Protokoll VHDL – Fritz 1130977 Umschaden 1010701

Aufgabe 1 ALU

Aufgabenstellung:

Entwerfen Sie eine einfache arithmetisch-logische Einheit (arithmetic logic unit, ALU) die zwei 2-bit-Zahlen addiert, multipliziert und logisch verknüpft (AND, OR) in der Xilinx ISE Umgebung. Das Ergebnis soll binär auf die LEDs LD0-LD3 und auf die 7-Segmentanzeige ausgegeben werden. Beachten Sie die Negativlogik und die Codierung der 7-Segmentanzeige. Die Schalter SW4-SW7 sollen für in1 und in2, SW2 und SW3 für die Umschaltung zwischen Addieren, Multiplizieren und den logischen Verknüpfungen verwendet werden (alumode). Die Implementierung soll zuerst mit Modelsim simuliert und verifiziert werden. Danach soll die Implementierung auf das FPGA Board übertragen und getestet werden.

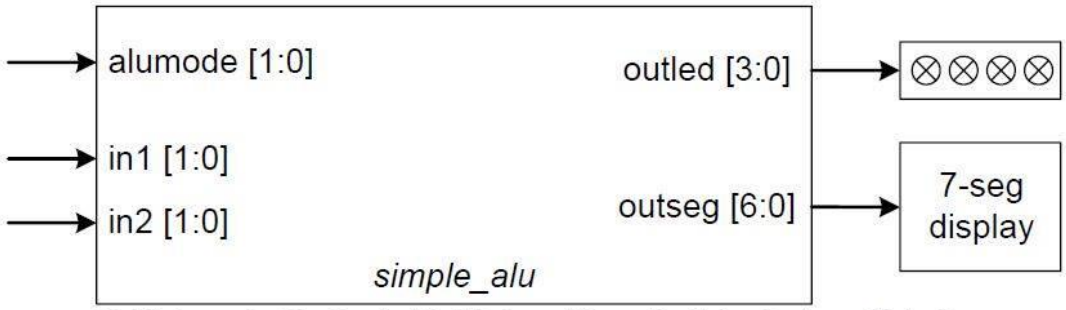


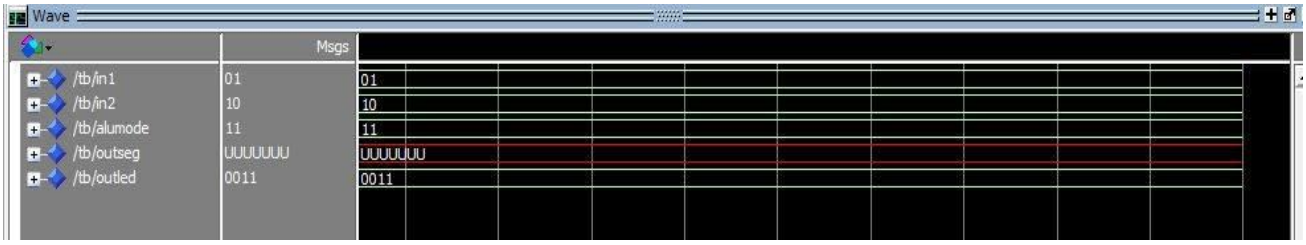
Abbildung 1: Blockschaltbild der arithmetisch-logischen Einheit

Code:

```
if alumode = ("00") then
   result_internal := (in1_internal + in2_internal);  --add

elsif alumode = ("01") then
   result_internal := (in1_internal * in2_internal);  --mult

elsif alumode = ("10") then
    result_internal :=  conv_integer(in1 and in2);  --and
elsif alumode = ("11") then
    result_internal := conv_integer(in1 or in2);  --or
end if;

outled <= std_logic_vector(to_unsigned(result_internal, 4));
```

In dieser Case Anweisung werden die beiden Eingänge in1 und in2 miteinander verknüpft. Das Ergebnis wird in der Variable result_internal gespeichert und über das outled ausgegeben.

Im nächsten Bild kann man die Testbench mit der OR-Verknüpfung sehen.



Code:

```
if result_internal = 9 then
    outseg <= "0000100";
       elsif result_internal = 8 then
    outseg <= "0000000";
       elsif result_internal = 7 then
    outseg <= "0001110";
       elsif result_internal = 6 then
    outseg <= "0100000";
       elsif result_internal = 5 then
    outseg <= "0100100";
       elsif result_internal = 4 then
    outseg <= "1101100";
       elsif result_internal = 3 then
    outseg <= "0000110";
       elsif result_internal = 2 then
    outseg <= "0010010";
       elsif result_internal = 1 then
    outseg <= "1001111";
       elsif result_internal = 0 then
    outseg <= "0000001";
end if;
```

Je nach Resultat wird auf der Anzeige nun die Ziffer in Dezimal dargestellt.

Aufgabe 2 Moore Automat

Aufgabenstellung:

Ein Moore-Automat ist ein endlicher Automat, dessen Ausgang nur von seinem aktuellen Zustand abhängt (vgl. Mealy-Automat). Ein Anwendungsbeispiel von Moore-Automaten ist z.B. im Bereich der Protokollüberprüfung bei seriellen und parallelen Datenübertragungssystemen. Es soll nun ein Schaltwerk zur Impulsfolgenerkennung erstellt werden. Der zu entwerfende Moore-Automat soll drei aufeinander folgende 2-Bit Kombinationen eines Eingangsvektors input in der Reihenfolge (01, 11, 10) erkennen und mit dem Ausgangssignal output=1 anzeigen. Der High-Pegel des Ausgangssignals output soll für eine Taktperiode verfügbar sein. Die zu synthetisierende Hardware des Automaten soll durch einen asynchronen RESET-Eingang in den Anfangszustand S0 versetzt werden. Für die Zustände ist ein Aufzählungstyp zu wählen. Es soll zuerst ein Zustandsdiagramm erstellt werden, bevor mit dem Codieren begonnen wird. Die Impulsfolgenerkennung ist mit drei Prozessen zu realisieren und nach der Simulation mit Modelsim auf die vorhandene Hardware zu bringen. Als Clock clk soll ein manuell mit dem Schalter BTN0 (M13) vorgegebener Takt dienen. Für den Ausgang output soll die LED LD0 (K12) und für die Eingänge, Enable und Reset SW4 – SW7 dienen.

Abbildung 2: Blockschaltbild Impulsfolgenerkennung