Aufgabe 7: Rechnereinheit

G. Kemnitz*, TU Clausthal, Institut für Informatik

19. Juli 2007

1 Aufgabenstellung

Entwerfen, simulieren und testen Sie eine Rechnereinheit für einen 4-Bit-Prozessor bestehend aus

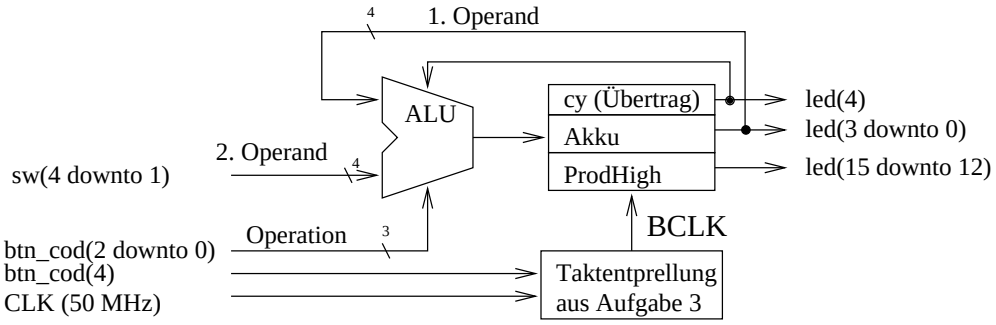
- 4-Bit-ALU (arithmetic logic unit)
- Akkumulator (4-Bit-Register für Operand b und Ergebnis)
- Zusatzregister für die oberen 4 Bits des Produktes bei Multiplikation
- Übertragsflag (cy).
- Handtakt mit Prellschutz wie in Aufgabe 3.

Für den Testaufbau soll gelten:

- Der Operand wird von den Schaltern sw(4 downto 1) geliefert.
- Operand b und das Ergebnis stehen im Akkumulator. Das obere Halbbyte des Registers ist in das Zusatzregister ProdHigh zu speichern.
- Die Operation wird durch den codierten Tasterwert btn_cod(2 downto 0) ausgewählt.
- Als Takt ist Signal btn_cod(4), dass wenn wenn mindestens eine Taste gedrückt ist 1, sonst 0) ist, zu entprellen und dann zu verwenden.
- Entprellung mit dem Automat aus Aufgabe 3.
- Akku-Inhalt, Übertragsflag und höherwertiger Produktteil sind auf Leuchtdioden auszugeben.

Die nachfolgende Abbildung zeigt die Gesamtschaltung:

*Tel. 05323/727116



Die ALU (arithmetic logic unit, Rechenwerk) soll in Abhängigkeit von der Nummer der gedrückten Taste folgende Funktionen ausführen:

BTN0: Lade Akku (Akku<=sw; cy und ProdHigh bleiben unverändert)

BTN1: Negation (Akku<= not Akku; cy und ProdHigh bleiben unverändert)

BTN2: UND (Akku<=Akku and sw; cy und ProdHigh bleiben unverändert)

BTN3: ODER (Akku<=Akku or sw; cy und ProdHigh bleiben unverändert)

BTN4: lade cy (cy<=sw(1) (Akku und ProdHigh bleiben unverändert)

BTN5: Addition (Akku<=Akku + sw + cy (ProdHigh bleibt unverändert)

BTN6: Subtraktion (Akku<=Akku - sw - cy (ProdHigh bleibt unverändert)

BTN7: Multiplikation (Akku mit den 4 niederwertigen und ProdHigh mit den 4 höherwertigen Bits des Produkts (Akku*sw) laden; cy bleibt unverändert)

Sonst: keine Operation (Akku, cy und ProdHigh bleiben unverändert)

2 Vorbereiten von Testbeispielen

Die Schaltung hat 7 Dateneingänge (sw(4 downto 1) und btn_cod(2 downto 0) und 9 Speicherzellen (Akku, ProdHigh und cy). Ein vollständiger Test mit allen Variationen der Eingabe und des internen Zustandes würde

$$AT = 2^{7+9} = 65536 \text{ Tests}$$

erfordern. Wir wollen uns auf eine Stichprobe von 32 Testschritten beschränken, wobei jede Operation drei bis vier mal mit unterschiedlichen Operanden ausgeführt werden soll. Bereiten Sie hierfür eine Tabelle mit nachfolgendem Aufbau vor:

Schritt	Operation	Taster	btn_cod	sw_n	cy_n	$Akku_n$	$ProdHigh_n$
1	lade Akku	BTN0	10000	0111	u	uuuu	uuuu
2	lade cy	BTN4	10100	xxx0	u	0111	uuuu
3	AND	BTN2		0110	0	0111	uuuu
4	keine Op.	BTNA					
5	Negation	BTN1					
6	Addition						
...	...						

Alle Testeingaben und Sollausgaben ausfüllen. Jeder Kursteilnehmer soll seinen eigenen Testsatz aufstellen.

3 Projekt vorbereiten

Legen Sie ein neues Verzeichnis

```
H:\TGP\Aufgabe6
```

an und kopieren Sie aus dem Netz Aufgabe6.npl, Aufgabe6.vhd, Taktentprellung.vhd, Rechnereinheit.vhd und Praktikum.ucf in das Verzeichnis und öffnen Sie das Projekt.

