Boundary Scan Days® 2009

Einsatz von Virtual JTAG (Altera) für Flash - & EEPROM - Programmierung

Dammert Tobias & Knüppel Lars
Nokia Siemens Networks GmbH & Co. KG – Standort Bruchsal
Test Engineering

Nokia Siemens Networks

Agenda

Herr Knüppel

- Einführung
- Motivation
- Lösungsansatz

Herr Dammert

- Anwendungsbeispiel
- Umsetzung/Erstellung

Nokia Siemens Networks GmbH & Co. KG

Standort Bruchsal

Nokia Siemens ist ein Gemeinschaftsunternehmen zwischen der deutschen Siemens AG & der finnischen Nokia Oyj

Das Nokia Siemens Networks Produktportfolio besteht aus Hard- und Softwarekomponenten für Sprach- und Datenkommunikation in Fest- und Mobilfunknetzen sowie Dienstleistungen zur Konfiguration, Installation und Wartung.

Motivation / Problemstellung

Motivation

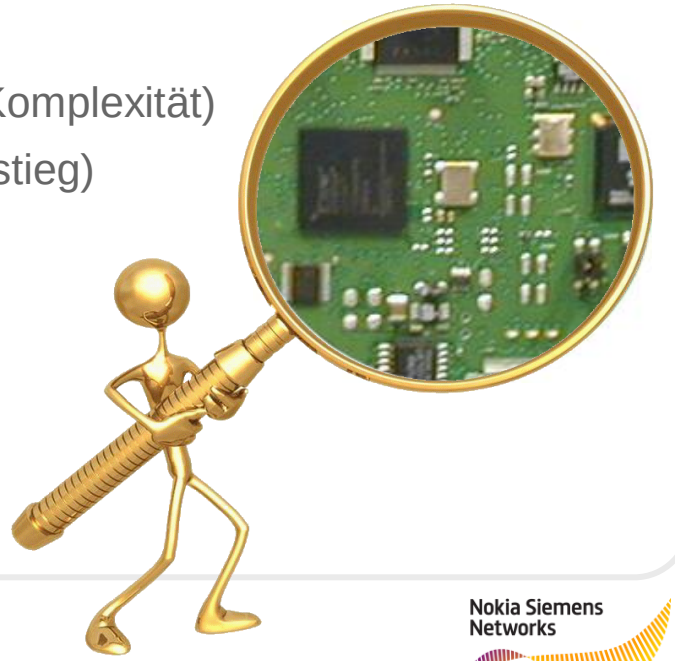
- Frequenzmessung (z.B. Quarze) aufgrund fehlerhafter Bestückung
- Erhöhung des Test Coverage
- Kein „Funktionstest“
- Programmierung von Config-EEPROMS
  - (Hervorragender Support von Goepel bei der Beseitigung der Quartus II Bugs)

