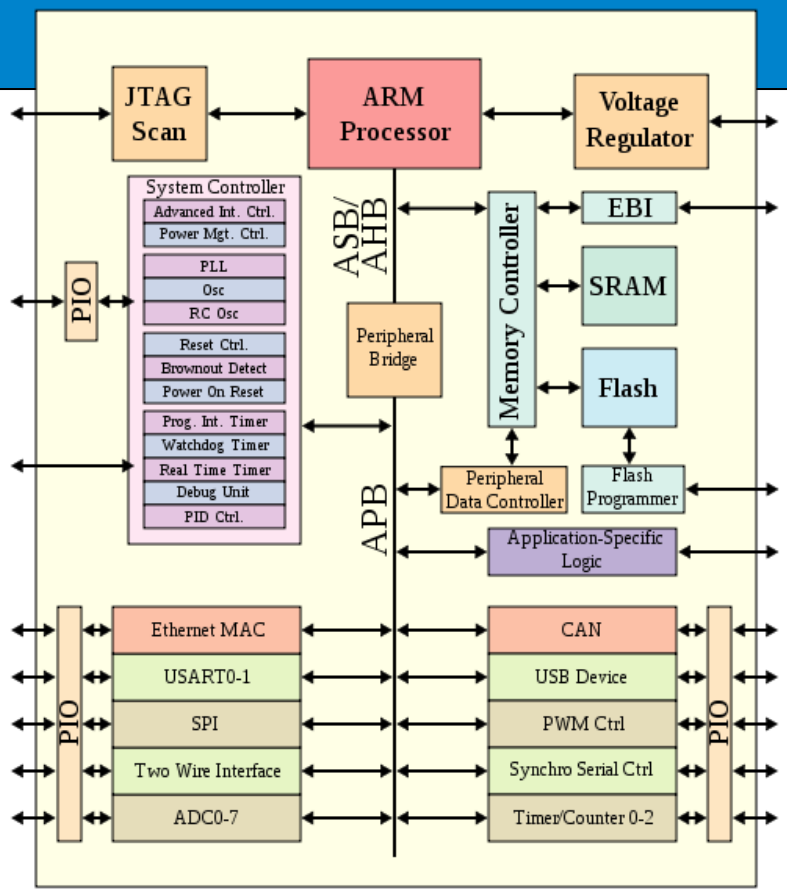
Architekturen und Entwurf von Rechnersystemen Wintersemester 2016/2017

TECHNISCHE UNIVERSITÄT DARMSTADT

Hörsaalübung 5:

Übungsbesprechung 5 und 6
Übungsvorstellung 7
Klausurfragen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Übung 5

