Allgemeine Hinweise zur Klausur

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

Erlaubte Hilfsmittel: beliebiges gedrucktes oder handgeschriebenes Material

Aufgabe 1 - Steuerwerk (34 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 LSL | LSR | LDI => ...;

when STR => ...;

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

next\_state<=Z2;

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

next\_state<=Z1;

when RET => ...; next\_state<=Z3;

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

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

next\_state<=Z1;

when others => null;

end case; -- T\_OPCODE

when Z3 =>

case T\_OPCODE is

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

when others => null;

end case; -- T\_OPCODE

when others => null;

end case; -- states

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

Frage 1.1 (6 Punkte): Zeichnen Sie ein Blockschaltbild des Automaten. 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 1.2 (4 Punkte): Zeichnen Sie das Zustandsdiagramm des Automaten. Hinweis: Verallgemeinern Sie die Bedingungen für die Zustandsübergänge, hierzu folgen noch weitere Fragen.

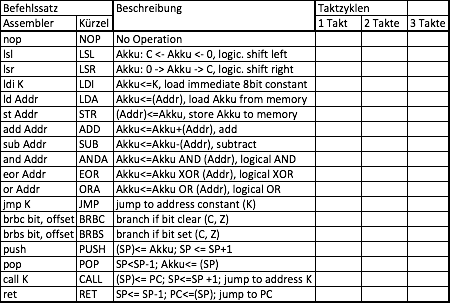
Frage 1.3 (6 Punkte): Welcher der im HDL-Text aufgeführten Prozesse ist taktabhängig? Welcher der Prozesse ist nicht taktabhängig? Wann genau erfolgen die im Zustandsdiagramm gezeigten Zustandsübergänge? Wann genau erfolgen die (im Programmauszug nicht näher genannten) Zustandsaktionen in Bezug auf das Zustandsdiagramm?

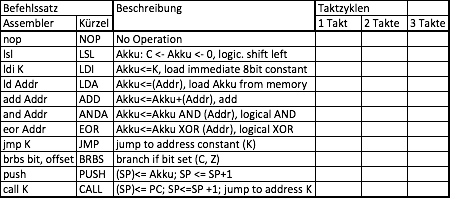
Frage 1.4 (6 Punkte): Im eingangs gezeigten Zustandsautomaten sind die Zustandsaktionen eine un-mittelbare Folge der Eingangssignale sowie des aktuellen Zustands. Folgendes Blockschaltbild zeigt eine alternative Realisierung.

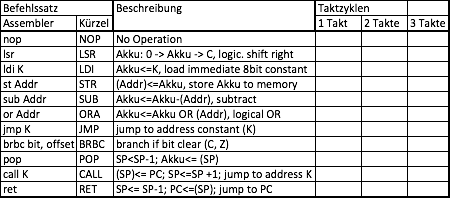


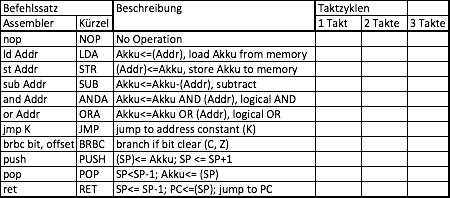
Wie wäre der eingangs gezeigte HDL-Code für diesen Automaten zu ändern? Skizzieren Sie die Struktur des HDL-Codes (grobe Struktur genügt). Hätte diese Variante Nachteile bzgl. des Zeitverhaltens? Hätte diese Variante Vorteile? Halten Sie diese Variante für diesen Zweck überhaupt für praktikabel? Hinweis: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

Frage 1.5 (6 Punkte): Folgende Tabelle zeigt die Befehle des Prozessors. Welche Befehle benötigen 1, 2 oder 3 Taktzyklen? Hinweis: Verwenden Sie für Ihre Antworten direkt das Aufgabenblatt.









Frage 1.6 (6 Punkte): Frage zu speziellen Befehlen. Erläutern Sie, warum die in Frage 1.5 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?

Aufgabe 2 - Prozessorarchitektur (24 Punkte)

Folgende Abbildung zeigt den Aufbau eines Mikrocontrollers.



Frage 2.1 (6 Punkte): Erläutern Sie den Ablauf folgender Befehle: (1) NOP, (2) LSL (Links schieben), (3) LDI K (Konstante laden), (4) ADD Addr (Wert aus Adresse Addr addieren), (5) jmp K (unbedingter Sprungbefehl), (6) BRBS bit, offset (bedingter Sprungbefehl).

