FH-OÖ/Hagenberg/HSD
Chip-Design 3 — CHD3
Markus Pfaff © 2023
Übung 5
Asynchron sequenzielle Schaltungen

1. Aufgabe *Einkomponentenübergang → Logik-Hazard: statischer Hazard → Fehlfunktion — Logik-Hazard beseitigen*

Im Verzeichnis `unitFeedbackMux` finden Sie eine VHDL-Beschreibung des rückgekoppelten Multiplexers aus der Vorlesung. Diese Schaltung stellt ein Latch dar. Verwirklicht wurde sie im VHDL-Modell durch Verwendung zweier Primimplikanden.

Die Schaltung ist für die gewünschte Funktion optimiert. Es sind also nicht alle Primimplikanden (aus dem KV-Diagramm) in der Schaltungsstruktur vorhanden. Aus diesem Grund können bei dieser Schaltung statische Hazards bereits in Folge von Einkomponentenübergängen (ein einziger Eingang wechselt seinen Wert) auftreten. Dies soll in der Simulation gezeigt werden.
Erstellen Sie daher eine *Testbench*, welche die interessanten Übergänge für die Eingangssignale `iD` und `iEn` an das *Latch* anlegt und dessen Verhalten beobachtbar macht. Es sollen also insbesondere jene Übergänge an den Eingängen angelegt werden, welche statische Hazards verursachen. Da es sich um Einkomponentenübergänge handelt, darf stets nur der Wert an einem Eingang wechseln. Nie also mehr als ein Eingang gleichzeitig, denn dann würde ein Mehrkomponentenübergang vorliegen.
Tritt ein *Hazard* auf, kann dies zu einer Fehlfunktion führen. Ursache hierfür ist, dass der Ausgangswert ebenfalls wieder als Eingangswert verwendet wird. Beim entsprechenden Übergang am passenden Eingang ist dann eine Fehlfunktion des Latches zu beobachten. Bevor dies nicht eindeutig gezeigt ist, bitte nicht in der Bearbeitung fortfahren!

Variieren Sie die Verzögerungszeiten für die Berechnung der *signals* `nEn` und `Impl1` so, dass letzterer Wert größer, kleiner oder gleich dem ersten Wert ist. Was ist in der Simulation zu beobachten? Wie erklären Sie sich diese Beobachtungen?
Verändern Sie nun die Verzögerungszeiten für die Berechnungen von `Impl2` und `Yn`. Welcher Einfluss auf das Simulationsergebnis ist feststellbar? Wie erklären Sie sich diesen?
Nun soll eine zweite *architecture* für das *Latch* erstellt werden (Name: `Hazardfree`), die sich von der ersten durch Hinzufügen eines dritten Primimplikanden unterscheidet, welcher den kritischen Übergang absichert. Mit dieser Maßnahme sind alle Primimplikanden für die gewünschte Schaltungsfunktion auch in der Schaltung vorhanden. Mit dieser Maßnahme werden Logik-*Hazards* beseitigt. Ein Einkomponentenübergang wird dann nicht mehr zu einem *Hazard* führen.
In der Simulation soll dieses Modell parallel zum bereits existierenden mit den gleichen Eingangswerten versorgt werden. So kann das Verhalten des neuen Modells direkt mit dem alten in der *Waveform* verglichen werden. Kann der dritte Primimplikand die gewünschte Funktion des *Latches* garantieren? Auch dann wenn Verzögerungszeiten gewählt werden, die sich im vorhergehenden Modell als problematisch erwiesen?