Ansatz

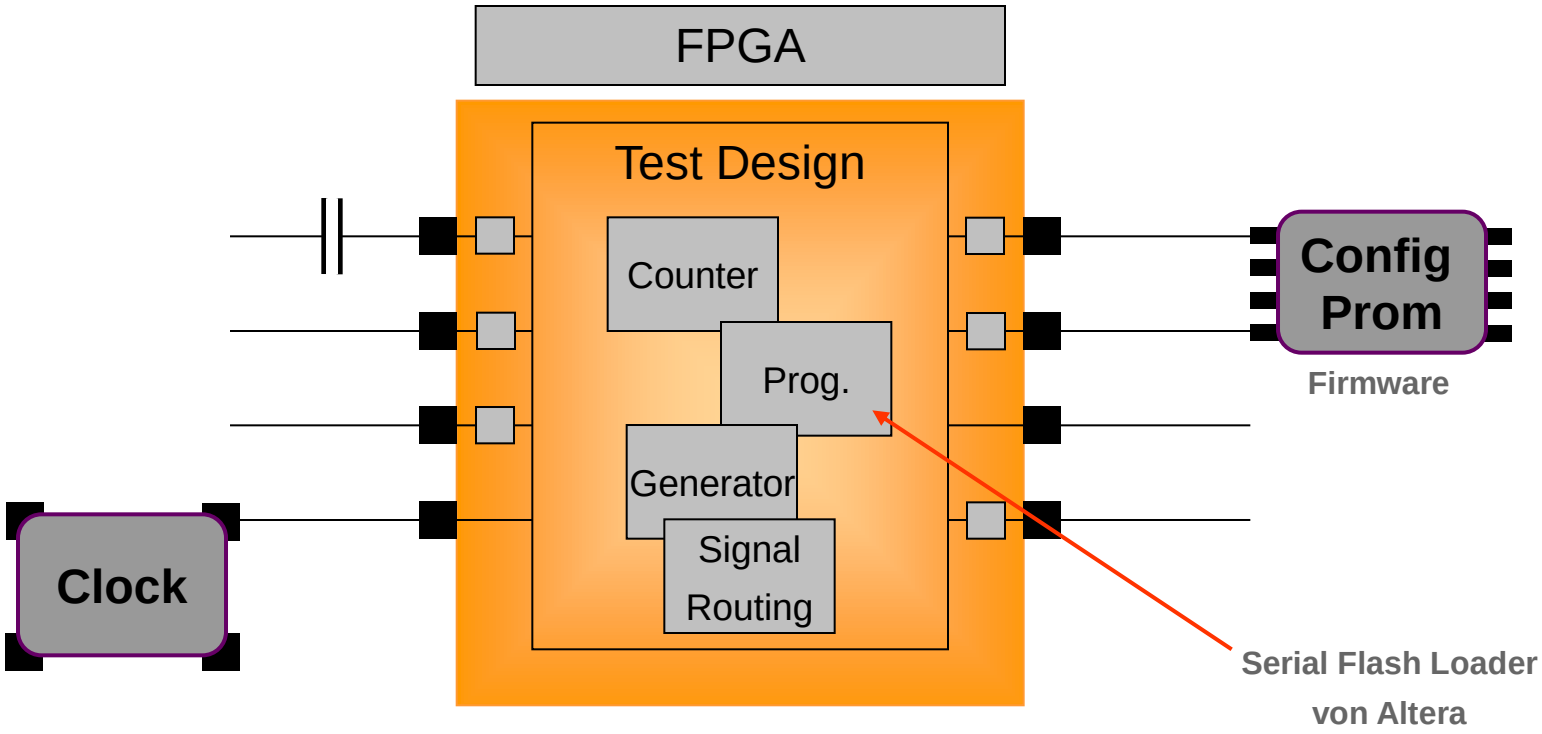
- Verwendung der prüflingseigenen Ressourcen
- a) durch Prozessoren (Umfangreiches Know-how, hohe Komplexität)
- b) durch FPGA‘s (Know-how erforderlich, schnellerer Einstieg)

Entscheidung

- Verwendung der Altera FPGA‘s
- Entwicklungstools verfügbar
- Know-how verfügbar

