Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

BTE5023 – Elektronische Systeme, Herbst 2019/2020

VHDL-Projekt RPN-Rechner

Markus Gafner Prof. Dr. Theo Kluter Prof. Dr. Torsten Mähne

2019-12-10

BFH-EIT, Biel/Bienne, Switzerland

Inhaltsverzeichnis

1 Zielsetzung

Ziel dieses VHDL-Projekts ist es mit Hilfe Ihres GECKO4-Education [1] einen einfachen Reverse Polish notation (RPN)-Rechner [2] zu implementieren. Dazu wird jeder Gruppe ein PmodKYPD™-Modul [3]–[5] von Digilent Inc. zur Verfügung gestellt. Auf diesem Modul (Abb. 1) befindet sich eine Tastatur mit den Ziffern Null bis neun, sowie den Buchstaben A bis F. Über diese Tastatur sollen Zahlen sowie auch die mathematischen Operationen eingegeben werden. Die Anzeige der eingegebenen Zahlen sowie der Rechenergebnisse soll über die Sieben-Segment-Anzeigen erfolgen. Der Inhalt des Stacks (Last In, First Out (LIFO)-Puffer), soll mittels der Light-Emitting Diode (LED)-Matrix visualisiert werden. Die Funktionalität ist gemäss den Vorgaben in Abschnitt 2 zu entwickeln. Die VHDL-Beschreibung wird in Quartus Prime [6] implementiert und auf dem GECKO4-Education-Board getestet.

2 Aufgaben

1. Vorbereiten des VHDL-Projektes auf dem GECKO4-Education-Board:
   - Erstellen Sie einen neuen Ordner mit der Bezeichnung `rpn_calculator/` mit folgenden Unterverzeichnissen:

Quelle: https://reference.digilentinc.com/reference/pmod/pmodkypd/start

Abbildung 1: Digilent PmodKYPD™

- `quartus/`
- `script/`
- `vhdl/`
- `sim/`

- Laden Sie die folgenden Tcl-Dateien von `https://gecko-wiki.ti.bfh.ch/` herunter und speichern Sie die Dateien im Ordner `script/`:
  - `clocks.tcl`
  - `led_array.tcl`
  - `pmod.tcl`
  - `seven_segment.tcl`
  - `switches.tcl`
- Erstellen Sie ein neues Quartus-Projekt in dem Ordner `quartus/` für Ihr GECKO4-Education-Board (FPGA-Typ: intel Cyclone IV EP4CE15F23C8).
- Modifizieren Sie die Tcl-Skripte für das Pin-Mapping gemäss Anleitung in Abschnitt 5.1 und führen Sie sie aus.
- Implementieren Sie die in Abb. 2 dargestellte Finite State Machine (FSM) zum Einlesen der Keypad-Eingaben. Verwenden Sie intern ein Register mit einer Breite von 5 bit zum Speichern der Eingabe: Das Most Significant Bit (MSB) zeigt, ob eine Taste gedrückt ist, und die verbleibenden 4 bit sollen den hexadezimal codierte Wert der gedrückten Taste repräsentieren. Der resultierende Funktionsblock soll über zwei Ausgänge verfügen: ein Strobe-Signal, welches signalisiert, dass eine neue Taste gedrückt wurde und ein Value-Signal, welches den Wert der gedrückten Taste repräsentiert.
- Erstellen Sie ein Schieberegister für 3 Ziffern mit jeweils 4 bit, welches die Ziffern auf der Sieben-Segment-Anzeige repräsentiert.
- Erstellen Sie ebenfalls die kombinatorische Schaltung zum Darstellen des Schieberegisterinhalts auf der Sieben-Segment-Anzeige.
- Verbinden Sie die erstellten Blöcke miteinander, so dass die letzten drei gedrückten Tasten auf den Siebensegmentanzeigen dargestellt werden. Synthetisieren Sie das Design, laden Sie es auf das GECKO4-Education-Board, und testen Sie es.
- Jetzt haben Sie die Grundfunktionalität des RPN-Rechner-Projekts auf das GECKO4-Education-Board.

