georg.von-der-brueggen [☺] tu-dortmund.de
daniel.kuhse [☺] tu-dortmund.de

Hardwarepraktikum
Sommersemester 23

Übungsblatt 6 – 24 Punkte

(Block B2 – insgesamt 68 Punkte)

Bearbeiten ab Samstag, 20. Mai 2023.
Abgabe bis spätestens Freitag, 26. Mai 2023, 23:59 Uhr.

Die sequentiellen Schaltungen auf den letzten Übungsblatt ermöglichen die bitweise Speicherung von Daten. In der Praxis werden Daten jedoch nicht in einzeln angesteuerten Flip-Flops gespeichert. Stattdessen werde sogenannte Schieberegister verwendet, die auf Flip-Flops aufbauen, aber auch weitere Funktionalitäten anbieten. Außerdem ermöglicht die Speicherung von Daten die Erstellung von endlichen Automaten.

6.1 Schieberegister in VHDL (8 Punkte)

Ein weiterer essentieller Baustein für die meisten sequenziellen Schaltungen ist das *Schieberegister*, welches für das Lagern und Bewegen von Daten zuständig ist, und deswegen gewöhnlicherweise für den Entwurf von Rechnern genutzt wird. Ein Schieberegister mit n Bits kann aus n Flip-Flops konstruiert werden, die über einen gemeinsamen Takt gesteuert werden. Bei jedem Takt sollen alle Bits des Registers gleichzeitig aktualisiert werden. Typischerweise gibt es vier Modi, in denen Schieberegister eingesetzt werden:

- Serial-in zu Parallel-out (SIPO): In diesem Modus werden die Eingabedaten seriell eingelesen und anschließend parallel ausgelesen.
- Serial-in to Serial-out (SISO): In diesem Modus werden die Eingabedaten seriell eingelesen und anschließend seriell ausgelesen.
- Parallel-in to Serial-out (PISO): In diesem Modus werden die Eingabedaten parallel eingelesen und anschließend seriell ausgelesen.
- Parallel-in to Parallel-out (PIPO): In diesem Modus werden die Eingabedaten parallel eingelesen und anschließend parallel ausgelesen.

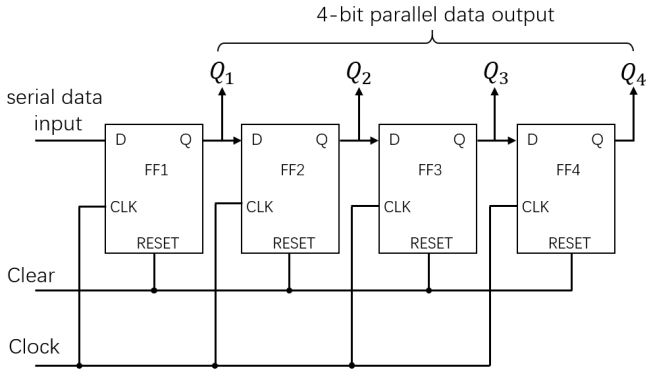

Abbildung 1: SIPO Schieberegister.

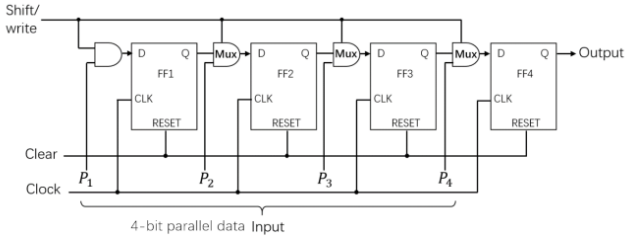

Abbildung 2: PISO Schieberegister.

Aufgaben:

a. (3 Punkte) Implementiert ein 4-Bit SIPO-Schieberegister auf Basis eines D-Flip-Flops in VHDL. Das entsprechende D-Flip-Flop wird Ihnen dabei zur Verfügung gestellt. (Das D-Flip-Flop ist ohne Ausgang $\overline{Q}$ implementiert, da dieser bei diesen Aufgaben nicht benötigt wird.)

b. (1 Punkt) Testen Sie das implementierte 4-Bit SIPO Schieberegister mit verschiedenen Eingabedaten.

c. (3 Punkte) Implementiert ein 4-Bit PISO-Schieberegister auf Basis eines D-Flip-Flops in VHDL.

d. (1 Punkt) Testen Sie das implementierte 4-Bit PISO Schieberegister mit verschiedenen Eingabedaten.

6.2 Bitmustererkennung (10 Punkte)

In dieser Aufgabe soll das das Verhalten eines Roboters implementieren werden, der sich vor einem LED-Bildschirm von links nach rechts bewegt. Auf diesem LED-Bildschirm wird eine Folge von Nullen und Einsen angezeigt. Bei jeder steigenden Taktflanke bewegt sich der Roboter ein Bit nach rechts. Am Kopf des Roboters ist ein Sensor installiert, der die Zahlen (0 und 1) erkennt. Jedes Mal wenn der Roboter in einer beliebig langen Eingabe die Sequenz 1011 erkennt, dann gibt er bis das nächste Bit gelesen wird einen Alarm aus. Der Ausgang Y ist 'TRUE', wenn der Roboter den Alarm auslöst und ansonsten 'FALSE'.