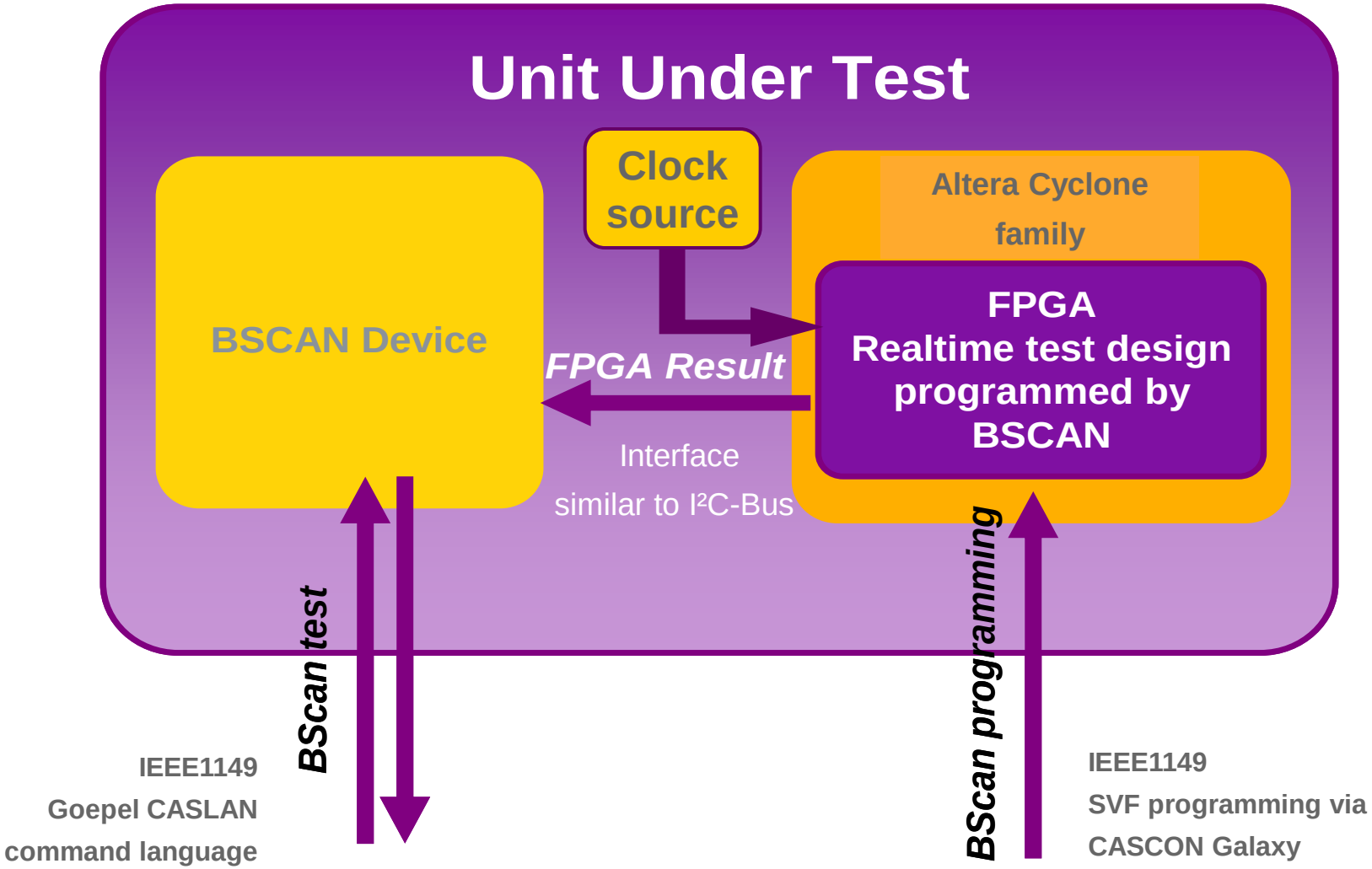


Prinzip / Ansatz



Zugriff auf alle Pins
(auch ohne BSCAN Funktionalität)

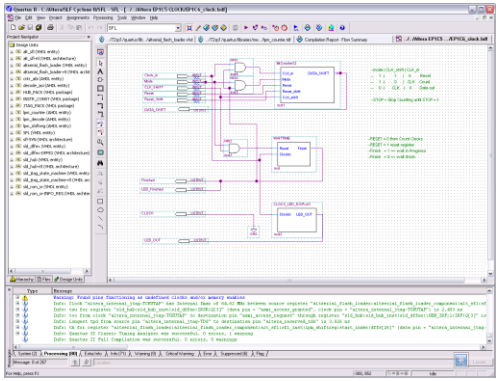
Die Anwendung



Ablauf / Fazit / Fragestellung

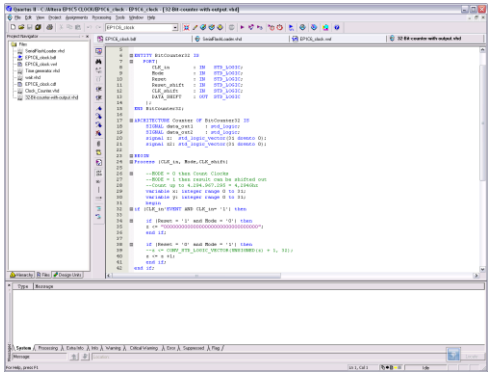
Ablauf

- Erstellen eines Prototypen FPGA Design mit VHDL
- Programmierung des Design via SVF mit CASCON GALAXY
- Entwurf der FPGA Steuerung mit der CASLAN-Syntax