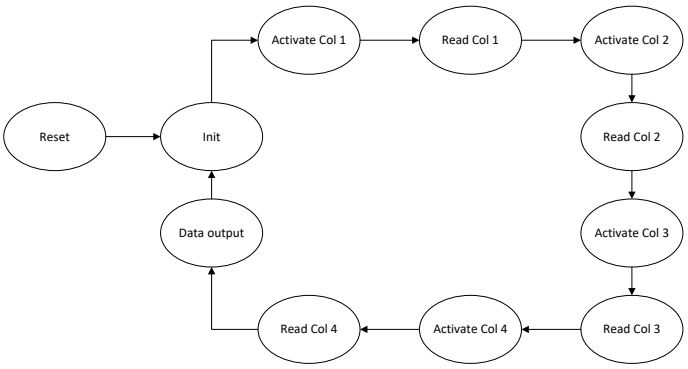


Abbildung 2: FSM to readout the keypad

2. Bedienung des Rechners:
   - Keypad: Ziffern Null bis Neun für die Eingabe einer Ziffer
   - Keypad: Taste A zum Addieren zweier Zahlen
   - Keypad: Taste B zum Subtrahieren zweier Zahlen
   - Keypad: Taste C zum Multiplizieren zweier Zahlen
   - Keypad: Taste D zum Dividieren zweier Zahlen
   - Keypad: Taste E zum Verschieben einer Zahl in den Stack
   - Keypad: Taste F zum Erzeugen eines Vorzeichenwechsels
   - Button SW6 (unten rechts) als Power on Reset (POR)
3. Interner Aufbau des Rechners
   - Dreistellige dezimale Anzeige inklusive Vorzeichen: Die jeweils aktuell eingegeben Zahl beziehungsweise das Rechenergebnis nach Auslösen einer Operation sollen mit den vier Sieben-Segment-Anzeigen dargestellt werden. Zur möglichst einfachen Ansteuerung der Anzeige bietet es sich an die Zahlen gegebenenfalls in die Binary Coded Decimal (BCD)-Darstellung umzuwandeln.
   - Stack/Stapel (LIFO-Puffer): mit über Generics einstellbarer Datenbreite und Tiefe. Dieser soll genutzt werden zum Zwischenspeichern von eingegebenen Zahlen im Bereich ± 999. Sehen Sie dazu 12 bit in Zweierkomplement- oder Sign-Magnitude-Darstellung vor. Damit sich die Implementierung des RPN-Rechners sowie die Visualisierung des Stack-Inhalts sich deutlich erleichtert, sollen zumindest die letzten 10 Werte vom LIFO immer an seinem Ausgang zur direkten Verwendung anliegen. Mit einem einen Taktzyklus langen *Read*-Strobe-Signal wird der letzte Wert aus dem LIFO gelöscht (Pop). Mit einem einen Taktzyklus langen *Write*-Strobe-Signal soll das Schreiben eines neuen Werts (Push) in das LIFO ausgelöst werden.
   - LED-Matrix: Visualisierung des Stack-Inhaltes (genauer der letzten 10 Einträge) auf Bit-Ebene.
4. Implementation der Rechner-Funktionalität:
   - Erstellen der VHDL-Beschreibung für einen LIFO-Puffer, in welchem die eingegebenen Zahlen zwischengespeichert werden.
   - Erstellen Sie die FSM, welche die Daten ins LIFO schreibt oder davon liest, das Resultat der gewünschten mathematischen Operation berechnet und dieses darstellt.

