Digital Systems Development les 2 samenvatting

Digital System Design: DSP

Definitie: Via een FPGA kunnen we hardware ontwerpen die allerhande signalen kunnen verwerken. Enkele toepassingen die gebruik maken van DSP zijn: afbeeldingen & video verwerking, data compressie, ontvangen en verzenden van data enzovoorts.

Afbeelding met tekst, diagram, Lettertype, lijn

Automatisch gegenereerde beschrijvingVoorbeeld Audio processing: De analoge signalen worden extern verwerkt via een module en de digitale signalen via de FPGA.

Digital System Design: DSP FIR filter

Afbeelding met diagram, wit, lijn, origami

Automatisch gegenereerde beschrijvingDefinitie: een Finite Impulse Filter is een basiscomponent in de digitale signaalverwerking en filtering. Het neemt een functie als ingangsvariabele en zal aan de hand van coëfficiënten en trappen een reactie geven.

Digital System Design: DSP IIR filter

Afbeelding met diagram, schets, Technische tekening, Plan

Automatisch gegenereerde beschrijvingDefinitie: een Infinite Impulse Filter is een basiscomponent in de digitale signaalverwerking en filtering. Het neemt een functie als ingangsvariabele en zal aan de hand van coëfficiënten, trappen en eerder berekende waardes een reactie geven.

Digital System Design: DSP FIR filter tap

Definitie: Een DSP tap is een coëfficiënt/delay paar. Het aantal paren DSP-taps op een FPGA geeft aan hoeveel geheugen er nodig is om een filter te bouwen, het aantal berekeningen en de hoeveelheid filtering een filter kan doen.

Een keyword om te onthouden is MAC of (Multiply Accumulate)

Digital System Design: DSP FIR MAC

Definitie: In een FIR-context is de Multiply Accumulate de operatie om een coëfficiënt met een bijhorende data sample te vermenigvuldigen. In de context van een FIR-filter is er een MAC per tap nodig.

De meeste DSP-microprocessors voeren een MAC operatie uit per klok instructie.

Afbeelding met diagram, schermopname, lijn, Plan

Automatisch gegenereerde beschrijving

Voorbeeld:

* De hoeveelheid geheugen om de filter te implementeren is 24 bits
* De hoeveelheid berekeningen is aan de hand van het aantal taps te bepalen
* Deze taps bepalen het gedrag van de filter

Digital System Design: Hoe DSP op een FPGA uitvoeren

Definitie: DSP op een FPGA kan op verschillende manieren worden toegepast met name: een volledig seriële verwerking, een volledig parallelle verwerking of een mix van seriële en parallelle verwerking.

Afbeelding met diagram, Plan, lijn

Automatisch gegenereerde beschrijving

Digital System Design: Keuzes maken in de methode van verwerken

Definitie: Efficiënte keuzes maken in het ontwerpen van een digitaal systeem kan op verschillende manieren invloed hebben op de complexiteit en kostprijs van een project. Deze keuzes zijn dan ook belangrijk in het ontwerp van een design.

Enkele voorbeelden:

* Het parallel verwerken van verschillende laagfrequente signalen in een multiplex en daarna aan hogere snelheden verwerken zorgt ervoor dat het stroomverbruik lager wordt en er meerdere flip-flops worden bespaard.

Afbeelding met diagram, lijn, Lettertype, schermopname

Automatisch gegenereerde beschrijving

* Het serieel verwerken van ingangssignalen heeft vaak meer nut omdat de kwaliteit van deze filter hiervan niet hoog moet zijn.

Afbeelding met schermopname, tekst, diagram, lijn

Automatisch gegenereerde beschrijvingDigital System Design: Pipelining

Definitie: Het soepel en vlot laten verlopen van hardware timing en verwerking met het plaatsen van flip-flops voor de diverse processen.

* Op deze manier kunnen er voor operating frequenties gekozen worden
* Op deze manier is er een veel performantere performance van het systeem

Digital System Design: DSP Slicing

Definitie: omdat DSP-berekeningen zo frequent voorkomen op een FPGA hebben de meest boards DSP-modules aan boord om deze berekeningen nog sneller uit te laten voeren

Goed om te weten: Je kan zeker DSP blokken in hardware ontwerpen zonder de DSP-modules te gebruiken. Dit is echter vele malen trager en heeft geen speciale meerwaarde behalve extra complexiteit in een project.

Afbeelding met diagram, schermopname, tekst, lijn

Automatisch gegenereerde beschrijvingvoorbeeld:

een DSP slice op de Basys3 boards de DSP48E1 cell

Deze slices kunnen allerhande berekeningen serieel, parallel of een combinatie hiervan uitvoeren.