Fazit

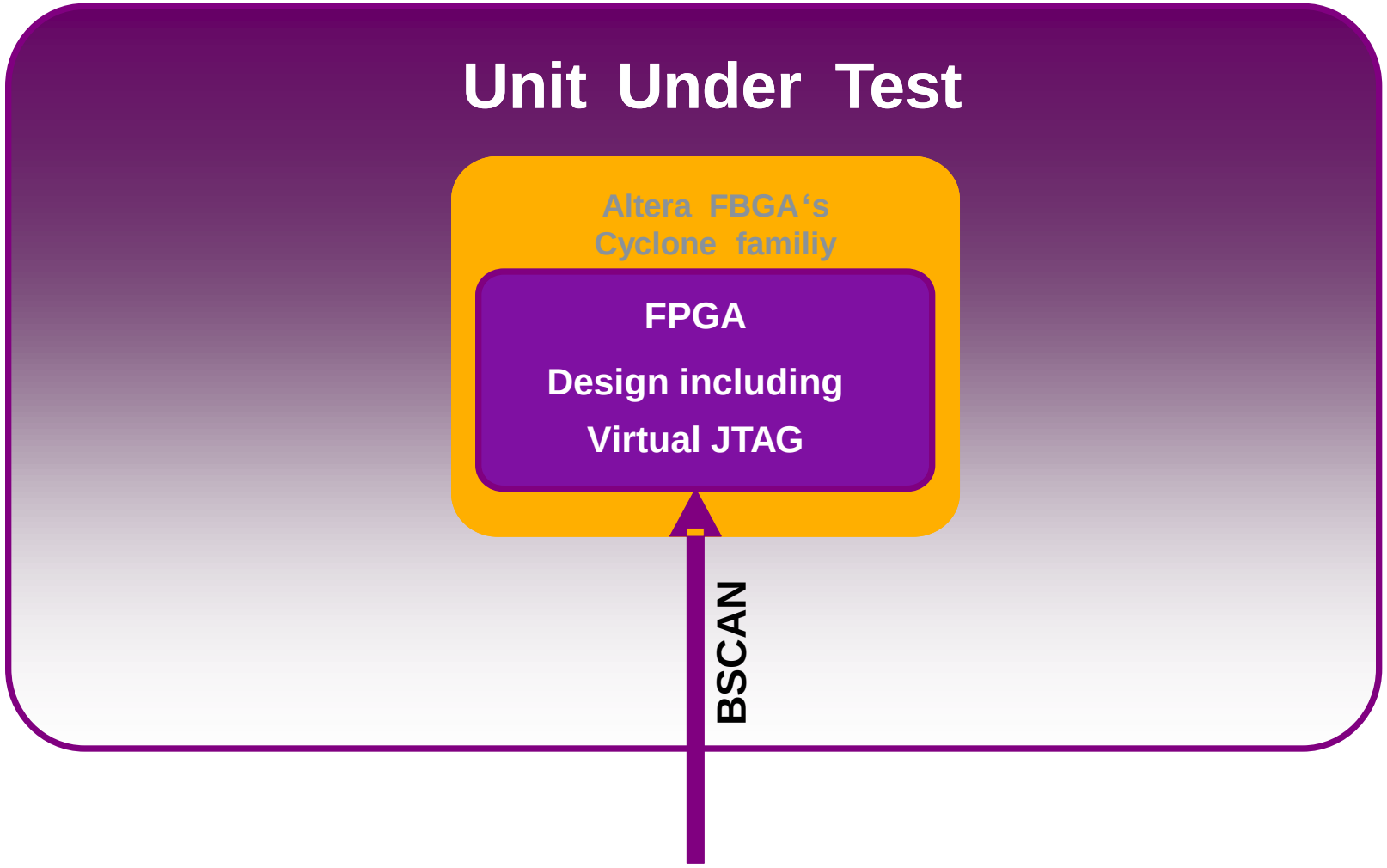
- Nutzen abhängig vom Board Design
- Einfache Anwendungen kurzfristig realisierbar
- Genauigkeit der Frequenzmessung abhängig vom BG Referenztakt
- Bei steigender Komplexität steigt der Steuerungsaufwand