Übung 5: Bildfilter

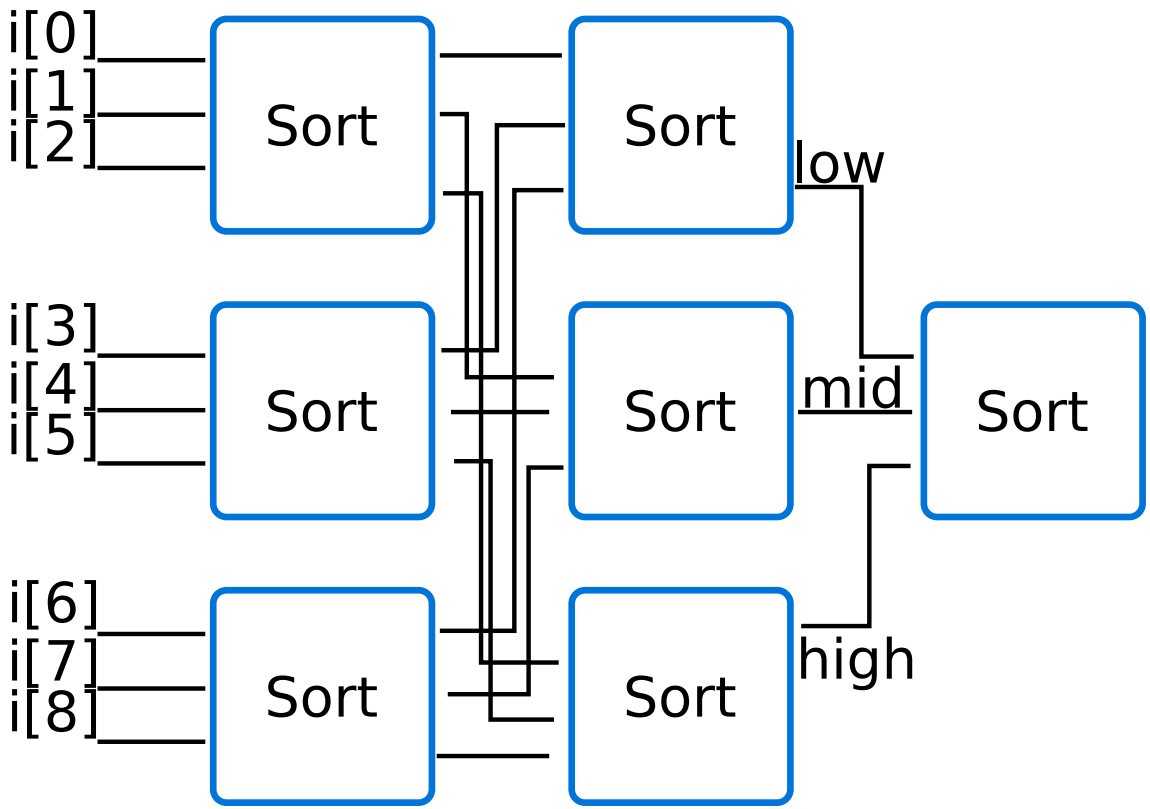
TECHNISCHE UNIVERSITÄT DARMSTADT

- Median Filter
- Schwierigkeit: Richtige Daten zur richtigen Zeit
- Bild-Buffer wird aus Gründen der Übersichtlichkeit nicht vorgestellt → Beispiellösung

Übung 5: Median

TECHNISCHE UNIVERSITÄT DARMSTADT

- Schritt 1: Median aus neun Werten finden



Übung 5: Drei-Eingang-Sortierer

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1    rule sort;
2        let i = in.first(); in.deq();
3        let xored = i[0] ^ i[1] ^ i[2];
4        Sorted tVal;
5        tVal.max = max(i[0], max(i[1], i[2]));
6        tVal.min = min(i[0], min(i[1], i[2]));
7        tVal.med = xored ^ tVal.max ^ tVal.min;
8        out.enq(tVal);
9    endrule
```

Übung 5: Erste Sortierstufe

TECHNISCHE UNIVERSITÄT DARMSTADT

Alle Werte einmal sortieren lassen

```
1   Vector#(7, Server#(Vector#(3, GrayScale), Sorted)) sortingNetwork <-
    ↪   replicateM(mkSort());
2
3   rule firstStage;
4       let t = in.first(); in.deq();
5
6       Vector#(3, Vector#(3,GrayScale)) c = unpack(pack(t));
7
8       for(Integer i = 0; i < 3; i = i + 1) begin
9           sortingNetwork[i].request.put(c[i]);
10      end
11  endrule
```

Übung 5: Zweite Sortierstufe

TECHNISCHE UNIVERSITÄT DARMSTADT

Zweite Stufe: Alle 'high'-Werte in obersten Sortierer usw.

```
1    rule secondStage;
2        Vector#(3, Vector#(3, GrayScale)) sorted;
3        for(Integer i = 0; i < 3; i = i + 1) begin
4            let tVal <- sortingNetwork[i].response.get();
5            sorted[i] = unpack(pack(tVal));
6        end
7
8        for(Integer i = 0; i < 3; i = i + 1) begin
9            Vector#(3, GrayScale) iSort;
10           for(Integer j = 0; j < 3; j = j + 1) begin
11               iSort[j] = sorted[j][i];
12           end
13           sortingNetwork[3 + i].request.put(iSort);
14       end
15   endrule
```

Übung 5: Dritte Sortierstufe

TECHNISCHE UNIVERSITÄT DARMSTADT

Dritte Stufe: ’low’-Wert aus oberem Sortierer, ’mid’-Wert aus mittlerem Sortierer und ’high’-Wert aus unterem Sortierer.