- Erstellen der VHDL-Beschreibung zum Darstellen des LIFO-Inhalts auf der LED-Matrix. Die erste Stelle leuchtet dabei immer, wenn ein gültiger Wert im LIFO steht, die zweite Stelle beschreibt das Vorzeichen, die nachfolgenden 10 Stellen visualisieren den Betrag der Zahl.
- Erstellen der VHDL-Beschreibung für die mathematischen Operationen zweier Zahlen:
  - Addition
  - Subtraktion
  - Multiplikation
  - Division

  Die Addition und Subtraktion können sie direkt unter Zuhilfenahme der entsprechenden Operatoren aus dem Paket `ieee.numeric_std` realisieren. Die Multiplikation und Division sind durch Sie jeweils manuell als rein kombinatorische Schaltungen zu realisieren in einer dedizierten Entity, möglichst mit der Bitbreite der Operanden als generischen Parameter `NBITS`. Das heisst, Sie dürfen die Operatoren `*` und `/` nicht verwenden! Der ganzzahlige Rest einer Division kann für dieses Projekt ignoriert werden. Eine Division durch Null ist als Fehler zu Signalisieren durch Ansteuerung des Dezimalpunkts auf der für die Darstellung des Minuszeichen reservierten Sieben-Segment-Anzeige ganz links! Eine Detektierung/Signalisierung von Overflows/Underflows des Rechenergebnisses ist nicht notwendig.

3 Ablauf

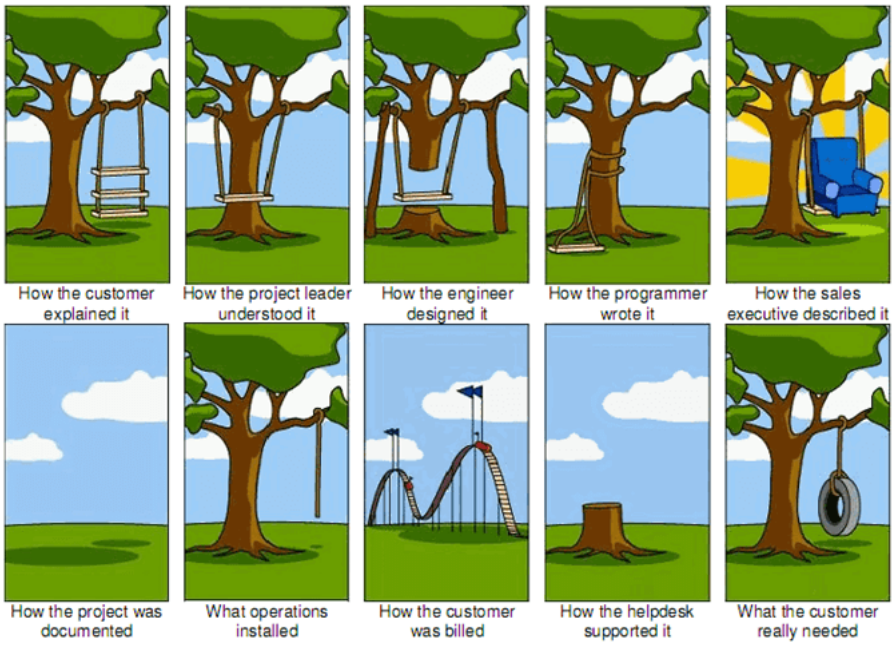
Das VHDL-Projekt soll in den folgenden Phasen ablaufen:

1. Bilden von Zweiergruppen
2. Analyse der Aufgabenstellung
3. Konzept (Blockschaltbild und Zeitplan, Abb. 3) zur Umsetzung der geforderten Funktionen
4. Definition der Module und Schnittstellen
5. Aufteilung der Arbeitsschritte innerhalb der Gruppe
6. Realisierung und Test der einzelnen Module
7. Test des Gesamtsystems
8. Dokumentation

4 Bewertung

Die folgenden Punkte werden bewertet:

- Entwurf: klar gegliedertes, hierarchisches und synchrones Design; keine Latches und keine Gated Clocks (5 %)
- Systematik bei der Durchführung des Projektes (5 %)
- Qualität der VHDL-Beschreibung: gute Lesbarkeit durch klare Bezeichner, korrekte Einrückung und Strukturierung mit Hilfe von Kommentaren, Einhaltung der Beschreibungsrichtlinien für synthetisierbare digitale Schaltungen, *klare Kennzeichnung der Autorschaft* im Kommentarkopf einer jeden VHDL-Datei. (5 %)



Quelle: https://www.tamingdata.com/2010/07/08/the-project-management-tree-swing-cartoon-past-and-present/