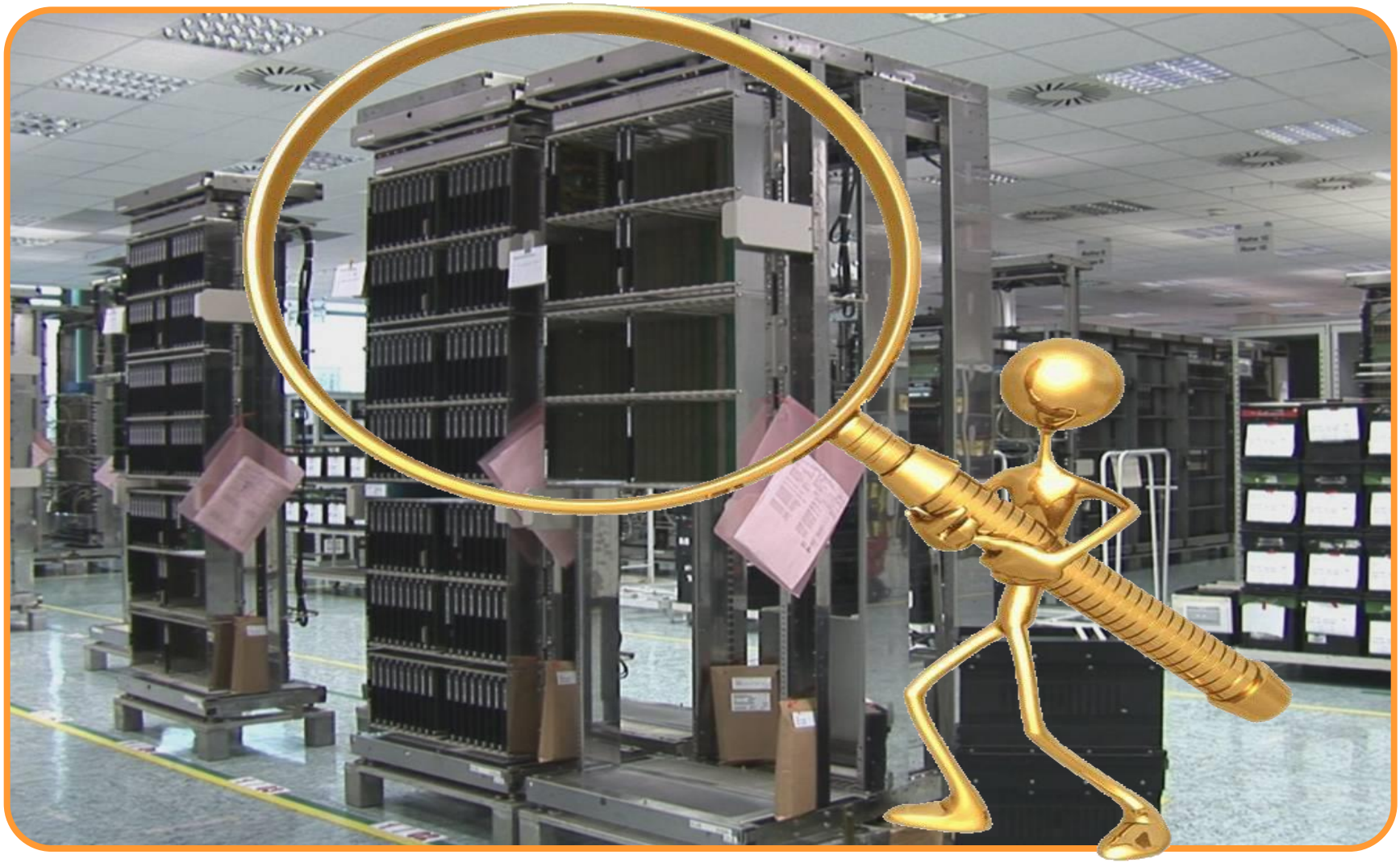
Fragen:

- Schnellere Programmierung möglich, Echtzeit?
- RAM, Flash, etc. testbar ?
- Einfachere Steuerungsfunktionalität?