```
1    rule thirdStage;
2        Vector#(3, GrayScale) sorted;
3        for(Integer i = 0; i < 3; i = i + 1) begin
4            let tVal <- sortingNetwork[3 + i].response.get();
5            Vector#(3, GrayScale) tSorted = unpack(pack(tVal));
6            sorted[i] = tSorted[2 - i];
7        end
8        sortingNetwork[6].request.put(sorted);
9    endrule
```

Übung 5: Testen mit BlueCheck

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sortierfunktion auf Listen definiert
- Sehr langsam in Hardware
- Perfekt zum testen

```
1    function ActionValue#(GrayScale) getMedian();
2        actionvalue
3            let s = specFIFO.first(); specFIFO.deq();
4            List#(GrayScale) l = toList(s);
5            l = sort(l);
6
7            return l[4];
8        endactionvalue
9    endfunction
```

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Übung 6: ESABus

Übung 6: ESABus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bus System ähnlich AXI4-Lite
- Getrennte Kanäle zum Lesen und Schreiben von Daten
- Einfacher Speicher als ESABus-Slave
- Verbindungsmöglichkeit über Connectable-Typklasse
- Verbindungsmöglichkeit für mehrere Master und Slaves

Übung 6: Das Interface

TECHNISCHE UNIVERSITÄT DARMSTADT

- Busseite und Modulseite getrennt
- Verbindung über `fab` Interface
- Erlaubt beliebig komplexe Busprotokolle zu verstecken
- Beispiel: DDR3 oder PCIe

```
1    interface ESABusWr_Master_Fabric;
2        method ActionValue#(Tuple2#(Bit#(16), Bit#(8))) request;
3    endinterface
4
5    interface ESABusWr_Master;
6        interface ESABusWr_Master_Fabric fab;
7
8        interface Put#(Tuple2#(Bit#(16), Bit#(8))) request;
9    endinterface
```

Übung 6: Der Master

TECHNISCHE UNIVERSITÄT DARMSTADT

- Dazu der passende Master

```
1   module mkESABusWr_Master(ESABusWr_Master);
2
3       FIFO#(Tuple2#(Bit#(16), Bit#(8))) requestIn <- mkFIFO();
4
5       interface ESABusWr_Master_Fabric fab;
6           method ActionValue#(Tuple2#(Bit#(16), Bit#(8))) request;
7               requestIn.deq();
8               return requestIn.first();
9           endmethod
10      endinterface
11
12      interface Put request = toPut(requestIn);
13  endmodule
```

Übung 6: ESAMem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Speicher der diesen Bus einsetzt

```
1   module mkESAMem(ESAMem);
2       ESABusWr_Slave writeSlave <- mkESABusWr_Slave();
3       ESABusRd_Slave readSlave <- mkESABusRd_Slave();
4
5       RegFile#(Bit#(12), Bit#(8)) mem <- mkRegFileFull();
6
7       rule handleWrite;
8           let r <- writeSlave.request.get();
9           mem.upd(truncate(tpl_1(r)), tpl_2(r));
10      endrule
11
12      rule handleRead;
13          let r <- readSlave.request.get();
14          readSlave.response.put(mem.sub(truncate(r)));
15      endrule
16
17      interface ESABusWr_Slave_Fabric s_wr = writeSlave.fab;
18      interface ESABusRd_Slave_Fabric s_rd = readSlave.fab;
19  endmodule
```

Übung 6: Testbench

TECHNISCHE UNIVERSITÄT DARMSTADT

- BlueCheck testet lokalen Speicher gegen Speicher über den Bus
- Verwendet Testbench mit Reset für schnellere Fehlerfindung

```
1    module [BlueCheck] mkESAMemSpec#(Reset r)();
2        RegFile#(Bit#(12), Bit#(8)) mem <- mkRegFileFull(reset_by r);
3        FIFO#(Bit#(16)) readRequest <- mkFIFO(reset_by r);
4        ESAMem impl <- mkESAMem(reset_by r);
5
6        ESABusWr_Master writeMaster <- mkESABusWr_Master(reset_by r);
7        ESABusRd_Master readMaster <- mkESABusRd_Master(reset_by r);
8        mkConnection(writeMaster.fab, impl.s_wr, reset_by r);
9        mkConnection(readMaster.fab, impl.s_rd, reset_by r);
10   ...
```

Übung 6: "Trap for young players"

TECHNISCHE UNIVERSITÄT DARMSTADT

- ESAConnect für die Verbindung von mehreren Mastern und Slaves
- Achtung: Implizite Bedingungen in Guards

```
1    rule sendRequest if(arbiter.clients[i].grant());
2        slaves[whichSlave].write(tpl_1(request), tpl_2(request));
3    endrule
```

Übung 6: "Trap for young players"

TECHNISCHE UNIVERSITÄT DARMSTADT

- ESAConnect für die Verbindung von mehreren Mastern und Slaves
- Achtung: Implizite Bedingungen in Guards

```
1    rule sendRequest if(arbiter.clients[i].grant());
2        slaves[whichSlave].write(tpl_1(request), tpl_2(request));
3    endrule
```