Abbildung 3: Project management tree swing cartoon

- Funktionalität des auf FPGA realisierten Gesamtsystems:
  - Keypad auslesen (10 %)
  - Ansteuerung der Sieben-Segment-Anzeige (5 %)
  - Binär-zu-BCD-Wandlung (5 %)
  - Dezimale Zahleneingabe (Vorzeichen sowie Schieberegister für die Ziffern) im Bereich von −999 bis 999 (5 %)
  - BCD-zu-Binär-Wandlung (Zweierkomplement oder Sign-Magnitude) (5 %)
  - generischer LIFO-Puffer inklusive Anzeige seines Inhalts auf der LED-Matrix (15 %)
  - kombinatorischer Addierer und Subtrahierer (5 %)
  - kombinatorischer Multiplizierer (10 %)
  - kombinatorischer Dividierer inklusive Division durch Null-Erkennung (10 %)
  - FSM zur Ablaufsteuerung des RPN-Rechners (5 %)
  - Integration und Test des Gesamtsystems (5 %)
- Dokumentation (5 %) in Form einer README-Datei, die mindestens folgende Punkte erläutert:
  - Ablauf des Projekts mit Arbeitsaufteilung in der Gruppe
  - Struktur des Projekts mit Liste aller relevanten Dateien, ihrer Funktion und der Autorschaft
  - Inbetriebnahme des Projekts auf dem GECKO4-Education
  - Stand des Projekts (Welche Funktionen sind vollständig umgesetzt? Welche Blöcke funktionieren nicht wie gewünscht? Welche Arbeiten sind noch offen?)
  - Alle von Ihnen erstellten Source-Dateien sind mit einem Kommentarkopf zu versehen, der zumindest folgende Informationen enthält: Name der Datei und Kurztitel, Autor(en), Erstellungsdatum, Projektname!

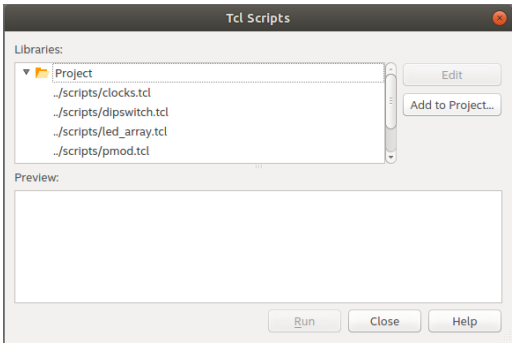


Abbildung 4: Tcl Scripts Dialog in Quartus Prime

5 Hinweise zur Umsetzung

5.1 Pin-Mapping mit Hilfe von Tcl-Skripten

Mit Hilfe von Tcl-Skripten lässt sich das Pin-Mapping einfach in Form von reinem Text konfigurieren. Man muss die I/Os also nicht mehr alle mühsam im Pin Planner konfigurieren. Es gibt für jeden I/O-Typ des GECKO4-Education-Boards eine separate Tcl-Datei [1]. Sie sollten die nötigen Tcl-Dateien bereits heruntergeladen haben (siehe Abschnitt 2). Die Tcl-Skripte können dann in Quartus eingelesen werden und Quartus führt direkt das Pin-Mapping durch.

1. Konfigurieren Sie die I/Os in den korrespondierenden Tcl-Skripten. Tragen Sie dazu einfach den Entity-Port-Namen des jeweiligen Signals am richtigen Ort ein. Hier einige Beispiele:
   - Wir werden auf dem GECKO4-Education den 50 MHz Takt verwenden. Setzen sie dazu in der Datei `clocks.tcl` folgende Zeile:

     ```
     set_location_assignment PIN_T1 -to CLK
     ```

   - Für Signale die mehr als 1 bit breit sind, wird einfach der Name des Signals und die Bit-Nummer in eckigen Klammern gesetzt. Beispiel Dual In-line Package (DIP) Switch:

