Versuch 4: Sequentielle Logik

Einführung

Im Versuch 4 wird wieder mit der PLD-EntwicklungssoftwareAltera MAX+plusII und dem PLD-Board SB2 gearbeitet. Der Versuch besteht aus vier Teilen die in der angegebenen Reihenfolge gelöst werden müssen, Teil 4 ist fakultativ:

1: Latches 2: Flip-Flops 3: Asynchron Zähler 4: Synchron Zähler

Latches und Flip-Flops sind die elementaren Bausteine zum Aufbau sequentieller Logik. Im Gegensatz zu kombinatorischer Logik werden damit Speichereigenschaften realisiert: sequentielle Logik hat ein Gedächtnis, die Ausgänge sind nicht nur eine Funktion der aktuellen Eingänge sondern und vor allem auch der Vergangenheit. Auf Torniveau erkennt man eine sequentielle Schaltung sehr leicht am Vorhandensein von Rückführungen, womit geschlossene Signalkreise entstehen (machen Sie den Test mit Ihnen bereits bekannten Schaltungen!).

Das Lösungsblatt zum Versuch erhalten Sie am Praktikumsnachmittag.

Vorbereitung am Praktikumsnachmittag:

Im Praktikumsverzeichnis DigiPrakt ein neues Arbeitsverzeichnis V4 anlegen (falls ein solches von Ihren Vorgängern her bereits existiert, dessen gesamten Inhalt löschen). In dieses den 7-Segment Decoder von Versuch 2 und den Addierer von Versuch 3 (nur für die faktultative Aufgabe 4 notwendig) laden (Ihre Files oder das bereitgestellte Paket Versuch 3 von der DigiPrakt Seite).

1: Latches

Einführung

Latches sind die einfachsten sequentiellen Schaltungen. Sie bestehen im Kern aus zwei im Kreis verschalteten invertierenden Toren (Ausgang 1 –> Eingang 2, Ausgang 2 -> Eingang 1). Dadurch entsteht die einfachst mögliche Speicherschaltung. Die Schaltung bleibt in einem Zustand, auch wenn die Eingangssignale die sie in diesen Zustand gebracht haben, nicht mehr vorhanden sind.

Aufgaben

1. RS – Latch. Legen Sie ein neues Grafik Editor File rs_latch.gdf an (dieses und alle weiteren im Verzeichnis V4 speichern). Zeichnen Sie das Schaltschema eines RS-Latches mit zwei NOR-Toren (Beuth 7.3). ➨ **B**

   Die beiden Eingänge S (Set) und R (Reset) mit den Drucktasten KA und KB ansteuern.

   Die Signale an den beiden Ausgängen (Q1 und Q2) auf den Anzeigen **B = Q1** und **A = Q2** darstellen. Dazu die 0-1 - Anzeigeschaltung nach Versuch 2 Teil 1 verwenden (nicht den vollständigen 7-Segment Decoder).

Das RS-Latch compilieren und in den PLD programmieren (Pin-Locking nicht vergessen).

Wahrheitstabelle experimentell bestimmen, das Verhalten beschreiben.

2. Was passiert, wenn Sie den irregulären Fall (Set und Reset gleichzeitig aktiv) durch 'gleichzeitiges' loslassen der beiden Drucktasten verlassen?

   Mehrmals wiederholen mit allen Gruppenmitgliedern und vertauschten Fingern, das Ergebnis zu erklären versuchen.

3. Ergänzen Sie das elementare RS-Latch mit zwei Eingangstoren (Einlesetore, AND) und einem Taktsignal C (erzeugt mit dem Schalter nSA1).

   Erweitertes RS-Latch compilieren und programmieren.

   Experimentell die Wahrheitstabelle bestimmen und den Unterschied im Verhalten gegenüber Aufgabe 2 beschreiben.

4. Ergänzen Sie das erweiterte RS-Latch zu einem D-Latch (D Eingang mit Drucktaste KA, Takt C mit Drucktaste KB erzeugen).

   D-Latch compilieren und in den PLD programmieren.