2. Aufgabe *Mehrkomponentenübergang → Funktions-Hazard → Fehlfunktion — Setup- und Hold-Zeiten*

In dieser Aufgabe soll gezeigt werden, dass in einer asynchronen Schaltung funktionale *Hazards* auftreten können, sobald mehrere Eingänge zur (annähernd) gleichen Zeit einen Wertewechsel haben (Mehrkomponentenübergang), obwohl alle Logik-*Hazards* in der Schaltung mit der Methode aus der vorigen Aufgabe eliminiert wurden.
Solche Mehrkomponentenübergänge führen zu *Hazards*, deren Folge auf Grund der Rückkopplung in der Schaltung wiederum Fehlfunktionen sein können. Diese Art der *Hazards* werden funktionale oder Funktions-*Hazards* genannt.
Der `FeedbackMux` implementiert ein pegelgesteuertes D-Latch. Bei einem solchen Latch spielen *Setup-* und *Hold-*Zeit eine Rolle.
Überlegen Sie sich zunächst die Antworten auf diese beiden Fragen:

- Welcher Zusammenhang besteht zwischen der Verletzung dieser Zeiten und Mehrkomponentenübergängen?
- Handelte es sich beim kritischen Übergang, den Sie in der vorigen Aufgabe gefunden haben um einen Mehrkomponentenübergang?

Verwenden Sie für diese Aufgabe die *architecture* `Hazardfree` aus der vorigen Aufgabe. Ergänzen Sie ihre *testbench* so, dass *alle* überhaupt möglichen Übergänge der Werte von `iD` und `iEn` an das *Latch* angelegt werden: Ausgehend von den vier möglichen Kombinationen der beiden Eingangssignale können jeweils drei weitere Kombinationen erreicht werden. Es sind also zwölf Übergangssituationen zu berücksichtigen[1] .
Die beiden Werte sollen nicht nur genau gleichzeitig, sondern auch zeitlich leicht versetzt geändert werden. Der Versatz von `iD` soll relativ zur Wertänderung an `iEn` gemessen werden. Ein Versatz von 4 ns bedeutet, dass die Änderung von `iD` um 4 ns nach der Änderung von `iEn` erfolgt. Ein Versatz von -4 ns bedeutet, dass die Änderung von `iD` um 4 ns vor der Änderung von `iEn` erfolgt. Erweitern Sie Ihre *Testbench* so, dass alle Übergangssituationen mit einem Zeitversatz von -10, -9, -8,... 0,... 9, 10 ns automatisch durchsimuliert werden.
In der Simulation kommt es je nach Zeitversatz zu bemerkenswerten Ergebnissen. Wie äußern sich diese? Was hat dieser Zeitversatz mit den Begriffen *Setup-* und *Hold-*Zeit zu tun? Gehen Sie davon aus, dass `iEn` die Rolle eines Takteingangs hat.

3. Aufgabe *Schaltplan DE1-SoC*

- Das FPGA ist im Schaltplan auf mehrere Symbole aufgeteilt. Erläutern Sie diese Aufteilung: Warum wird es aufgeteilt? Was ist die Aufgabe der einzelnen Blöcke?
- Auf *Sheet* 7 ist rechts oben der Steckverbinder für einen Lüfter und die dazugehörige Leistungselektronik zu erkennen. Welches Bauteil soll gekühlt werden?

  Suchen Sie zunächst ein Datenblatt der seltsamen Diode SM2T3V3A. Hierbei handelt es sich um eine (vom Hersteller ST) so genannte „Transil-Diode“ (siehe z.B. Wikipedia). Wie schaut ihre u/i-Kennlinie aus? Worin besteht der Unterschied zu einer Zenerdiode? Welche Funktion wird durch diese Diode in der Schaltung erfüllt?

[1] Da nur Fälle interessant sind, bei denen eine Änderung von *iEn* vorliegt, kann man die Anzahl der zu simulierenden Fälle gegenüber diesen Möglichkeiten reduzieren.

Welche Funktion erfüllt der Leistungs-MOSFET FDV305N? Ist dies ein N- oder ein P-MOSFET?

Die Steuerung des Lüfters erfolgt über die Leitung `FAN_CTRL`. Wer steuert diese Leitung an? Wie kann die Temperaturmessung stattfinden?

Welches grundlegende Problem muss vor dem Einstecken eines Lüfterkabels in den Stecker `J16` überwunden werden? Was bedeutet in diesem Zusammenhang die Abkürzung *DNI* im Schaltplan, die sich bei den Bauteilen für die Lüftersteuerung befindet?