4 Oberste Beschreibungsebene

Die Datei Aufgabe6.vhd beschreibt die Verbindungen zwischen Rechnereinheit, Tastenentprellung, externen Schaltkreisanschlüssen und einem Zähler, der die Takte/Testschritte zählt. Zeichnen Sie die Schaltung, wie sie programmiert ist, auf Papier.

5 Taktentprellung

In das VHDL-Modul Taktentprellung.vhd können Sie weitgehend die Funktionsbeschreibung aus Aufgabe 3 übernehmen. Lediglich einige Signalnamen sind an die geänderte Schnittstellenbeschreibung im entity-Teil anzupassen. Zum Testen dient der Zähler in der obersten Beschreibungsebene. Die Entprellung ist in Ordnung, wenn der Zähler bei jeder Tasterbetätigung exakt einen Schritt weiterzählt.

6 Entwurf der Rechnereinheit

Die Funktion der ALU soll in einer eigenen VHDL-Datei gemeinsam mit den Registern in einem einzigen Prozess mit dem Handtakt in der Sensitivity-Liste beschrieben werden. Folgendes ist zu beachten:

Bitanzahl der Ergebnisse arithmetischer Operationen Summe, Differenz (incl. Übertrag) und Produkt sind größer als der Akku. Sie müssen zuerst einem ausreichend großen Bitvektor zugewiesen werden, bevor sie auf die einzelnen Ergebnisbestandteile aufgeteit werden können.

Signal oder Variable als Zwischengrößen? Eine interne Zwischengröße kann ein zwischen ARCHITECTURE und BEGIN vereinbartes Signal oder eine zwischen PROCESS und BEGIN vereinbarte Variable sein. Im folgenden Beispiel ist es eine Variable:

```
process(BCLK)
  variable Summe: std_logic_vector(4 downto 0);
begin
  if BCLK'event and BCLK='1' then
    ...
     Summe:=sw + Akku + cy;
     Akku<=Summe(3 downto 0);
     cy<=Summe(4);
    ...
  end if;
end process;
...
```

Die Variable Summe hat in der nachfolgenden Zeile ihren zugewiesen Wert. Ein Signal an dieser Stelle hätte den neuen Wert erst, wenn die Taste das nächste mal gedrückt wird. Wie würde sich die Funktion der Schaltung ändern. Probieren Sie es in Zweifelsfällen aus.

Signale, die als Schaltungsausgänge definiert sind, dürfen nicht auf der rechten Seite einer Zuweisung stehen: Der Akku- und der cy-Inhalt dient als Eingabe für die ALU und benötigen ein entsprechendes interne(s) Signal/Variable.

Kombinatorische Ausgabe: Der Inhalt von Akku, ProdHigh und cy muss, damit man etwas sieht, Leuchtdioden zugewiesen werden. Das soll natürlich nicht mit einem Takt Verzögerung erfolgen. Müssen diese Zuweisungen im Prozess oder außerhalb des Prozesses stehen, oder ist es egal, wo sie stehen? (Die Antwort hängt davon ab, ob Sie Signale oder Variablen verwenden.)

7 Simulation der Rechnereinheit

Wenn Sie die Datei "Rechnereinheit.vhd" fertig geändert haben und testen wollen, gehen Sie in folgenden Schritten vor:

1. Syntaxtest (Speichern, Auswahl "Rechnereinheit" im Source-Fenster und "Check Syntax" im "Tool-Fenster".

2. Testbench-Waveform erstellen: (Auswahl "Recheneinheit" im Source-Fenster und "Create New Source" im "Tool-Fenster"; Typ Testbench Waveform, Name beliebig; Taktperiode etc. Standardvorgaben lassen; als Eingaben die 32 Eingabewerte des selbst aufgestellten Testsatzes eingeben)

3. Simulation: (Simulate Behavioral Model; Vergleich der Simulationsergebnisse mit den zuvor berechneten Sollwerten; wenn Fehler in der eigenen Schaltungsbeschreibung gefunden werden: Simulator beenden, Fehler beseitigen, Syntaxtest, Simulator neu starten).

8 Test des Gesamtsystems

Wenn die beiden Komponente "Entprellung" den separaten Test und die Komponente "Recheneinheit" die Simulation bestanden hat, sollte das Gesamtsystem bis auf wenige Syntax-Fehler funktionieren. Arbeiten Sie nach dem "Download" den gesamten Testsatz ab.

9 Zusatzaufgabe

Erweitern Sie den Befehlssatz z.B. um:

BTN9: Tausche Akku und ProdHigh (Akku<=ProdHigh; ProdHigh<=Akku; cy bleibt unverändert)

BTNA: Rotation rechts durch cy (Akku<= cy & Akku(3 downto 1); cy<=Akku(0); ProdHigh bleibt unverändert)

BTNB: Rotation links durch cy ...

oder selbst definierte Befehle.

10 Aufräumen

- Über Menüpunkt "Project, Cleanup Project Files" automatisch generierte Design-Files löschen.
- Netzteil zur Spannungsversorgung aus der Steckdose ziehen.
- Modelsim und Projektnavigator beenden.