Übung 6

In den vorangegangenen Übungen wurde die ALU des Prozessors entworfen. In dieser Übung soll das Registerfile in VHDL spezifiziert, validiert und synthetisiert werden.

Registerfile

Der Prozessor soll mit 8 Universalregistern ausgestattet werden, die als Quell- und Zieloperanden für ALU-Befehle als auch Operanden für Sprung-, Unterprogramm- und Datentransportbefehle speichern können. Da der Befehlssatz maximal zwei Quelloperanden und einen Zieloperanden vorsieht, werden die Register wie in Abbildung 14 dargestellt zu einem Registerfile zusammengefasst. Dieses liefert zwei 16-Bit Worte als Ausgabe und akzeptiert ein 16-Bit Wort als Eingabe.

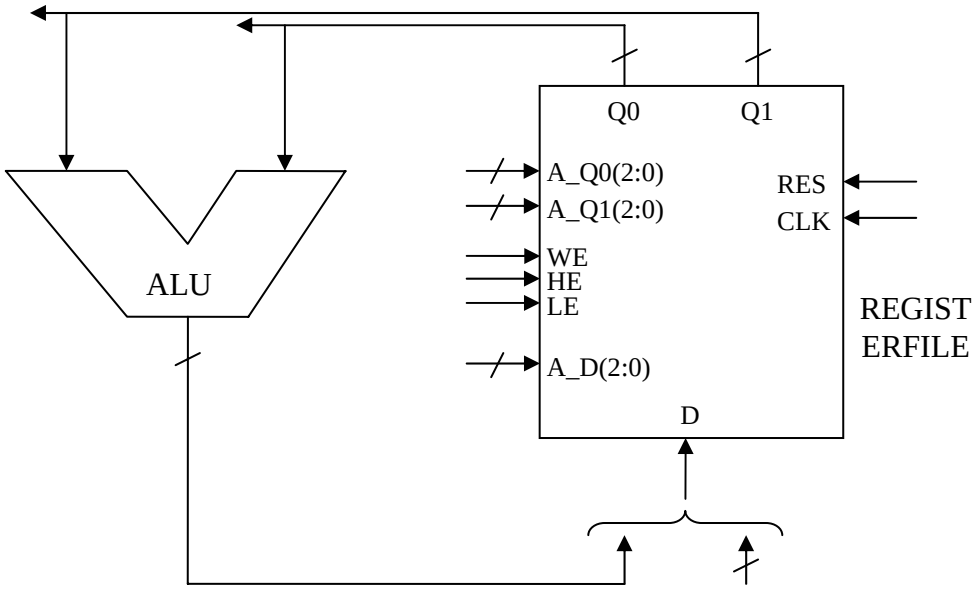

Abbildung 14: Registerfile mit Anbindung an die ALU

Port	***Richtung***	***Bedeutung***
A_D(2:0)	IN	Adresse für den Dateneingang D
D(15:0)	IN	Dateneingang des Registerfiles
WE	IN	Write Enable. Bei WE='1' werden die Daten am Eingang D in das Register mit Adresse A_D geschrieben (HE und LE geben dabei den Schreibmodus an). Der Schreibvorgang erfolgt bei steigender Taktflanke an CLK.
HE	IN	High Enable. Bei HE='1' werden beim Schreibvorgang (WE='1') die niederwertigen Bits am Dateneingang als oberes Halbwort ins Register mit Adresse A_D geschrieben. $R_{A\_D}(15:8)<=D(7:0)$.
LE	IN	Low Enable. Bei LE='1' werden die niederwertigen Bits am Dateneingang als unteres Halbwort ins Registerfile geschrieben. $R_{A\_D}(7:0)<=D(7:0)$ (vgl. HE)

Port	*Richtung*	*Bedeutung*
A_Q0(2:0) A_Q1(2:0)	IN	Adressen der Register für die Ausgangsdaten Q0 und Q1.
Q0(15:0) Q1(15:0)	OUT	Datenausgänge, Ausgabe erfolgt ungetaktet Q0<=R_{A_Q0}, Q1<=R_{A_Q1}.
RES	IN	Reset. Bei RES='1' werden alle Register auf Null gesetzt.
CLK	IN	Taktsignal

Tabelle 7: Portbeschreibung des Registerfiles

- Spezifizieren Sie das Registerfile in VHDL (*regfile.vhd*). Der Prozessor soll vollständig synchron entworfen werden, d.h. alle Speicherelemente sollen mit dem gleichen Takt betrieben werden und auf die positive Taktflanke triggern.
- Der Inhalt des Registerfiles lässt sich gut mit Hilfe eines Signal-Feldes (`array (0 to7) of STD_LOGIC_VECTOR(15 downto 0)`) repräsentieren.
- Um flankengesteuertes Verhalten zu modellieren, wird in VHDL das Attribut **event** für Signale bereitgestellt. Alternativ kann auch die Funktionen `rising_edge` aus dem Package `IEEE.std_logic_1164` verwendet werden.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use …
      …
entity REG_FILE is
      port(
            …);
end REG_FILE;

architecture BEHAVIORAL of REG_FILE is
      type regs is array(0 to 7) of STD_LOGIC_VECTOR(15 downto 0);
      signal contents: regs;
begin
      …
end BEHAVIORAL;
```

Synthese und Validierung des Registerfiles

Testen Sie das Registerfile mit dem Waveform-Simulator. Schreiben Sie eine Testbench *reg_file_tb.vhd*, welche die korrekte Funktion des Registerfiles bestätigt. Instanzieren sie dazu eine Komponente reg_file und belegen sie die Ports mit ausgewählten Testmustern. Benutzen Sie die Anweisung **`assert`**, um das gewünschte Verhalten während der Simulation zu überprüfen. Das File besitzt in etwa folgende Struktur:

```
Library IEEE;
use ...

entity reg_file_tb is
end reg_file_tb;

architecture reg_file_tb of reg_file_tb is
      component REG_FILE is
            port(
                  ...
                  );
      end component REG_FILE;
```

```
	--signals and constants
	signal		l_WE, l_HE, l_LE, l_RES, l_CLK: STD_LOGIC;
	signal		l_A_D, l_A_Q0, l_A_Q1: STD_LOGIC_VECTOR(2 downto 0);
	signal		l_D, l_Q0, l_Q1: STD_LOGIC_VECTOR(15 downto 0);
	constant period: time:=10ns;

begin
	clockgen:process is
	begin
		l_clk<='0';
		wait for period/2;
		l_clk<='1';
		wait for period/2;
	end process;

	test_component: reg_file
	port map(...);

	test_reg_file: process is
	begin
		-- test reset function
		l_RES <= '1';		-- general reset
		l_WE <= '0';
		l_HE <= '0';
		l_LE <= '0';
		wait for period;

		l_A_Q0 <= "000";	-- reg0, reg7 -> data output
		l_A_Q1 <= "111";

		l_RES <= '0';		-- operation modus
		wait for period;

		assert l_Q0 = "0000000000000000"	-- check data output
			report "RES error"
			severity failure;
		assert l_Q1 = "0000000000000000"
			report "RES error"
			severity failure;

		-- test write function
		...
		report "test passed";
		wait;
	end process test_reg_file;

end architecture reg_file_tb;
```

Synthetisieren Sie das Modul mit Synplify-Pro *nach Absprache mit dem Übungsleiter*. Kontrollieren Sie den Synthese-Report (Log-File) und suchen Sie den Abschnitt „Resource Usage“ am Ende des Reports. Versuchen Sie durch Optimieren Ihres Quellcodes möglichst wenige logische Zellen des FPGAs für das Registerfile-Modul zu verwenden.