   Wahrheitstabelle experimentell bestimmen und das Verhalten beschreiben.

5. Versuchen Sie aus dem D-Latch durch die Einführung einer Rückkopplung vom Q Ausgang über einen Inverter auf den D-Eingang ein T-Latch (T für toggle) zu realisieren.

   Wichtig: Die Rückführung muss über Pins realisiert werden, von **nDA5 auf nSA3** (**SA3** muss in Stellung **off** sein). Verbinden Sie dazu die entsprechenden Kontaktstifte **HW11 und HW9** mit einem **Jumpe**r (roter Kurzschluss-Stecker).

   Bei jedem Taktimpuls soll das Inverse des aktuellen Zustandes ins Latch eingelesen werden. Dies entspricht einem 1-Bit Zähler: mit jedem Taktimpuls (Betätigung von KB) soll das Latch seinen Zustand wechseln. Überprüfen Sie die Funktion experimentell, funktioniert es wie erwartet oder was beobachten Sie?

Um dem unerwarteten Verhalten unseres T-Latches auf den Grund zu gehen, müssen wir das Ausgangssignal mit dem KO ansehen. Verwenden Sie dazu unbedingt den **10:1 Tastkopf**.

6. Stellen Sie das Ausgangssignal Q des T-Latches auf dem KO bei gedrückter Taste KB (Takt = 1) dar. ➨ **W**

   Was stellen Sie fest und wie erklären Sie sich das Verhalten ?

2: Flip-Flops

Einführung

Flip-Flops sind die universellsten sequentiellen Schaltungen. Im Gegensatz zu den Latches (die auch als Ein-Speicher-Flip-Flops bezeichnet werden), kann sich der Ausgang eines Flip-Flops nur bei einer Taktflanke (steigend oder fallend, je nach Auslegung) ändern. Es handelt sich hier um eine Zwei-Speicher-Schaltungen. Mit Flip-Flops werden am häufigsten Register und Zähler aufgebaut.

Aufgaben

7. Setzen Sie in einem neuen Graphik-Design-File (d_ff.gdf) aus 2 D-Latches ein Master-Slave D-Flip-Flop zusammen. Generieren Sie davon für die weitere Verwendung ein Symbol.
8. Bauen Sie in einem neuen Graphik-Design-File (t_ff.gdf) das D-Flip-Flop zu einer 1-Bit Zählstufe, einem T-Flip-Flop aus. Das T-Flip-Flop ändert bei jedem Taktimpuls seinen Zustand. Dies wird durch die Rückführung des invertierten Ausgangssignals (Q) auf den D-Eingang erreicht (die Rückführung muss nicht mehr über Pins erfolgen, direkt im Schema einzeichnen) ➨ **B**

   Takteingabe über die Taste KB und Zustandsanzeige (0, 1) auf Anzeige DA.

   T-Flip-Flop compilieren, in den PLD-programmieren und austesten. Schaltung mit Demonstration auf dem Lösungsblatt visieren lassen.
9. Auf welche Taktflanke (aktive Taktflanke, steigend oder fallend) ändert Ihr T-Flip-Flop den Zustand? ➨ **B**

3: Asynchron Zähler

Einführung

Asynchrone Zähler (Beuth S.311) sind die einfachste Art von Zählern. Sie werden meistens als Frequenzteiler eingesetzt. Ihr Vorteil ist die einfache Schaltung und die Möglichkeit eines minimalen Stromverbrauchs. In jeder batteriebetriebenen Armbanduhr geschieht die Teilung der 32kHz Quarzfrequenz auf den Sekundentakt mit einem asynchronen Zähler. Ihr Nachteil ist, dass sich die Ausgänge der einzelnen Stufen nicht synchron mit dem Takt ändern, von Stufe zu Stufe wird die Verzögerung immer grösser.

Aufgaben

10. Bauen Sie als neues Projekt einen 4-Bit Asynchronzähler aus 4 der im Teil 2 gebauten D-Flip-Flops auf. Der Zählerstand soll über einen 7-Segment Decoder (Versuch 2) auf der Anzeige DB angezeigt, das Taktsignal mit der Taste KB erzeugt werden. ➨ **B**

    Zähler compilieren, programmieren und austesten.