Digital System Design: video 7 series DSP resources

Definitie: This video introduces the DSP-slice features of the 7 Series FPGAs. In addition, discusses the Pre-Adder and Dynamic Pipeline control resources. (“7-Series DSP Resources - Xilinx”)

In deze video leer je:

* Het beschrijven van de voornaamste gebruikstoepassingen van DSP-slices
* Het beschrijven van DSP-slices in de 7 series FPGA

Digital System Design: video 7 series FIR Filter

Definitie: een FIR-filter is van nature een parallel proces

* N geeft het aantal taps weer
* N vermenigvuldigingen moeten parallel gebeuren voor een correcte werking (pipelining)

Afbeelding met tekst, diagram, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Digital System Design: video 7 series FIR Filter serieel versus parallel

Definitie: dit voorbeeld geeft duidelijk aan waarom we soms serieel tegenover parallel of omgekeerd kiezen. Het onderscheid tussen het aantal en de snelheid waarop de berekeningen worden uitgevoerd is hier duidelijk te zien.

Het verschil van de uitgangen van 303 KSP/s tegenover 600 MSP/s is bijzonder groot.

Afbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijving

Digital System Design: video 7 series FIR Filter serieel versus parallel

Definitie: een DSP-slice op de artix-7

Afbeelding met tekst, diagram, schermopname, lijn

Automatisch gegenereerde beschrijving



Afbeelding met tekst, diagram, schermopname, lijn

Automatisch gegenereerde beschrijving

|  |  |
| --- | --- |
| 1. Input conditioning blok met pipelining registers 2. Multiplier 3. Multiplier pipelining registers 4. Controleblokken voor de DSP-slice 5. De adder 6. Adder pipelining registers | * De multiplier en de adder zijn de basisblokken van de DSP-slice * Zijn de registers die de pipelining in de DSP-slice voorzien * De operation & control registers laten het toe om de DSP-slice functionaliteit aan te passen aan de noden van een project. |

Voorbeeld FIR-filter naar een DSP Slice

Digital System Design: video 7 series non-DSP functions

Definitie: de functionaliteit van een DSP-slice moet niet per se voor DSP-toepassingen gebruikt worden.

Waarom zouden we dit doen? Het antwoord is: Besparen van flip-flops en performance.

Afbeelding met diagram, Plan, lijn, pixel

Automatisch gegenereerde beschrijving

Voorbeeld:

De adder is een blok die voor verschillende toepassingen gebruikt kunnen worden

We gebruiken de adder van de DSP-blok zodat we deze niet in nodeloos complexe hardware moeten verwerken.

Digital System Design: video 7 series Symmetrische filters

Definitie: wanneer coëfficiënten van een filter symmetrisch zijn kunnen de kostprijs van performance en multiplicaties met de helft verminderd worden.

Via de DSP-slice kan dit worden uitgevoerd met hulp van de pre adder

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijvingAfbeelding met lijn, diagram, Perceel

Automatisch gegenereerde beschrijving

Digital System Design: video 7 series Symmetrische filters

|  |  |
| --- | --- |
| Zonder pre-adder | Met pre-adder |
| Afbeelding met schermopname, tekst, ontwerp  Automatisch gegenereerde beschrijving |  |

Digital System Design: video 7 series Dynamic pipeline control

Definitie: de functionaliteit van pippeling in een DSP-slice voor een bepaalde toepassing al dan niet gebruiken.

Waarom zouden we dit doen? We kunnen op deze manier dynamisch controleren wat twee pipeline registers doen op basis van een klok operatie. Op deze manier kunnen we het doel van het register op een dynamische manier gaan controleren.

Een voorbeeld: via een enkele DSP-slice kunnen we door het dynamisch te regelen van zijn registers complexe berekeningen uitvoeren. We gaan dus dynamisch de functionaliteit regelen. Op deze manier kunnen we een grote hoeveelheid snelheid winnen.

18 bit data gaan berekenen via de interne flip flops duurt vaak tientallen klokpulsen afhankelijk van de te berekenen waarde. Via een DSP-slice zijn dat er gegarandeerd vier.

Afbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijving

Voorbeeld: hetzelfde geld voor grote berekeningen

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Digital System Design: video 7 series IP support and inference

Definitie: de functionaliteit IP blokken

Voorbeeld 16 x 16 multiplier IP-blok:

Afbeelding met tekst, schermopname, menu, nummer

Automatisch gegenereerde beschrijving

Opzoeken -> DSP FIR, IIR IP blok voorbeelden

Digital System Design: video FPGA math signed vs unsigned.

Definitie: het proces van unsigned en signed getallen.

|  |  |
| --- | --- |
| Wanner kies je voor unsigned getallen. | Wanner kies je voor signed getallen. |
| * Wanneer u waardes onder nul kunnen gaan. * Complexe getallen * Complexe berekeningen | * Wanneer u waardes niet onder 0 kunnen gaan * Reële getallen * Discrete berekeningen |

Belangrijk in VHDL:

* Declareer u variabele signed of unsigned: signal factor1 : signed(7 downto 0)
* Gebruik de bibliotheek: iee.numeric\_std.all om foute en slechte berekeningen te voorkomen.

