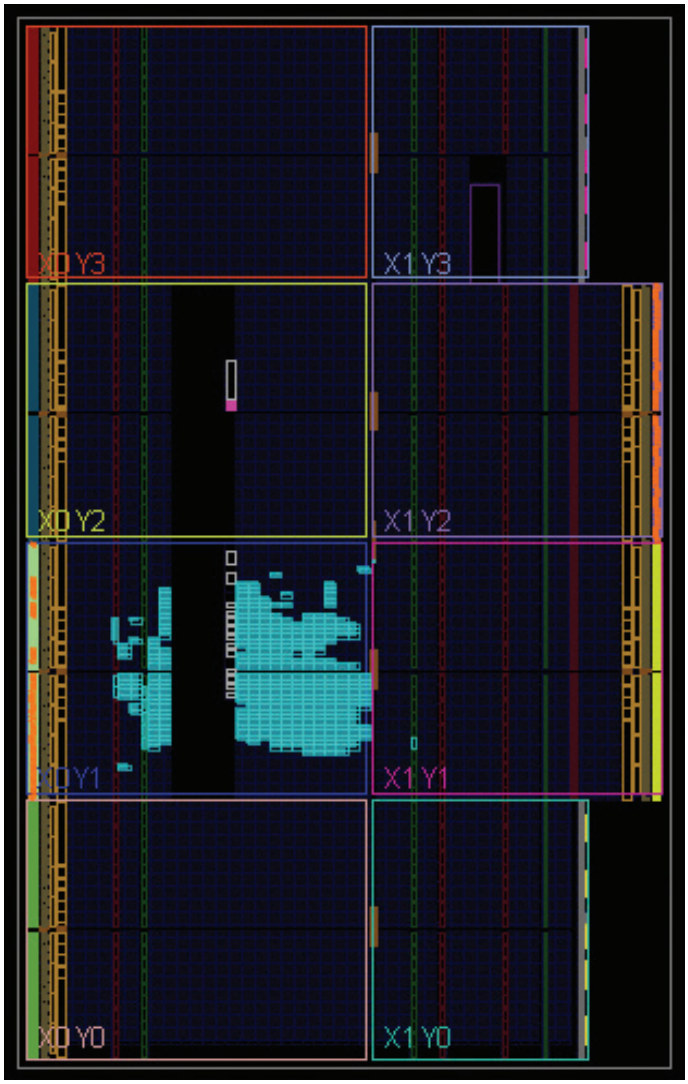


Figuur 2. Xilinx XCA35T-floorplanner (Alchitry Au).



Figuur 4. Alchitry Loader.

210347-03



Figuur 3. Xilinx XCA100T-floorplanner (Alchitry Au+).

Listing 3. Gedeelte van het Xilinx XCA100T-rapport.

```
1. Slice Logic
--------------

+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs              | 2185 |     0 |     63400 |  3.45 |
|   LUT as Logic          | 2185 |     0 |     63400 |  3.45 |
|   LUT as Memory         |    0 |     0 |     19000 |  0.00 |
| Slice Registers         | 1377 |     0 |    126800 |  1.09 |
|   Register as Flip Flop | 1377 |     0 |    126800 |  1.09 |
|   Register as Latch     |    0 |     0 |    126800 |  0.00 |
| F7 Muxes                |    3 |     0 |     31700 | <0.01 |
| F8 Muxes                |    0 |     0 |     15850 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

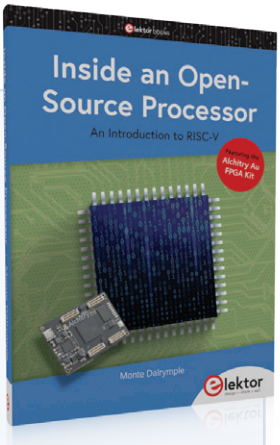
Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via monted@systemyde.com of naar de redactie van Elektor via redactie@elektor.com.

Een bijdrage van

Tekst en illustraties: Monte Dalrymple
Redactie: Jan Buiting
Vertaling: LinQuake
Layout: Giel Dols

GERELATEERDE PRODUCTEN



- **Boek: M. Dalrymple, Inside an Open-Source Processor (Elektor 2021) (SKU 19826)**
  www.elektor.nl/19826
- **E-boek: M. Dalrymple, Inside an Open-Source Processor (Elektor 2021) (SKU 19827)**
  www.elektor.nl/19827
- **Alchitry Au FPGA Kit (SKU 19638)**
  www.elektor.nl/19638