Allgemeine Hinweise zur Klausur

Prüfungsdauer: 120 Minuten Aufgaben: 5 Punktezahl: 100 von 116

Erlaubte Hilfsmittel: beliebiges gedrucktes oder handgeschriebenes Material

Aufgabe 1 - Datenspeicher (20 Punkte)

Ein Signalprozessor soll mit einem Speicherbereich ausgestattet werden, der als Dual-Ported RAM ausgeführt ist. Dieser Speicher soll den Zugriff auf 2 Ports zulassen, die unabhängig von einander zum Lesen bzw. zum Schreiben durch den Prozessor und einen Koprozessor verwendet werden können. Der Speicher soll eine Größe von 4k Worten mit einer Wortbreite von jeweils 16 Bit zur Verfügung stehen.

Frage 1.1 (4 Punkte): Skizzieren Sie ein Blockschaltbild des Speichers mit seinen Eingängen und Ausgängen. Hinweis: Adressleitungen, Datenleitungen und Steuersignale.

Frage 1.2 (4 Punkte): Konzeption und Umsetzung. Beschreiben Sie den wesentlichen Teil der Reali-sierung des Speichers und dessen Umsetzung in HDL.

Frage 1.3 (6 Punkte): Skizzieren Sie eine mögliche Realisierung des Speichers in HDL. Erläutern Sie Ihre gewählte Realisierung (z.B. gegenüber anderen Optionen).

Frage 1.4 (6 Punkte): Was geschieht, wenn beide Ports von der gleichen Adresse lesen möchten? Was geschieht, wenn ein Port eine Adresse beschreiben möchte, die auf dem anderen Port gerade ausgelesen wird? Was geschieht, wenn beide Ports auf die gleiche Adresse schreiben möchten? Begründen Sie Ihre Aussage. Mit welchen Massnahmen lassen sich Konflikte gege-benenfalls verhindern?

Aufgabe 2 - Signalverarbeitung (18 Punkte)

Zur Signalverarbeitung wird folgender Ausschnitt aus einem Rechenwerk eingesetzt.

entity Filter is

port ( CLK, CLR : in std\_logic;

ENR, ENO : in std\_logic;

IN, H : in std\_logic\_vector (15 downto 0);

OUT : out std\_logic\_vector (15 downto 0));

end Filter;

architecture RTL of Filter is

signal Q : signed (31 downto 0);

begin

process (CLK)

begin

if rising\_edge(CLK) and (ENO = '1') then

OUT <= std\_logic\_vector(Q(31 downto 16));

end if;

end process;

process (CLK)

begin

if rising\_edge(CLK) then

if (CLR = '1') then Q <= (others => '0');

elsif (ENR = '1') then

Q <= Q + signed(H) \* signed(IN);

end if;

end if;

end process;

end RTL;

Frage 2.1 (4 Punkte): Skizzieren Sie das Blockschaltbild mit allen Eingängen und Ausgängen.

Frage 2.2 (6 Punkte): Rekonstruieren Sie das Innenleben (Blockschaltbild) des Rechenwerks mit den beteiligten Prozessen als Skizze. Hinweis: Verwenden Sie die Prozesse zusammen mit den zugehörigen Signalen. Rekonstruieren Sie die Struktur des Rechenwerks. Welcher Algorithmus wird hiermit ausgeführt?

Frage 2.3 (8 Punkte): Tests. Skizzieren Sie die Struktur von Testfällen in einem Testprogramm für den Algorithmus. In dem Algorithmus werden 16-Bit Festkommazahlen addiert und multipliziert. Hierbei sei angenommen, dass |Σ H(i) | < 1. Wie können Überläufe im Rechenwerk vermieden werden? Wie können Sie auf Überläufe testen?

Aufgabe 3 - Steuerwerk eines Prozessors (30 Punkte)

Folgender Programmtext zeigt einen Ausschnitt aus dem Zustandsautomaten eines Prozessors:

update\_state\_register: process(Clk, RS)

begin

if (RS='1') then

current\_state <= Z1;

elsif rising\_edge(Clk) then

current\_state <= next\_state;

end if;

end process update\_state\_register;

logic\_next\_state\_and\_actions: process(current\_state, T\_OPCODE)

begin

case current\_state is

when Z1 =>

case T\_OPCODE is

when NOP => ...;

when ADD | SUB | ANDA | EOR | ORA | LDA => ...;

next\_state<=Z2;

when LSL | LSR | LDI => ...;

when STR => ...;

when JMP => ....; next\_state<=Z2;

when PUSH => ...;

when POP => ...; next\_state<=Z2;

when CALL => ...; next\_state<=Z2;

when RET => ...; next\_state<=Z2;

when BRBC =>

case T\_PSR is

when "10000" => if (T\_Zero = '0') then

...; next\_state<=Z2;

elsif (T\_Zero ='1') then

...;

end if;