```
1    Rule: sendRequest
2    Predicate: slave_1_writeSlave_requestIn.i_notFull &&
     ↪   slave_2_writeSlave_requestIn.i_notFull && request_1.whas &&
     ↪   arbiter_grant_vector.wget[1]
3    Blocking rules: (none)
```

Übung 6: "Trap for young players"

TECHNISCHE UNIVERSITÄT DARMSTADT

- Stattdessen: Eine Rule pro Slave

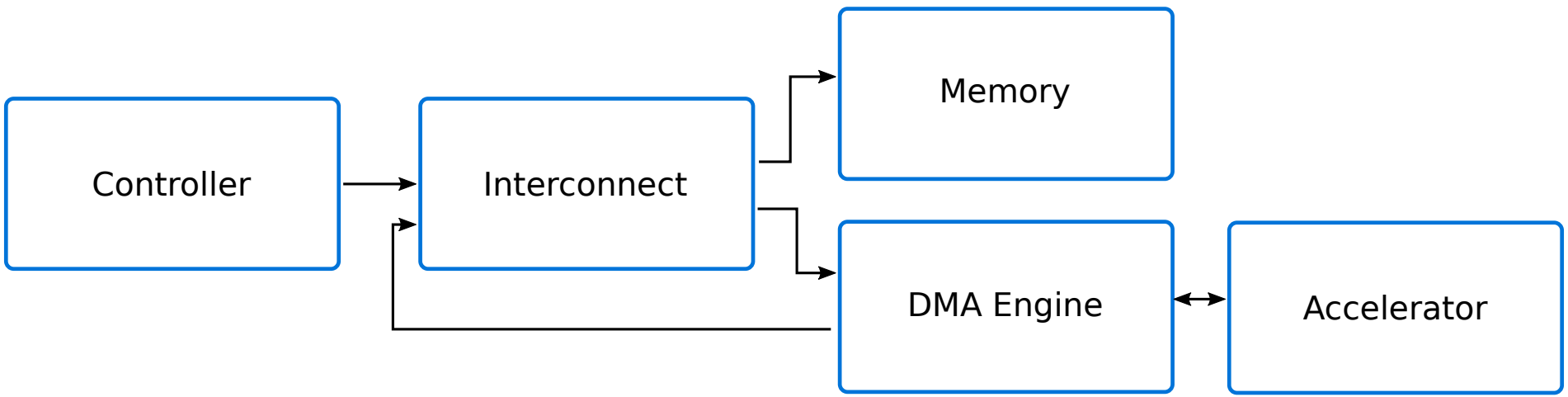
```
1    for(Integer j = 0; j < valueOf(nSlaves); j = j + 1) begin
2     rule sendRequest if(arbiter.clients[i].grant() && whichSlave ==
     ↪   fromInteger(j));
3       slaves[j].write(tpl_1(request), tpl_2(request));
4     endrule
5    end
```