- Über den DIP-Switch `SW10` auf der Platinenunterseite wird die Art der Konfiguration des FPGAs eingestellt. Der Standardwert ist im Schaltplan mit angegeben (*Sheet* 7). Passt die Schalterstellung auf ihrem Board zur Angabe im Schaltplan? Welche Werte werden hierdurch an den entsprechenden Leitungen des FPGAs eingestellt? Warum entspricht die Schalterstellung *ON* laut Aufdruck auf dem Schaltergehäuse einer `'0'`?
- Was ist die Bedeutung der Widerstände `R138`... unterhalb der Schaltung für den DIP-Switch? Beachten Sie, dass auch dort wieder *DNI* angemerkt ist. Wo befinden sich diese Widerstände auf der Platine?
- Wie erfolgt die Konfiguration in der Standardeinstellung *AS*?
- Finden Sie die Bedeutung der FPGA-Pins `CONF_DONE`, `NSTATUS`, `NCONFIG` und `NCE` heraus (*Cyclone V Handbook*).

4. Aufgabe *Fakultativ: Flankengesteuertes D-FlipFlop; Setup-, Hold-Zeit*

Die in Abb. 1 dargestellte asynchron sequenzielle Schaltung stellt eine von vielen Möglichkeiten zur Realisierung eines flankengesteuerten D-FlipFlop dar. Dieses verfügt über einen Dateneingang `D`, einen Takteingang `C`, sowie optional über je einen (taktunabhängigen, also asynchronen) Setz- (`nPr`) und Löscheingang (`nClr`).

In Abb. 2 ist die gleiche Schaltung nochmals dargestellt, wobei die drei Rückkopplungszweige explizit herausgezeichnet sind. Die Setz- und Löscheingänge sind dort der Übersichtlichkeit halber nicht dargestellt.

Beschreiben Sie das D-FlipFlop in VHDL. Von den optionalen Eingängen brauchen Sie nur den Eingang `nClr` zu implementieren. Die Gatter können Sie mit dem Nand-Operator (`nand`) modellieren. Allerdings ist bei Gattern mit drei Eingängen Vorsicht geboten, da `A nand B nand C` nicht das selbe ist, wie `not (A and B and C)`. Richtig ist in diesem Fall der zweite Ausdruck. Warum?
Als Verzögerungsmodell sollte *inertial* mit einstellbarer Trägheit verwendet werden (z.B.: `... <= reject gRejectDuration3 inertial ... after gPropagationDelay3;`).
Diese Verzögerungszeiten sollten für jedes Gatter (daher die Zahl in den Namen) als *generics* in das Modell eingespeist werden. Die konzentrierten Verzögerungselemente aus Abb. 2 sind für unsere Zwecke eine zu ungenaue Modellierung, weshalb auf sie verzichtet wird. Im Verzeichnis `unitPosEdgeTrigDFF` finden Sie bereits entsprechend vorbereitete Dateien.

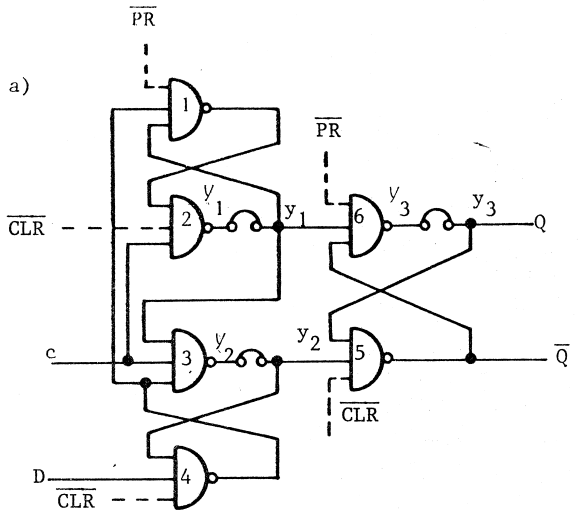

Abbildung 1: Positiv-Flankengesteuertes D-Flipflop realisiert durch eine asynchron sequenzielle Schaltung. Die Rückkopplungen sind in dieser Schaltung durch kurzgeschlossene Klemmen kenntlich gemacht. Der Überstrich kennzeichnet ein *active-low*-Signal.