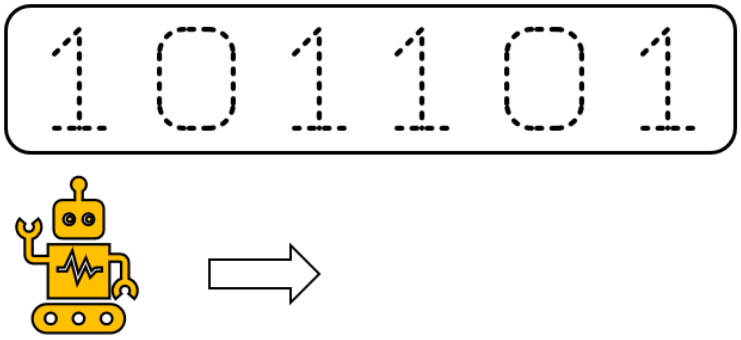

Abbildung 3: Roboter zur Bitmustererkennung.

Aufgaben:

a. (2 Punkte) Geben Sie die formale Beschreibung des Roboterverhaltens als Moore-Automaten an und zeichnen Sie danach den Automaten.

b. (2 Punkt) Unter den beigefügten Dateien finden Sie als Beispiel eine Implementierung des Roboters als Mealy Automat, der die Sequenz 1011 in beliebig langen Eingaben erkennt. Notieren Sie die Unterschiede und Vor- und Nachteile zwischen den beiden Automatentypen (Moore und Mealy), auch bezogen auf digitale Schaltungen.

c. (1 Punkt) Erklären Sie, warum diese Implementierung nicht leicht so erweitert werden kann, dass die zu erkennende Sequenz (bevor die Eingabe analysiert wird) als Parameter übergeben werden kann (wenn z.B. statt 1011 die Sequenz 1101 in einer beliebig langen Eingabe erkannt werden soll). Dazu können Sie zum Beispiel anhand einer Beispiel Eingabe erläutern welche Probleme dabei auftreten und wie sich das Verhalten des Roboters dadurch ändert. Gehen sie dabei davon aus, dass die zu findende Sequenz als 4-Bit Vektor anliegt. Der Roboter kann auf diese Sequenz also jederzeit zugreifen und muss sie nicht speichern.

d. (1 Punkt) Welches Probleme tritt zusätzlich auf, wenn die zu erkennenden Sequenz geändert werden soll wenn bereits mit dem Einlesen der Eingabe begonnen wurde?

e. (4 Punkte) Die in der vorherigen beiden Teilaufgaben beschriebenen Probleme sind mit Hilfe eines endlichen Automaten nur sehr schwer zu lösen. Verwenden Sie stattdessen eines der in Aufgabe 6.1 erstellten Schieberegister als Basis für den Roboter. Dieser soll die zu findende 4-Bit Sequenz über einen Parameter übergeben bekommen (diese muss als nicht gespeichert werden). Dazu können weitere Basisgatter verwendet werden die auch entsprechend angepasst werden können (z.B. indem die Anzahl der Eingänge verändert wird). Testen Sie Ihren Roboter mit Hilfe der vorgegebenen Testbench.

6.3 Synchroner Vorwärts-Rückwärts-Zähler (6 Punkte)

In dieser Aufgabe soll ein synchroner Vorwärts–Rückwärts-Zähler entworfen werden. Der Zähler soll vier Eingänge und zwei Ausgänge haben:

- Einen Reseteingang (Reset).
- Einen Takteingang (Takt).
- Einen Steuereingang (Count).
- Einen Steuereingang (Down).
- Zwei Ausgänge (Q1 und Q0, Q1 ist der höherwertige).

Die Schaltung soll mit jeder positiven Taktflanke zählen und den Zählerstand als Binärzahl an den Ausgängen Q1 und Q0 ausgeben. Im Grundzustand, d.h. vor dem Einschalten des Taktes, soll die Schaltung sich im Zählerstand Q1Q0 = 00 befinden.

- Wenn am Steuereingang Count = 0 anliegt, soll die Schaltung nicht zählen.
- Wenn am Steuereingang Count = 1 anliegt, soll die Schaltung zählen.
- Wenn am Steuereingang Down = 0 anliegt, soll die Schaltung vorwärts zählen (Count=1).
- Wenn am Steuereingang Down = 1 anliegt, soll die Schaltung rückwärts zählen (Count=1).

Die Schaltung soll mit zwei JK-Flipflops realisiert werden. Die miteinander verbundenen Takteingänge der beiden Flipflops bilden den Takteingang der gesamten Schaltung.

Die Zustände des zu entwerfenden Automaten sind identisch mit den Ausgangszuständen, d.h. die Ausgabefunktion ist die identische Abbildung. [0,0], [0,1], [1,0], und [1,1] sind die vier möglichen Zählerzustände des Automaten. Es gilt: [0,0] = S0, [0,1] = S1, [1,0] = S2, [1,1] = S3.

Hinweis: Entwerfen Sie einen Zähler ohne Überlauf (wenn von 11 hochgezählt wird bleibt es bei 11).

Aufgaben:

a. (2 Punkte) Zeichnen Sie den entsprechenden Mealy-Automaten auf, und geben Sie die formale Beschreibung an.

b. (3 Punkte) Implementieren Sie den Zähler in VHDL. Nutzen Sie dazu zwei JK-Flipflops. Testen Sie Ihre Implementierung der Komponenten und des gesamten Zählers in einer Testbench (hier kann der Reset ignoriert werden).

c. (1 Punkte) Ergänzen Sie die Schaltung durch einen synchronen Reset (Schalter mit R), der es ermöglicht, den Grundzustand Q0 = Q1 = 0 jederzeit einzustellen und testen Sie diesen.

Hinweis: Aufgabenteil b und c können auch zusammen abgegeben werden (müssen Sie aber nicht). Geben Sie dies bitte entsprechend an. In diesem Fall muss die zugehörige Testbench sowohl die korrekte Funktion des Zählers als auch den Reset überprüfen.