Labo Digitaal Ontwerp I
CPU Datapad

3ELICTS

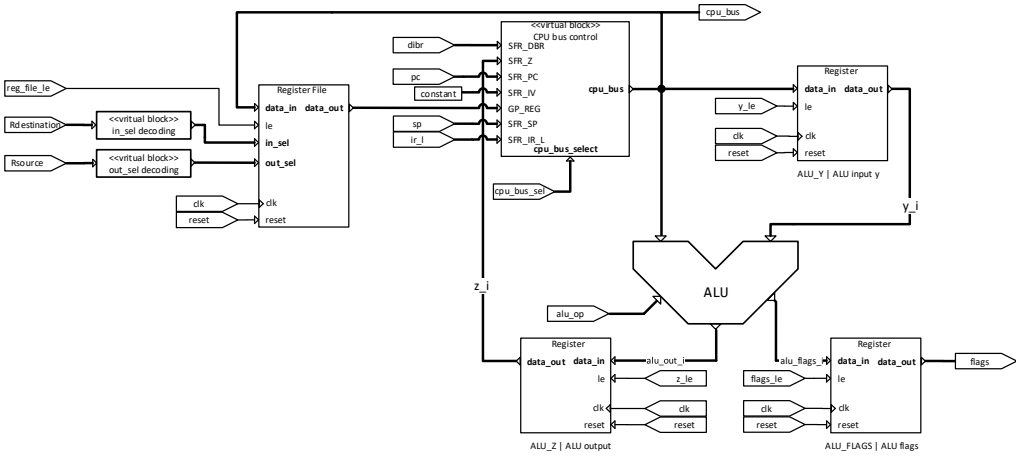
Geoffrey Ottoy, Stijn Wielandt & Bert Cox – KU Leuven

23 november 2018

1 Inleiding

Het pad dat de gegevens volgen binnenin een processor, over bussen van registers naar de ALU en terug, wordt het CPU datapad genoemd. Alles wat een processor doet –of dat nu het afspelen van een film is of het aansturen van een wasmachine– is uiteindelijk een opeenvolging van heel eenvoudige operaties. De data wordt verplaatst van registers naar de ALU, vervolgens bewerkt, waarna het resultaat opnieuw opgeslagen in registers. Eén zo'n cyclus (registers→ALU→registers) wordt de datapad cyclus genoemd.

Elke processor voorziet in een aantal *general purpose registers* die de programmeur kan gebruik om resultaten in op te slaan. Typisch worden deze worden in hardware gegroepeerd in een zogenaamde *register file*. De register file vormt dus een integraal onderdeel van het datapad.



Figuur 1: Blokschema van het datapad van onze processor.

In Figuur 1 staat het blokschema van het datapad van de processor die jullie ontwerpen. Daarin zijn o.a. de ALU en enkele `basic_registers` opgenomen; deze hebben jullie al in voorgaande opdrachten ontworpen. De CPU bus vormt de hoofdader van het datapad.

Via een bus multiplexer kan gekozen worden welke register een waarde op de CPU bus zet. Dit kan een *general purpose register* zijn (uit de register file), een extern register of het register dat het ALU resultaat bijhoudt (`Z`).

Daarnaast is er ook een register voorzien dat de ALU vlaggen zal bijhouden en een register dat de secundaire input van de ALU vasthoudt (`Y`).

Tijdens dit labo ontwerpen we eerst de registerfile. Vervolgens combineren we de verschillende designs tot een werkend datapad. De goede werking wordt geverifieerd aan de hand van simulaties. Na het doorlopen van deze opgave moet je in staat zijn om complexe bussystemen te ontwerpen en aan te sturen.

2 register_file.vhd

2-1 Maak een nieuw project aan in Vivado en ontwerp een register file.

2-1-1 Maak een nieuw project aan in Vivado. Geef een naam en locatie mee.

2-1-2 Voeg de voorziene template toe (`register_file.vhd`) en je eigen registerontwerp (`basic_register.vhd` – opgave L3).

2-1-3 Vervolledig de VHDL code zodat deze een register file beschrijft met dezelfde functionaliteit als de schakeling in Figuur 2.

2-1-4 Het aantal registers wordt bepaald door `C_NR_REGS`. Je kan het benodigde aantal registers instantiëren met behulp van de `for ... generate` syntax.