    Schema und funktionierende Schaltung auf dem Lösungsblatt visieren lassen
11. An Stelle der Taste KB den Schalter nSA1 zur Eingabe des Taktsignals verwenden. Zählt der Zähler damit immer noch bei jedem Ein-Ausschalten von SA1 um genau Eins weiter? Wenn nicht, versuchen Sie das Verhalten zu erklären.

4: Synchron Zähler

Einführung

Dieser 4. Teil des Versuches ist fakultativ. Sie können damit Bonuspunkte erhalten. Viel wichtiger sollte Ihnen jedoch sein, nochmals etwas Neues zu lernen.

Im Gegensatz zum Asynchronzähler werden in einem synchronen Zähler die Takteingänge aller Flip-Flops parallel durch das Taktsignal angesteuert (Beuth S.335). Dadurch ergibt sich als Vorteil, dass alle Ausgänge synchron, mit der gleichen (kleinen) Verzögerung auf die aktive Taktflanke, den Zustand ändern. Als Nachteil ergibt sich eine grösserer Schaltungsaufwand und ein höherer Stromverbrauch. Synchrone Zähler sind effektiv etwas degenerierte endliche synchrone Automaten (Finite State Machines, kurz FSM) und können deshalb mit den weit entwickelten Methoden zur deren Behandlung in allen erdenklichen speziellen Ausführungen realisiert werden.

Aufgaben

In dieser Aufgabe wollen wir einen synchronen 4-Bit Vorwärts – Rückwärtszähler (Aufwärts – Abwärts, Up - Down) mit D-Flip-Flops und dem im Versuch 3 entwickelten 4-Bit Addierer aufbauen.

Das Prinzip ist sehr einfach: Der Zählerstand wird in 4 D-Flip-Flops (4-Bit Register) gespeichert. Je nach Zählrichtung muss dazu Eins addiert (Up) oder subtrahiert (Down, –1 addiert) werden um den nächsten Zählerstand zu erhalten. Dieser wird mit der folgenden aktiven Taktflanke in die D-Flip-Flops eingelesen und wird damit zum neuen aktuellen Zählerstand.

In FSM Terminologie sind die D-Flip-Flops der Zustandsspeicher, der Addierer die kombinatorische Folge-Zustands-Logik (Next-State Logik).

12. Bauen Sie als neues Projekt einen 4-Bit Synchronzähler aus 4 D-Flip-Flops und dem 4-Bit Addierer aus dem Versuch 3 auf.

    D-Flip-Flops aus der Bibliothek prim einsetzen, nicht Ihr D-Flip-Flop aus Teil 2 des Versuchs (führt in dieser Anwendung zu Timing-Problemen). Set- (PRN) und Clear- (CLRN) Eingänge offen lassen (per default inaktiv 1).

    Als Taktsignal den langsamen Taktgenerator auf dem Board verwenden (Signalbezeichnung GCLK1), dazu den Clock – Select Jumper auf S (slow) stellen (Postion rechts).

    Die 4 Q Flip-Flop Ausgänge sind auf die 4 A Eingänge des Addierers, die 4 S Summenausgänges desselben zurück auf die 4 D-Eingänge der Flip-Flops zu verdrahten. Auf die 4 B Eingänge des Addierers muss je nach gewählter Zählrichtung +1 oder –1 (2er Komplement Darstellung) gegeben werden.

    Up-Down Steuersignal mit Taste A (KA) erzeugen, Zählerstand über 7-Segment Decoder (Versuch 2) auf der Anzeige DB angzeigen.

13. Compilieren, programmieren, austesten; Demonstration und Schaltung auf dem Lösungsblatt visieren lassen. ➨ **B**

Bevor Sie den PC herunterfahren und alles ausschalten unbedingt sämtliche Files und Directories des Versuches löschen sowie auch noch den Papierkorb leeren.