Digital System Design: video FPGA math berekeningen

Definitie: het uitvoeren van berekeningen op de FPGA

Belangrijk in VHDL:

* Bij optellingen of vermenigvuldigingen moet als basisregel de uitgang een bit groter zijn dan de grootst mogelijke uitkomst.
* Declareer u variabele signed of unsigned
* Probeer altijd te vermenigvuldigen of te delen met 2, shift left en shift right functies helpen hiermee.

Digital System Design: DSP moving average filter.

Afbeelding met tekst, lijn, Perceel, Lettertype

Automatisch gegenereerde beschrijving

Definitie: de moving avarage filter neemt het gemiddelde van een signaal over tijd.

Afbeelding met Lettertype, tekst, lijn, wit

Automatisch gegenereerde beschrijving

Voorbeeld Moving average filter in VHDL

|  |  |
| --- | --- |
|  | |
| * Declareren kloksignaal * Declareren enable en reset * Declareren van in en uitgangen |  |
| * Declareren van signalen * Starten van de architectuur * reset waarde = 0 * reset logica moving average filter |  |
| * reset waarde = 1 en rising edge * logica moving average filter * Eindigen van het proces |  |

Voorbeeld FIR-filter in VHDL

|  |  |
| --- | --- |
|  | |
| * Declareren entiteit FIR filter * Declareren kloksignaal * Declareren enable en reset * Declareren van in en uitgangen | aanpassen 8 bit naar 24 bit  aanpassen 16 bit naar 32 bit |
| * Declareren van het gedrag van de FIR filter component |  |
| * Bepalen van de coeficienten in de FIR filter |  |
| * Gedrag filter op de FPGA beschrijven |  |

|  |  |
| --- | --- |
| Afbeelding met diagram, cirkel, lijn, schermopname  Automatisch gegenereerde beschrijving | |
| * Entitieit IIR filter declareren * FIR filter vertaald naar het schema * Clk => kloksignaal FPGA * Sample\_valid\_in => check * Sample\_valid\_out => check * iir\_in => * iir\_out => * busy => check * multipliers forward * multipliers backward |  |
| * Start architectuur * Declareer status of positie van de FIR filter. * Multiplier registers Z1 * Multiplier registers Z2 * Pipelining blokken   Signal: temp  Signal: temp\_in, in\_z1, in\_z2, out z1, out z2 |  |
| * Start proces multiplier * Bij iedere klokpuls vermenigvuldig   Register multiplier input A met Register multiplier input B En schrijf dit naar het register multiplier output |  |
| * Start een proces op het ritme van de klok |  |
| * Klokpuls 1 * Laad het register van multiplier A in het register van de IIR input * Laad het register van multiplier A in het register van de IIR input * Bied het register van multiplier B aan a0 aan. * State = 1 * Busy = 1 | Optellen van waarde a0 |
| * Klokpuls 2 * Resize en shift van multiplier\_in omdat temp 40 bit groot is * Temp is 40 bit groot om resources te besparen. * Multiplier in A wordt in register in\_z1 opgeslagen * Multiplier in B wordt aangeboden aan a1 * State = 2 | Optellen van waarde a1 |
| * Klokpuls 3 * Resize en shift van multiplier\_in omdat temp 40 bit groot is * Temp is 40 bit groot om resources te besparen. * Multiplier in A wordt in register in\_z2 opgeslagen * Multiplier in B wordt aangeboden aan a2 * State = 3 | Optellen van waarde a2 |
| * Klokpuls 4 * Resize en shift van multiplier\_in omdat temp 40 bit groot is * Temp is 40 bit groot om resources te besparen. * Multiplier in A wordt in register out\_z1 opgeslagen * Multiplier in B wordt aangeboden aan b1 * State = 4 | aftrekken van waarde b1 |
| * Klokpuls 5 * Resize en shift van multiplier\_in omdat temp 40 bit groot is * Temp is 40 bit groot om resources te besparen. * Multiplier in A wordt in register out\_z2 opgeslagen * Multiplier in B wordt aangeboden aan b2 * State = 5 | aftrekken van waarde b2 |
| * Klokpuls 6 * Resize en shift van multiplier\_in omdat temp 40 bit groot is * Temp is 40 bit groot om resources te besparen. * State = 6 | Opslaan van de som in een register |
| * Klokpuls 7 * Resize van iir\_out en out\_z1 om deze voor te bereiden op een re-write * Doorsturen out\_z2 naar out\_z1 * Doorsturen in\_z2 naar in\_z1 * Doorsturen in\_z1 naar temp\_in * Sample\_valid\_out op 1 * Indicatie reeks samples klaar * State = 7 | verwerken van de data |
| * Klokpuls 8 * Voorbereiden nieuwe reeks samples * State = 0 * Busy = 0 |  |
| * Einde proces * Einde gedrag |  |