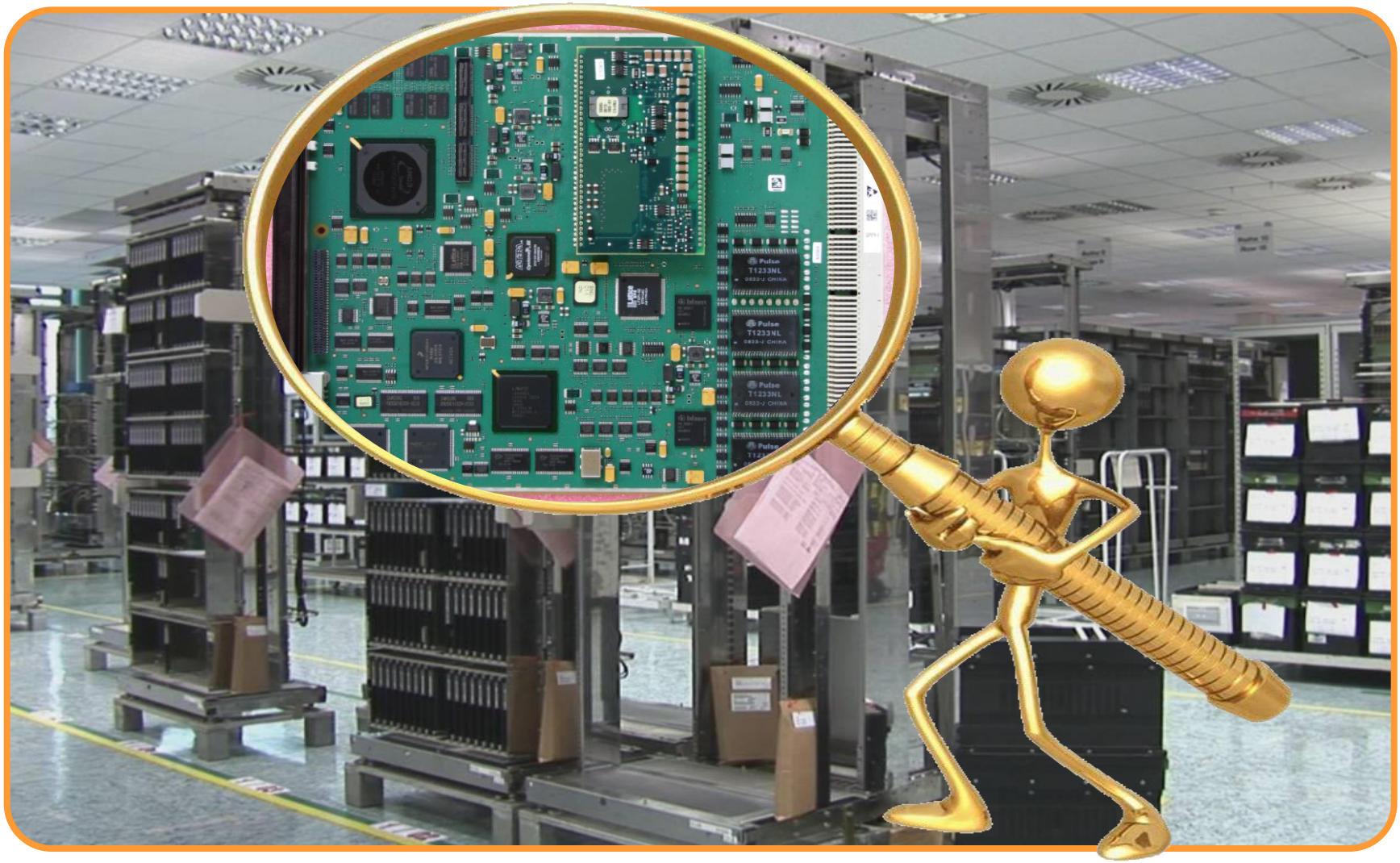
Altera FPGA Virtual JTAG Interface



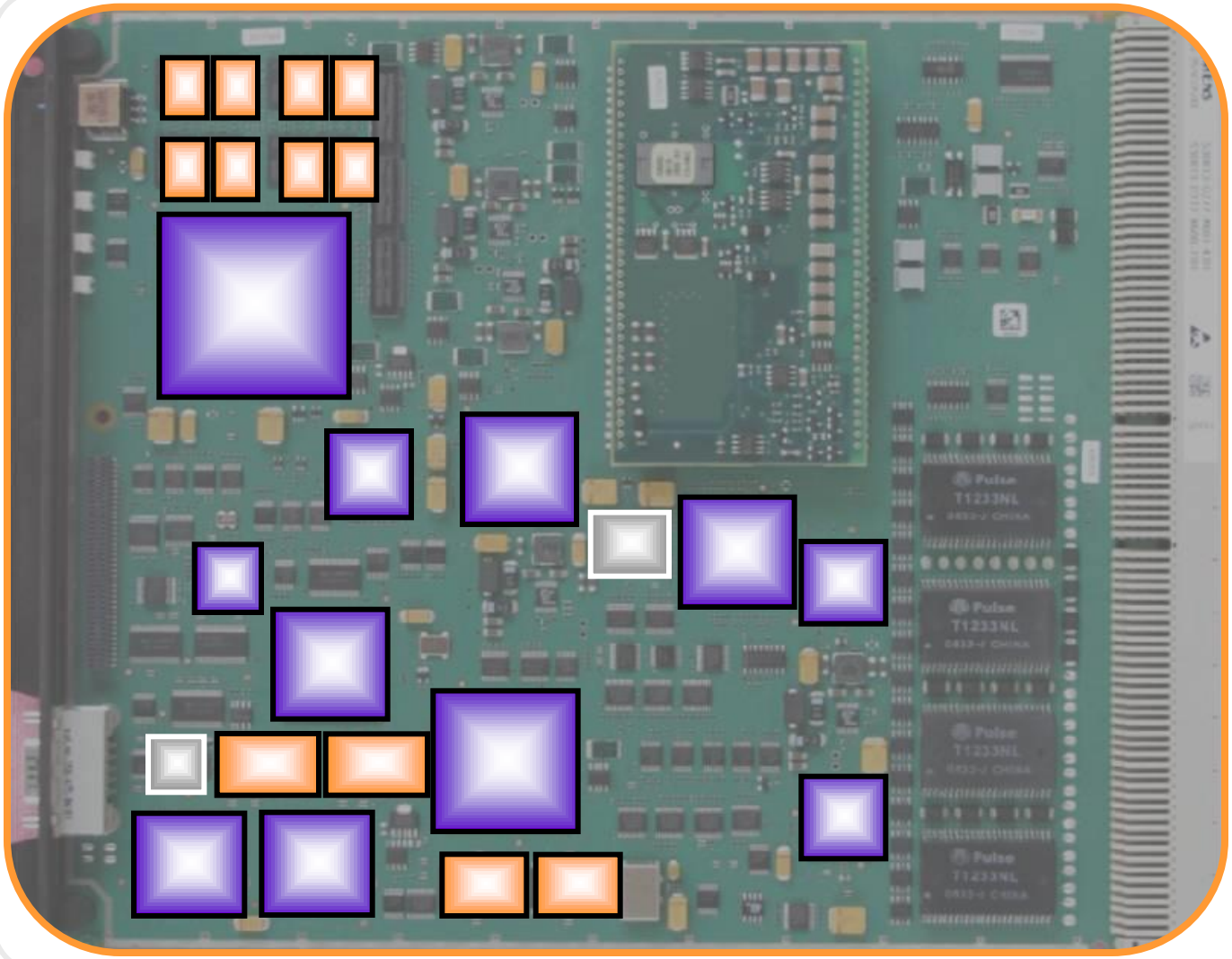
Anwendungsbeispiel(1)
Prüfling



Anwendungsbeispiel(2)
Prüfling



Anwendungsbeispiel(3)
Aufbau des Prüflings



2 * FLASH - Bausteine

