Aufgabe 1 (24 Punkte)

Zynq Architektur

Beantworte Sie die folgenden Fragen stichpunktartig. Formulieren Sie die Antworten so knapp wie möglich und so ausführlich wie nötig.

1. Welche der folgenden PS-PL Schnittstellen wird für die Kommunikation zwischen den ARM-Prozessoren und AXI GPIOs im Zynq Baustein verwendet?
   (a) S_AXI_GP
   (b) M_AXI_GP
   (c) S_AXI_ACP
   (d) S_AXI_HP

   (2)

2. Das AXI Interconnect Schema dient der Umsetzung mehrerer Funktionen. Zählen Sie mindestens drei Funktionen vom AXI Interconnect auf.
   - ~~Adressierung~~ Dient als Master für angeschlossene AXI-IP-Instanzen
   - verwaltet & lenkt den Verkehr zwischen angeschlossenen AXI Interfaces
   - agiert als Slave vom PS

3. Was ist der Unterschied zwischen AXI Interconnect und AXI ~~Interface~~?
   Interface ist eine ~~Punkt zu Punkt~~ Verbindung zum Weiterleiten von Daten, Adressen & ~~Handshake~~-Signalen.
   Interconnect fungiert als Switch und verwaltet & lenkt den Verkehr zwischen AXI Interfaces.

4. Was sind die maßgeblichen Vorteile vom Zynq gegenüber einem traditionellen FPGA (z.B. Virtex 7) mit Softcore-Prozessorkern (z.B. MicroBlaze)?
   Durch ein festes Processing System mit AXI-Interfaces kann die PL so gestaltet werden wie man möchte.

5. Inwiefern können die absoluten Laufzeiten einer HW-/SW-Implementierung auf einer Zynq-Plattform vorhergesagt werden? Wovon hängen sie ab?
   - Zeitdauer des kritischen Pfades
   - von den benötigten Zyklen sowie den anzusteuernden Speicher (intern, extern)

6. Erläutern Sie die Notwendigkeit für das Schlüsselwort *volatile* in der folgenden Definition von Xil_In32():

```
u32 Xil_In32(u32 Addr)
{
   RETURN *(VOLATILE u32 *) Addr;
}
```

stellt eine unveränderbare Variable dar.

(0)

Aufgabe 2 (14 Punkte)

Performance und Timing im Zynq

Berechnen Sie anhand der unten abgebildeten Tabelle die Latenz in Nanosekunden für die Kommunikation zwischen:

1. DMA und DDR unter Verwendung der AXI-HP Schnittstelle

2. ARM und internem Speicher des AXI-Slaves im PL unter Verwendung der AXI-GP Schnittstelle

Hinweis: Entscheidend ist wie schnell das Processing System läuft, nicht der absolute Wert des On-Board Referenzclock-Bausteins. Das Clocking der PL ist unabhängig vom PS.

	L1 Cache	L2 Cache	DDR	OCM	IOP Slave	M_AXI_GP0
CPU Pipeline	1	25	67	20	122	86
Peripheral Master	–	–	136	106	–	126
S_AXI_ACP	27	32	89	27	124	–
S_AXI_HP0	–	–	76	46	–	–
S_AXI_GP0	–	–	118	88	144	–

Aufgabe 3 (16 Punkte)

Speicher im FPGA

In FPGAs existieren für gewöhnlich unterschiedliche Möglichkeiten Speicher zu realisieren.

1. Nennen Sie mindestens vier wesentliche Unterschiede zwischen Distributed RAM und Block RAM im Zynq Baustein.

   1. Block-RAM hat Port Aspect Ratios
   2. BlockRAM hat möglich 2 Ports, D-RAM hingegen 4
   3. D-RAM hat unabhängige Carry Logik
   4. B-RAM wird basierend auf der Konfig des BRAM Interface Controller IP erzeugt

2. Ermitteln Sie die Anzahl der Adress-, Daten- und Parity-Bits für PORT A eines 36k BRAM mit 2048 Speicherplätzen, der als True Dual-Port RAM konfiguriert ist.

3. Bestimmen Sie die Anzahl der MLUTs, die zum Implementieren von 128 x 8 Dual-Port RAM erforderlich sind.

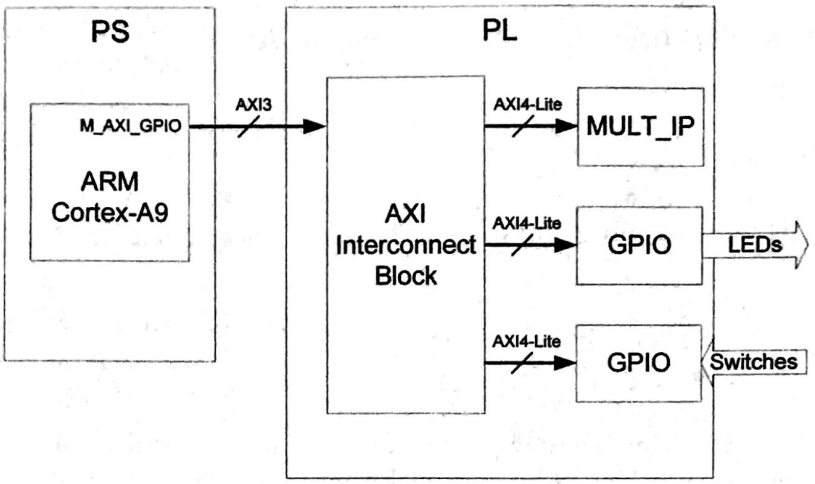
   32 MLUTs

   (0)