```
1    Rule: sendRequest
2    Predicate: slave_1_writeSlave_requestIn.i_notFull && request_1.whas &&
     ↪   arbiter_grant_vector.wget[1]
3    Blocking rules: (none)
```

Übung 7: ESASoC

TECHNISCHE UNIVERSITÄT DARMSTADT

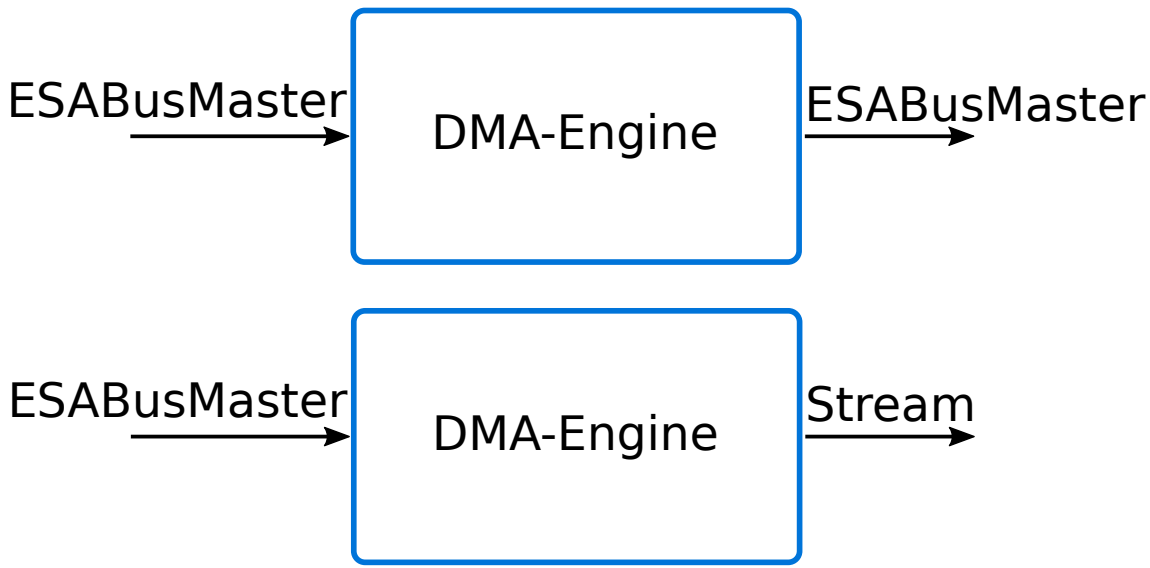
- Ein SoC mit den Komponenten der vorherigen Übungen bauen
- Statt Prozessor: StmtFSM als Steuerung



Übung 7: DMA-Einheit

TECHNISCHE UNIVERSITÄT DARMSTADT

- Direct-Memory-Access: Liest Daten von einer Stelle und schreibt an eine Andere
  - Liest Daten von Speicher A und schreibt in Speicher B
  - Liest Daten von Speicher A und stellt sie als Stream zur Verfügung
- Prozessor muss nur noch DMA-Einheit konfigurieren



Klausurfragen

Beispielfragen zur Klausur: Einfacher Einstieg

TECHNISCHE UNIVERSITÄT DARMSTADT

Nennen Sie zwei fundamentale Eigenschaften einer Bluespec Regel.

Beispielfragen zur Klausur: Einfacher Einstieg

TECHNISCHE UNIVERSITÄT DARMSTADT

Nennen Sie zwei fundamentale Eigenschaften einer Bluespec Regel.

- rules are atomic
- rules fire or don’t at most once per cycle
- rules don’t conflict with other rules

Beispielfragen zur Klausur: BSV Verständnisfragen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1   typedef Server#(Tuple2#(Int#(32), Int#(32)), Int#(32)) PythagorasServer;
2     module mkCalcPythagoras(PythagorasServer);
3       FIFO#(Tuple2#(Int#(32), Int#(32))) in <- mkFIFO();
4       FIFO#(Int#(32)) out <- mkFIFO();
5       Server#(Int#(32), Int#(32)) sqrt <- mkSqrt(); // Module to calculate
    ↪   Square Root
6       rule calc; // c = sqrt(a^2 + b^2)
7         let val = in.first(); in.deq();
8         let a = tpl_1(val);
9         let b = tpl_2(val);
10        let aSqr = a * a;
11        let bSqr = b * b;
12        sqrt.request.put(aSqr + bSqr);
13      endrule
14      rule fetchResult;
15        let val <- sqrt.response.get();
16        out.enq(val);
17      endrule
18    interface Put = toPut(in);
19    interface Get = toGet(out);
20  endmodule
```

Welches Problem ergibt sich bei diesem Modul in Hinblick auf den kritischen Pfad?

Beispielfragen zur Klausur: BSV Verständnisfragen

TECHNISCHE UNIVERSITÄT DARMSTADT

Welches Problem ergibt sich bei diesem Modul in Hinblick auf den kritischen Pfad?

```
1    rule calc; // c = sqrt(a^2 + b^2)
2      let val = in.first(); in.deq();
3      let a = tpl_1(val);
4      let b = tpl_2(val);
5      let aSqr = a * a;
6      let bSqr = b * b;
7      sqrt.request.put(aSqr + bSqr);
8    endrule
```

Beispielfragen zur Klausur: Erweiterte Bluespec Features

TECHNISCHE UNIVERSITÄT DARMSTADT

Geben Sie die korrekten Provisos für folgendes Modul an. Der Eingabetyp maximumValue gibt den höchsten zu verarbeitenden Wert an. Hinweise:

- Erzeugen Sie mit Hilfe der Provisos die Typen operandsBits und resultBits.
- Die Bitbreite des Ergebnisses einer Multiplikation ist höchstens die Summe der Bitbreite der Operanden.
- Stellen Sie sicher, dass resultBits kleiner oder gleich 64 bit ist.
- Stellen Sie sicher, dass operandsBits kleiner oder gleich resultBits ist.

```
1    module mkMult(Mult#(maximumValue))
2      provisos(
```