...;

end case;

...;

when others => null;

end case; -- T\_OPCODE

when Z2 =>

case T\_OPCODE is

when ADD | SUB | ANDA | EOR | ORA | LDA => ...;

next\_state<=Z1;

when BRBS | BRBC| JMP | CALL => ...;

next\_state<=Z1;

when RET => ...; next\_state<=Z1;

when POP =>...; next\_state<=Z1;

when others => null;

end case; -- T\_OPCODE

when others => null;

end case; -- states

end process logic\_next\_state\_and\_actions;

Frage 3.1 (6 Punkte): Zeichnen Sie ein Blockschaltbild des Automaten ausgehend von der gegebenen HDL Realisierung. Welche Signale liegen an den Eingängen? Welche Signale liegen voraussichtlich an den Ausgängen? Wie viele Leitungen werden für die Kodierung der Zustände benötigt? Wie viele Eingangsleitungen werden für einen RISC-Prozessor mit 32 Befehlen benötigt?

Frage 3.2 (4 Punkte): Zeichnen Sie das Zustandsdiagramm des Automaten. Hinweis: Verallgemeinern Sie die Bedingungen für die Zustandsübergänge, hierzu folgen noch weitere Fragen.

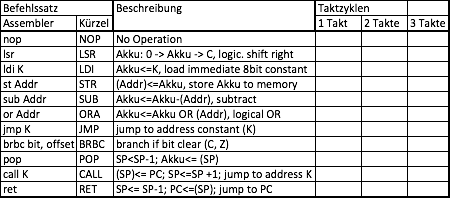
Frage 3.3 (4 Punkte): Welcher der im HDL-Text aufgeführten Prozesse ist taktabhängig? Welcher der Prozesse ist nicht taktabhängig? Wann genau erfolgen die im Zustandsdiagramm gezeigten Zustandsübergänge? Wann genau erfolgen die (im Programmauszug nicht näher genannten) Zustandsaktionen in Bezug auf das Zustandsdiagramm?

Frage 3.4 (6 Punkte): Im eingangs gezeigten Zustandsautomaten sind die Zustandsaktionen eine un-mittelbare Folge der Eingangssignale sowie des aktuellen Zustands. Folgendes Blockschaltbild zeigt eine alternative Realisierung.



Wie wäre der eingangs gezeigte HDL-Code für diesen Automaten zu ändern? Skizzieren Sie die Struktur des HDL-Codes (grobe Struktur genügt). Hätte diese Variante Nachteile bzgl. des Zeitverhaltens? Hätte diese Variante Vorteile? Halten Sie diese Variante für diesen Zweck überhaupt für praktikabel? Hinweis: Begründen Sie Ihre Aussagen.

Frage 3.5 (4 Punkte): Folgende Tabelle zeigt einen Ausschnitt der Befehle des Prozessors. Welche Befehle benötigen gemäß der eingangs gegebenen Realisierung (HDL) 1, 2 oder 3 Taktzyklen? Hinweis: Verwenden Sie für Ihre Antworten direkt das Aufgabenblatt.



Frage 3.6 (6 Punkte): Erläutern Sie, warum die in Frage 3.5 in der Tabelle aufgeführten Befehle einen, zwei oder drei Takte benötigen. Warum ist die Behandlung der bedingten Sprungbefehle im eingangs gezeigten HDL-Text im Vergleich zum unbedingten Springbefehl so aufwändig? Welche Probleme gibt es in dieser Realisierung mit dem Befehl RET?

Aufgabe 4 - Architektur des Prozessors (24 Punkte)

Folgende Abbildung zeigt den Aufbau eines Mikrocontrollers.



Frage 4.1 (6 Punkte): Erläutern Sie den Ablauf folgender Befehle: (1) NOP, (2) CALL K (Aufruf Unterprogramm), (3) st Addr (Wert an Adresse Adr speichern), (4) AND Addr (mit Wert aus Adresse Addr verknüpfen), (5) LDI K (Konstante laden), (6) BRBC bit, offset (bedingter Sprungbefehl).

Frage 4.2 (6 Punkte): Pipeline und Sprungbefehle. Folgende Abbildung zeigt die Kette vom Programmzähler (Befehlszähler) bis zum Befehlsregister (Dekoder).



Erläutern Sie den normalen Programmablauf: Wann zählt der Programmzähler weiter? Wie kommt es, dass auf Zeitdiagrammen der Programmzähler dem aktuell dekodierten und vom Steuerwerk verarbeiteten Befehl immer 2 Takte vorauseilt? Sprungbefehle: Was genau geschieht bei einem bedingten bzw. unbedingten Sprungbefehl? Was muss wegen der dargestellten Kaskade (Pipeline) bei Sprungbefehlen beachtet werden?