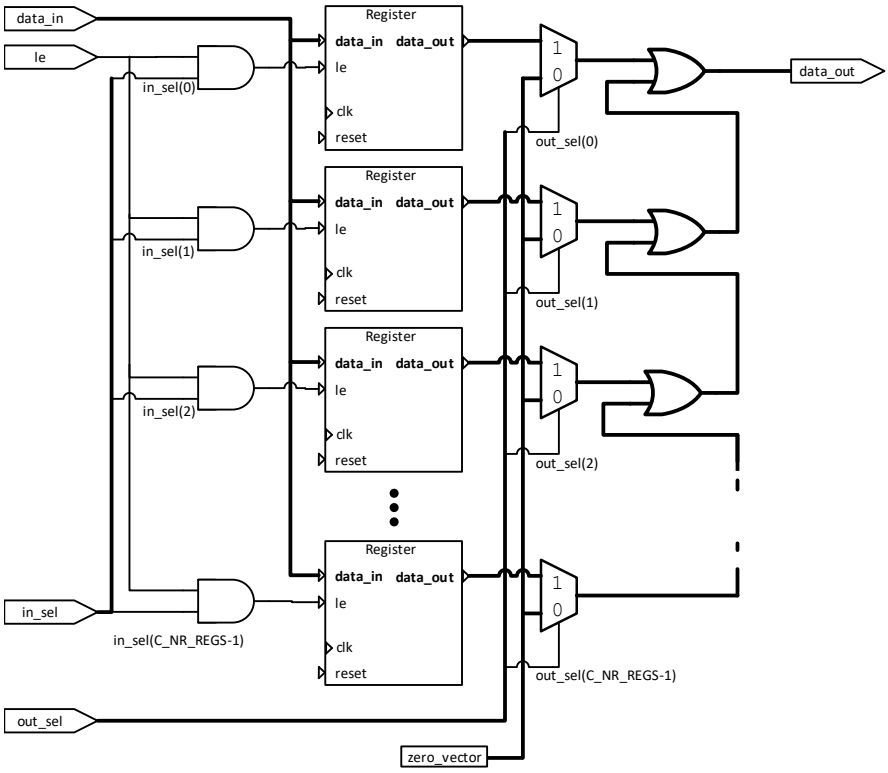
2-1-5 De sturing van de `data_out` uitgang doe je best met een aparte `for ... generate` constructie.

Hint: De eenvoudigste code krijg je als je een nieuw data type aanmaakt, namelijk een array met daarin `C_NR_REGS` `std_logic_vectors` van `C_DATA_WIDTH` breed.

2-2 Simuleer de werking van `register_file.vhd`.

2-2-1 Voeg de `register_file_tb.vhd` testbench file toe (via **Add Sources** → **Add Simulation Sources**).

2-2-2 Ga de werking van `register_file.vhd` door middel van simulatie.



Figuur 2: De register file bevat k n-bit registers. Een k-bit selectie-ingang bepaalt in welk register de n-bit data-ingang wordt ingeladen (als `le` hoog is en op de stijgende klokflank). Een tweede k-bit selectie-ingang bepaalt van welk register de uitgang naar de register file data-uitgang gekoppeld wordt. Om de eenvoud te bewaren zijn de klok en resetlijnen niet getekend. Het spreekt voor zich dat deze wel aangesloten worden.

3 data_path.vhd

3-1 Ontwerp een CPU datapad met register file en ALU.

3-1-1 Voeg de `data_pad.vhd` template file toe.

3-1-2 Voeg eveneens de ALU design files toe (`ALU.vhd`, `ADD.vhd`, `FA1B.vhd` en `processor_pkg.vhd`) die jullie voor opgave L2 hebben aangemaakt.

3-1-3 Vervolledig de template file zodanig dat deze dezelfde functionaliteit implementeert als in Figuur 1.

3-1-4 De register file ondersteunt 8 registers. Het input- en outputregister kunnen geselecteerd worden a.d.h.v. de datapad-ingangen `Rdestination` en `Rsource` (resp.). Dit zijn echter 3-bit ingangen. Er moeten dus decoders voorzien worden.

3-1-5 De CPU bus verbindt de verschillende data-ingangen van het datapad en de register file met de ALU `X`-ingang, het `ALU_Y` register en de registerfile.

3-1-6 Door middel van de `cpu_bus_sel` ingang wordt bepaald van uit welke bron er data op de CPU bus geplaatst wordt. Je kan gebruik maken van de daarvoor gedefinieerde constanten (in `processor_pkg.vhd`).

3-1-7 De interrupt vector (`IV`) is één van de bronnen van de CPU bus. Dit is een constante waarde (=2, voorgedefinieerd).

3-2 Simuleer de werking van `data_pad.vhd`.

3-2-1 Voeg de `data_pad_tb.vhd` testbench file toe.

3-2-2 Ga de werking van `data_pad.vhd` door middel van simulatie.

4 Het verslag.

Voor deze opgave maak je een verslag met daarin (minstens) de volgende elementen:

- Voor elk van de drie basisschakelingen:
  - De broncode. Zorg dat deze voorzien is van commentaar en degelijk is opgemaakt.
  - De RTL schema's (register file en datapad).
  - De simulatie output (TCL console) en golfvormen.[1].
- Een antwoord op deze vragen:
  - Wat is een interrupt vector?
  - Wat is de functie van het `ALU_Y` register?
  - Waarom wordt in de register file gebruik gemaakt van de structuur met de OR-poorten om de uitgang te controleren en niet van een multiplexer (wat eenvoudiger is)?
- Een besluit, bestaande uit 3 puntjes:
  1. Wat is gerealiseerd?
  2. Wat kan er beter? Wat zijn de beperkingen? Welke uitbreidingen zijn mogelijk?
  3. Hoe kunnen de voorgaande problemen opgelost en uitbreidingen uitgevoerd worden? Wees concreet.

Referenties

[1] William Stallings, *Computer Organization and Architecture – Designing for Performance*, Cha. 20, Control Unit Operation, pp.731–752. 2016.

[1] Voor de screenshots van de golfvormen kan je nog steeds gebruik maken van **Shutter**