Frage 2.1 (6 Punkte - ALT2.1): 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 2.2 (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 2.3 (6 Punkte): Unterprogramme. Mit dem Befehl CALL K kann in ein Unterprogramm verzweigt werden. Aus dem Unterprogramm kehrt man mit dem Befehl RET wieder ins Hauptprogramm zurück. Erläutern Sie die diesbzgl. Abläufe bzgl. Programmzähler (PC) und Stapelzeiger (SP). Beachten Sie hierbei die in der Abbildung zu Frage 2.2 gezeigte Kette (Pipeline).

Frage 2.3 (6 Punkte \_ALT 2.3): 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 2.4 (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 3 - Rechenwerk (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 3.1 (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 3.2 (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 5

st $0

ldi 3

add $0

end: st $1

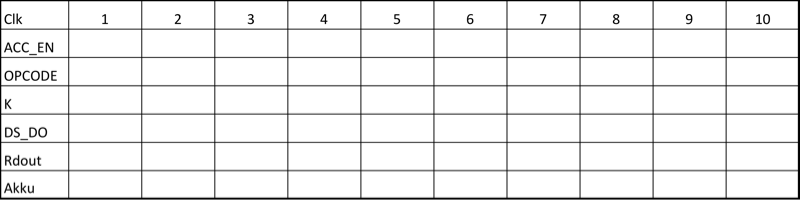
start: ldi 1

st $0

lsl

eor $0

end: st $1



Frage 3.3 (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.

Punkte - ALT 3.3): Wozu dient die Rückkopplung im Blockdiagramm (Signal Akku)? Könnte man das Rechenwerk dadurch beschleunigen, dass man auf diese Rückkopplung ver-zichtet und den Akku ohne Takt ebenfalls als Schaltnetz ausführt? Begründen Sie Ihre Aus-sage.

Frage 3.5 (6 Punkte): Welchen Zweck hat das ACC\_EN Signal? Welcher Programmablauf ergäbe sich, wenn man auf dieses Signal verzichtet und der Akku als Register ständig aktiv wäre? Begründen Sie Ihre Aussage.

Frage 3.6 (6 Punkte - ALT 3.4): 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.

Aufgabe 4 - Signalverarbeitung (24 Punkte)

Zur Signalverarbeitung wird folgender Ausschnitt aus einem Rechenwerk eingesetzt.

process (Clk) begin

if rising\_edge(Clk) then

for i in 0 to 31 loop

Q(i) <= Q(i+1) + P(i);

end loop;

end if;

end process;

process (S, H) begin

for i in 0 to 31 loop

P(i) <= S \* H(i);

end loop;

end process;

process (Clk) begin

if rising\_edge(Clk) then

S <= signed(IN(15 downto 0));

end if;

end process;

OUT <= Q(0)(31 downto 15);

Frage 4.1 (4 Punkte): Die zugehörige Entity hat folgende Eingänge und Ausgänge:

Clk : in std\_logic;

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

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

Skizzieren Sie das Blockschaltbild des Rechenwerks inklusive der intern verwendeten Signale.

Frage 4.2 (6 Punkte): Skizzieren Sie das Blockschaltbild des Rechenwerks mit den beteiligten Prozessen. Hinweis: Nur die Prozesse mit den Signalen, die sie lesen, schreiben oder intern verwenden. Zum detaillierten Innenleben der Prozesse folgt eine weitere Frage.

Frage 4.3 (6 Punkte): Das Signale H ist ein Array von 16-Bit Worten vom Datentyp signed, S ist ein 16 Bit-Wert vom Datentyp signed. Die Signale P und Q sind ein Arrays von 32-Bit Worten vom Datentyp signed. Skizzieren Sie das Innenleben der Prozesse (den internen Signalfluss) und rekonstruieren Sie die Struktur des Rechenwerks. Welcher Algorithmus wird hiermit ausgeführt?

Frage 4.4 (8 Punkte): Tests. Skizzieren Sie die Struktur von Testfällen in einem Testprogramm für den Algorithmus. Hinweis: Konzept und Aufbau genügen, HDL ist nicht unbedingt erforderlich. 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 5 - Registerblock (14 Punkte)

Der Prozessor soll mit einem Registerblock ausgestattet werden, der aus Dual-Ported RAM ausgeführt ist, also den Zugriff zum Lesen und Schreiben auf 2 Ports zulassen soll. Insgesamt sollen 256 Register mit einer Wortbreite von jeweils 16 Bit zur Verfügung stehen.

Frage 5.1 (4 Punkte): Skizzieren Sie ein Blockschaltbild des Registerblocks mit seinen Eingängen und Ausgängen.

Frage 5.2 (6 Punkte): Skizzieren Sie eine mögliche Realisierung des Registerblocks in HDL. Hinweis: Struktur mit Erläuterung und Ausführung der wesentlichen Teile genügt.

Frage 5.3 (4 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.