Frage 4.3 (6 Punkte): Stapel. Mit dem Befehl PUSH kann der Inhalt des Prozessors im Stapel konserviert werden. Mit dem Befehl POP holt man diesen Inhalt wieder in den Prozessor. Erläutern Sie den Zweck dieser Befehle und die diesbzgl. Abläufe bzgl. Programmzähler (PC) und Stapelzeiger (SP). Beachten Sie hierbei die in der Abbildung zu Frage 2.2 gezeigte Kette (Pipeline).

Frage 4.4 (6 Punkte): Statusbits. Das Rechenwerk enthält am Ausgang der ALU ein sogenannten Prozessor-Status-Register (PSR, siehe Abbildung zur Architektur eingangs zur Frage). Erläutern Sie die Funktionsweise und den Zweck dieses Registers. Wie wird dieses Register z.B. bei Sprungbefehlen verwendet? Welche Aktionen leitet das Steuerwerk aus diesem Register ab? Wann genau steht dem Steuerwerk diese Information bei der Ausführung eines Befehls zur Verfügung?

Aufgabe 5 - Rechenwerk des Prozessors (24 Punkte)

Folgender HDL-Text zeigt einen Ausschnitt aus dem Rechenwerk des Prozessors.

process (OPCODE, K, DS\_DO, Akku)

variable RsVar, RdVar, RdoutVar : D\_Type;

variable RVar : signed(D\_Width downto 0); -- 9 bits wide

begin

RsVar := DS\_DO; -- copy Rs from data memory

RdVar := Akku; -- copy Rd from Akku

RdoutVar := Zero; -- avoids latches in synthesis

case OPCODE is

when LSL => CFlag <= RdVar(D\_Width-1); -- msb

RdOutVar(D\_Width-1 downto 1) := RdVar(D\_Width-2 downto 0);

RdOutVar(0):='0'; -- lsb

when LSR => CFlag <= Akku(0); -- lsb

RdOutVar(D\_Width-2 downto 0) := Akku(D\_Width-1 downto 1);

RdOutVar(D\_Width-1):='0'; -- msb

when ADD => RVar := signed('0' & RdVar) + signed('0' & RsVar);

RdOutVar := std\_logic\_vector(RVar(D\_Width-1 downto 0));

NFlag <= RVar(D\_Width);

when SUB => RVar := signed('0' & RdVar) - signed('0' & RsVar);

RdOutVar := std\_logic\_vector(RVar(D\_Width-1 downto 0));

NFlag <= RVar(D\_Width);

when ANDA=> RdOutVar := RdVar AND RsVar;

when EOR => RdOutVar := RdVar XOR RsVar;

when ORA => RdOutVar := RdVar OR RsVar;

when STR | PUSH => DS\_DI <= Akku;

when LDA | POP => RdOutVar := DS\_DO;

when LDI => RdOutVar := K;

when others => null; -- no action

end case;

if (RdOutVar = Zero) then ZFlag <='1'; end if;

RdOut <= RdOutVar;

end process;

load\_akku : process (Clk) begin

if (ACC\_EN = '1') then

if rising\_edge(Clk) then

Akku <= RdOut;

end if;

end if;

end process load\_akku;

end RTL;

Frage 5.1 (6 Punkte): Der HDL-Text lässt sich als Schaltnetz in Kombination mit einem Register inter-pretieren. Skizzieren Sie ein Blockschaltbild mit Eingängen und Ausgängen.

Frage 5.2 (6 Punkte): Skizzieren Sie ein Zeitdiagramm für den folgenden Ablauf mit den Signalen aus dem Blockschaltbild. Hinweis: Verwenden Sie die Vorlage auf dem Aufgabenblatt.

start: ldi 4

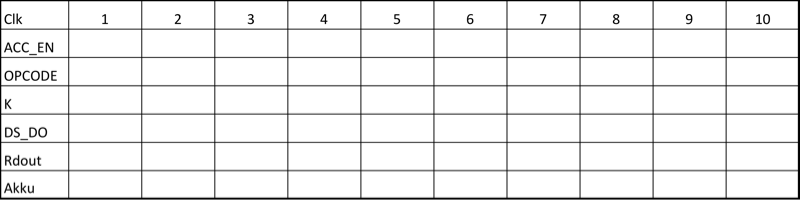
st $0

ldi 2

lsl

add $0

end: st $1



Frage 5.3 (6 Punkte): Weshalb ist der Akku als Register ausgeführt (d.h. taktabhängig)? Könnte man den Akku auch als Schaltnetz ausführen um hierdurch das Rechenwerk zu beschleunigen? Begründen Sie Ihre Aussage.

Frage 5.4 (6 Punkte): Durch welche Komponente des Prozessors wird das ACC-EN Signal gesteuert? Woher weiss diese Komponente, welchen Zustand dieses Signal annehmen soll? Begründen Sie Ihre Aussage und nennen Sie ggf. ein Beispiel.