Aufgabe 4 (28 Punkte)

Systementwurf

Es soll ein HW-/SW-System gemäß folgender Abbildung realisiert werden.



1. Der IP Core MULT_IP stellt einen Hardware Beschleuniger für die Multiplikation zweier 16 Bit Zahlen dar (Signale *ain* und *bin*). Er arbeitet synchron auf die steigende Taktflanke des Clock-Signals getriggert und macht Gebrauch von einem low-active Reset-Signal, um das Ergebnis-Signal zurückzusetzen. Vervollständigen Sie das auf der nächsten Seite gegebene VHDL-Gerüst gemäß beschriebener Funktionalität.

2. Für die Einbindung über AXI werden zwei Dateien generiert: mult_ip_v1_0_S00_AXI.vhd und mult_ip_v1_0.vhd. Skizzieren Sie die hierarchischen Zusammenhänge und erläuern Sie stichpunktartig Sinn und Aufgabe der beiden Dateien.

3. Ergänzen Sie den umseitig abgebildeten VHDL-Auszug für die Integration des fertigen IP Cores MULT_IP in das AXI Umfeld. Welcher Datei ist dieser Auszug zuzuordnen, mult_ip_v1_0_S00_AXI.vhd oder mult_ip_v1_0.vhd?

(07)

VHDL-Gerüst für MULT-IP

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;
ENTITY mult_logic IS
    PORT ( ain : IN  std_logic_vector(          );
           bin : IN  std_logic_vector(          );
           clk : IN  STD_LOGIC;
           rstn : IN  std_logic;
           result : OUT  std_logic_vector(          ));
END mult_logic;
ARCHITECTURE arch OF mult_logic IS
BEGIN

END arch;
```

VHDL-Auszug aus AXI-spezifischer Implementierung

```
ARCHITECTURE arch_imp OF                    IS
   -- Add user signals and constants here

   -- User user signals and constants end here
BEGIN
   -- Add user logic here

   -- User logic ends
END arch_imp;
```

Aufgabe 5 (18 Punkte)

Hardware Beschleunigung und High-Level Synthese

Häufig lässt sich ein Performance-Gewinn erzielen, indem Funktionen in Hardware ausgelagert werden.

1. Stellen Sie stichpunktartig den Zusammenhang her zwischen Design-Geschwindigkeit und Taktgeschwindigkeit, sowohl für eine Software- als auch für eine Hardware-Implementierung.

PS: Hängt ab von den benötigten Zyklen sowie den anzusteuernden Speichern (intern, extern)

PL: Zeitdauer des kritischen Pfades beachten

2. Schätzen Sie ab wie lang der folgende Code für einen zufällig generierten 32-Bit-Wert benötigen würde. Dabei geht es nicht um die absolute Anzahl an Maschinenbefehlen (ARM Instruction Set nicht vollständig im Unterricht behandelt), sondern um das Abschätzen welche und wieviele Elementaroperationen in einem Befehl (der höheren Programmiersprache C) zusammengefasst sind. Unter Elementaroperationen fallen z.B. arithmetische Grundoperationen (Addition, Subtraktion, Multiplikation, Division), Vergleiche, Sprünge, Zuweisungen und Zeiger- und Indexauswertungen (Speicherzugriff).

```
int32_t hamming_distance(uint32_t x, uint32_t y)
{
   int32_t dist = 0;          2
   uint32_t val;              1
   val = x ^ y; // XOR        3
   // Count the number of bits set
   WHILE (val != 0) {         bei 10 Durchläufen
      dist++;                 1
      val &= val - 1;         3
   }
RETURN dist;
```

Multiplikation = 6 Zyklen

Wenn val mit 10 durchlaufen wird, würde ich auf 16 Zyklen * schätzen.

3. Wie würde ein entsprechender Hardware Beschleuniger aussehen? Was wären die möglichen Performance-Gewinne?

4. Welche FPGA Ressourcen (BRAM_18K, DSP48E, FF, LUT) würden vermutlich durch ein High-Level Syntheseergebnis aloziiert werden? Hier ist auch zu begründen wie (wofür) diese Ressourcen genutzt werden.

Kein DSP48E, da keine Multiplikation.
BRAM-18K zum Zwischenspeichern der Werte.
FF und LUT würden zum Einsatz kommen.

(0)