Beispielfragen zur Klausur: Erweiterte Bluespec Features

TECHNISCHE UNIVERSITÄT DARMSTADT

Geben Sie die korrekten Provisos für folgendes Modul an. Der Eingabetyp maximumValue gibt den höchsten zu verarbeitenden Wert an. Hinweise:

- Erzeugen Sie mit Hilfe der Provisos die Typen operandsBits und resultBits.
- Die Bitbreite des Ergebnisses einer Multiplikation ist höchstens die Summe der Bitbreite der Operanden.
- Stellen Sie sicher, dass resultBits kleiner oder gleich 64 bit ist.
- Stellen Sie sicher, dass operandsBits kleiner oder gleich resultBits ist.

```
1   module mkMult(Mult#(maximumValue))
2     provisos(
```

```
1       Log#(maximumValue, operandsBits),
2       Mul#(operandsBits, 2, resultBits),
3       Add#(operandsBits, someOtherVal, resultBits),
4       Add#(resultBits, someVal, 64)
5     );
```

Beispielfragen zur Klausur: Verständnisfrage SoC

TECHNISCHE UNIVERSITÄT DARMSTADT

Warum benötigt ein SoC wie das Zynq7000 verschiedene Speicher?

Beispielfragen zur Klausur: Verständnisfrage SoC

TECHNISCHE UNIVERSITÄT DARMSTADT

Warum benötigt ein SoC wie das Zynq7000 verschiedene Speicher?

Speicher werden für verschiedene Einsatzzwecke benötigt. Dabei unterscheiden sich die Speicher z.B. in Platzierung, Interfaces, Verzögerungen, Bandbreite etc.

Beispielfragen zur Klausur: Verständnisfrage SoC

TECHNISCHE UNIVERSITÄT DARMSTADT

Nennen Sie zwei Speicher die auf dem Zynq 7000 SoC vorhanden sind und deren Funktion.

Beispielfragen zur Klausur: Verständnisfrage SoC

TECHNISCHE UNIVERSITÄT DARMSTADT

Nennen Sie zwei Speicher die auf dem Zynq 7000 SoC vorhanden sind und deren Funktion.

OCM: Zugriff mit geringer Latenz
BlockRAM: Schnelle Speicher für Beschleuniger in PL

Beispielfragen zur Klausur: Verständnisfrage SoC

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Art von Recheneinheit im Zynq 7000 SoC ist für Vektoroperationen optimiert?

Beispielfragen zur Klausur: Verständnisfrage SoC

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Art von Recheneinheit im Zynq 7000 SoC ist für Vektoroperationen optimiert?

Die NEON-Einheit mit SIMD-Instruktionen.

Beispielfragen zur Klausur: AXI

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Funktion hat AXI in einem System-on-Chip?

Beispielfragen zur Klausur: AXI

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Funktion hat AXI in einem System-on-Chip?

Bussystem. Vernetzung von Komponenten.

Beispielfragen zur Klausur: AXI

TECHNISCHE UNIVERSITÄT DARMSTADT

Was ist der wesentliche Unterschied zwischen AXI ACP und AXI HP Ports auf Zynq SoCs?

Beispielfragen zur Klausur: AXI

TECHNISCHE UNIVERSITÄT DARMSTADT

Was ist der wesentliche Unterschied zwischen AXI ACP und AXI HP Ports auf Zynq SoCs?

ACP = Cache-Kohärent - Teilt sich Cache mit CPU
HP = Cache-Inkohärent - Direkter Zugriff auf DDR3

Beispielfragen zur Klausur: AXI

TECHNISCHE UNIVERSITÄT DARMSTADT

Welchen Vorteil bieten Burst-Transfers gegenüber einzelnen Anfragen?

Beispielfragen zur Klausur: AXI

TECHNISCHE UNIVERSITÄT DARMSTADT

Welchen Vorteil bieten Burst-Transfers gegenüber einzelnen Anfragen?

Handshake-Overhead wird vermieden.
Latenz Vermeidung (z.B. bei DDR).

Beispielfragen zur Klausur: Bluespec Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Signale hat folgendes Bluespec Interface nach der Synthese in Verilog?

```
1    interface Foo;
2      method Action bar();
3      method ActionValue#(UInt#(32)) foobar();
4    endinterface
```

Beispielfragen zur Klausur: Bluespec Interfaces

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Signale hat folgendes Bluespec Interface nach der Synthese in Verilog?

```
1    interface Foo;
2      method Action bar();
3      method ActionValue#(UInt#(32)) foobar();
4    endinterface
```