Überprüfen Sie die Funktion Ihrer Beschreibung mit einer geeigneten *testbench*. Zunächst stellen Sie alle Verzögerungszeiten auf 0 ns.

Kommen wir zum Kern der Sache: Legen Sie die Trägheitszeit für alle Gatter auf 2 ns und die Verzögerungszeit auf 9 ns. Nun sollen Sie sich auf die Jagd nach der *Setup*- und der *Hold*-Zeit des FlipFlops machen.

Gestalten Sie hierzu Ihre *Testbench* so (um), dass die Zeitabstände zwischen Änderungen von `C` und `D` automatisch variieren (d.h., dass diese automatisch über einen bestimmten Zeitraum verändert werden). `nClr` setzen Sie für diesen Test auf `'1'`. Besonders bequem ablesbar sind die kritischen Zeiten, wenn in der *waveform* auch der Zeitabstand zwischen einer Änderung der Daten und der steigenden Taktflanke mit angezeigt wird. Dieser Zeitunterschied kann beispielsweise als Integer-Wert in einer *for loop* automatisch über den in Frage kommenden Bereich variiert werden. Aus dem Integer-Wert kann durch Mulitplikation mit `1 ns` eine Zeitdauer gemacht werden.
Beachten Sie bitte, dass das Flip-Flop sich unterschiedlich verhalten kann, je nachdem ob der D-Eingang von `'0'` oder in die umgekehrte Richtung wechselt.
Notieren Sie die Zeiten, die zu einer Verletzung der

- *Setup*-Zeit und der
- *Hold*-Zeit

führen.

Müssen Sie auch einen bestimmten Zeitbezug zwischen `C` und `nClr` gewährleisten? Worin besteht Ihrer Meinung nach das Fehlverhalten, wenn Sie `nClr` knapp vor/nach einer steigenden Flanke ändern? Ist die Richtung der Änderung wichtig?

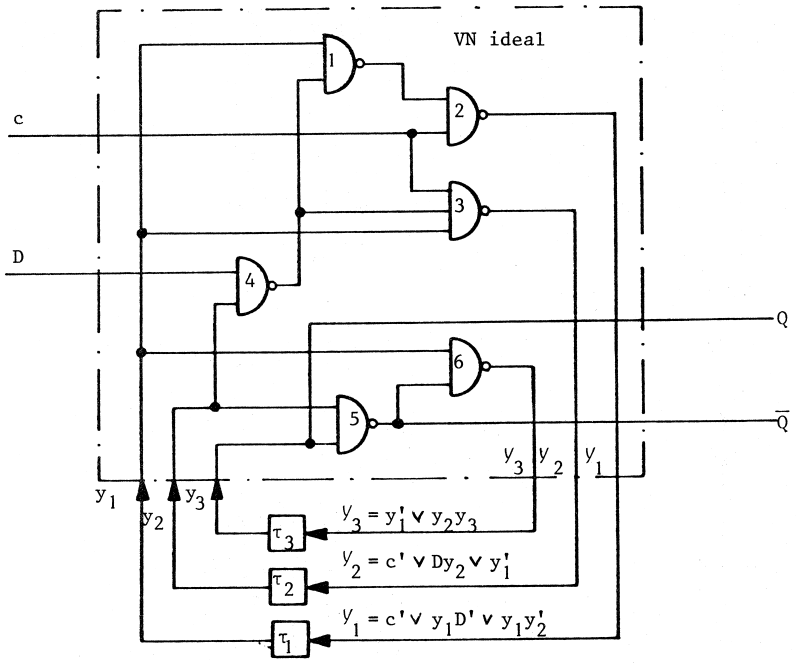

Abbildung 2: Die gleiche Schaltung wie in Bild 1, allerdings ohne die Eingänge für die Signale `nClr` und `nPr` (der Überstrich kennzeichnet wieder ein *active-low*-Signal). Schaltnetz und Rückkopplungen sind getrennt gezeichnet. Die Rückkopplungsleitungen enthalten konzentrierte Verzögerungselemente, die jedoch in unserem Modell nicht verwendet werden sollen.