     ```
     set_location_assignment PIN_V11 -to SW[0]
     ```

2. Fügen Sie die Tcl-Skripte dem Quartus-Projekt hinzu. Dazu gehen Sie auf „Tools → Tcl Scripts...“. Es öffnet sich der Dialog in Abb. 4.
3. Klicken Sie auf „Add to Project“, navigieren Sie in den Ordner `scripts/`, markieren Sie alle Tcl-Dateien, und klicken Sie schliesslich auf „Open“ (Abb. 5).
4. Wählen sie alle Skripte aus und klicken sie auf „Run“ (Abb. 6). Damit das Pin-Mapping angepasst.
5. Öffnen Sie den Pin Planner (`Ctrl-Shift-n`) und überprüfen Sie das Pin-Mapping.

Literatur

[1] T. Kluter, T. Mähne und D. Holzer, *GECKO4-Education*, Englisch, Wiki, BFH-EIT, Biel/Bienne, Switzerland, 2015–2017. Adresse: https://gecko-wiki.ti.bfh.ch/geck4education:start (besucht am 02. 12. 2018).

[2] Wikipedia contributors, *Reverse polish notation*, Englisch, in *Wikipedia, The Free Encyclopedia*, Wikimedia Foundation, Inc., Hrsg., 18. Nov. 2019. Adresse: https://en.wikipedia.org/wiki/Reverse_Polish_notation (besucht am 18. 11. 2019).

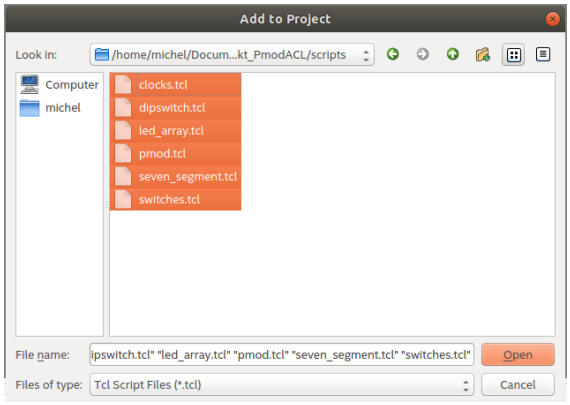


Abbildung 5: Hinzufügen der Tcl-Skripte zum Quartus-Projekt

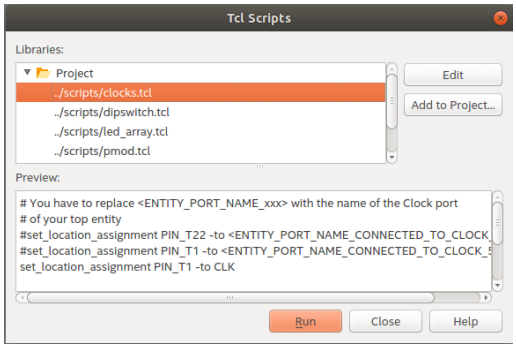


Abbildung 6: Ausführen der Tcl-Skripte in Quartus Prime

[3] Digilent Inc., *PmodKYPD™ reference manual*, Englisch, 502-195, rev. B, Digilent Inc., Pullman, WA, USA, 8. Mai 2016, 2 S. Adresse: https://reference.digilentinc.com/_media/reference/pmod/pmodkypd/pmodkypd_rm.pdf (besucht am 18. 11. 2019).

[4] ——, *PmodKYPD™ schematics, Keypad, connectors*, Englisch, 500-195, rev. B.0, Digilent Inc., Pullman, WA, USA, 12. Jan. 2010, 1 S. Adresse: https://reference.digilentinc.com/_media/reference/pmod/pmodkypd/pmodkypd_sch.pdf (besucht am 18. 11. 2019).

[5] ——, *PmodKYPD™ resource center*, Englisch, Digilent Inc., Pullman, WA, USA, 2012–2018. Adresse: https://reference.digilentinc.com/reference/pmod/pmodkypd/ (besucht am 18. 11. 2019).

[6] Intel. (24. Sep. 2018). Quartus Prime Lite Edition v18.1. Englisch, Adresse: http://fpgasoftware.intel.com/18.1/?edition=lite (besucht am 04. 12. 2018).