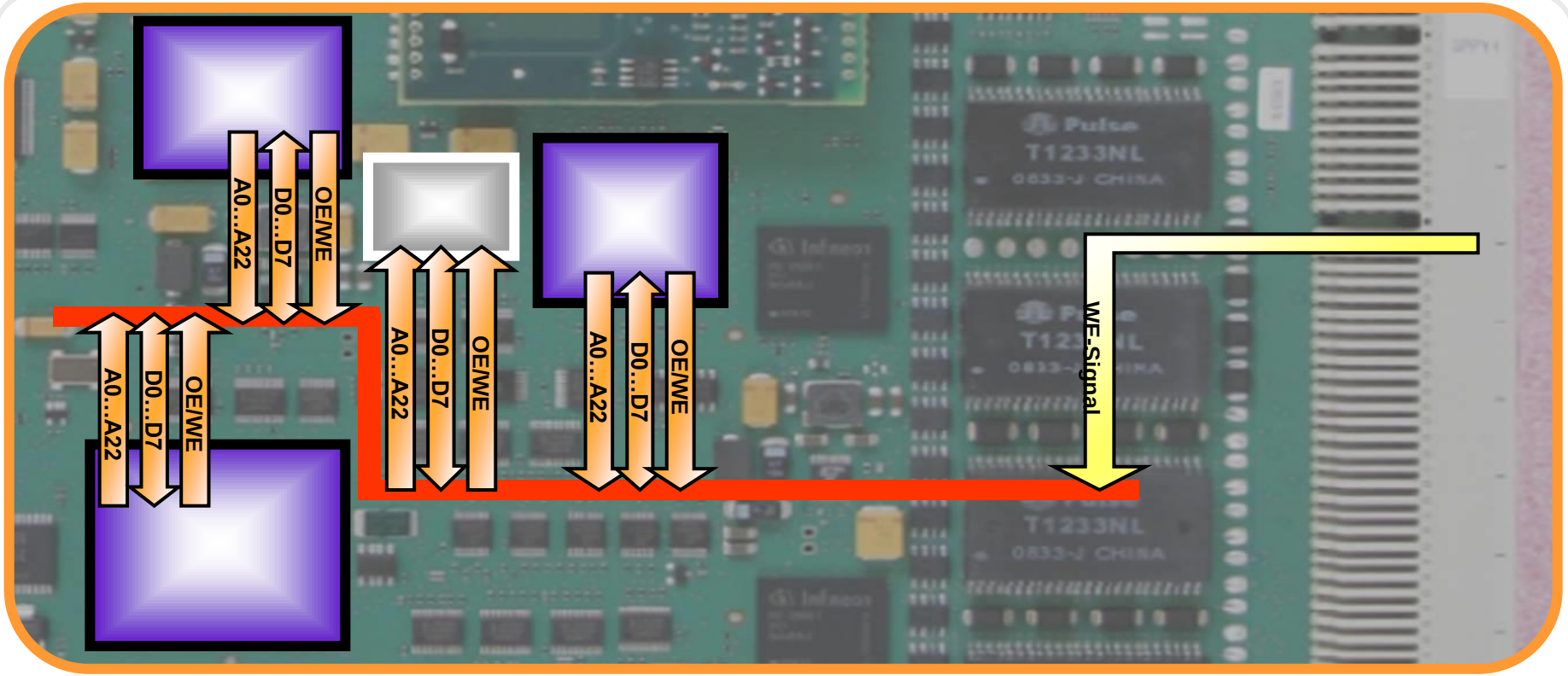
10 * RAM - Bausteine

11 * BSCAN - IC‘s

ca. 50 * Logic - Cluster

Anwendungsbeispiel(4)

Aufbau - Flash-Zugriffe



Die Ansteuerung des Flash ist über 3 BSCAN - Bauelemente möglich	Das WE-Signal ist über die Backplane zugreifbar !

Anwendungsbeispiel(5)

Besonderheit des Prüfling (1)

Der Flash
kann
vollständig
über einen FPGA
angesteuert werden

Einsatz für

Virtual JTAG

von ALTERA

Anwendungsbeispiel(5)

Besonderheit des Prüfling (2)

Der Flash kann vollständig über einen FPGA angesteuert werden

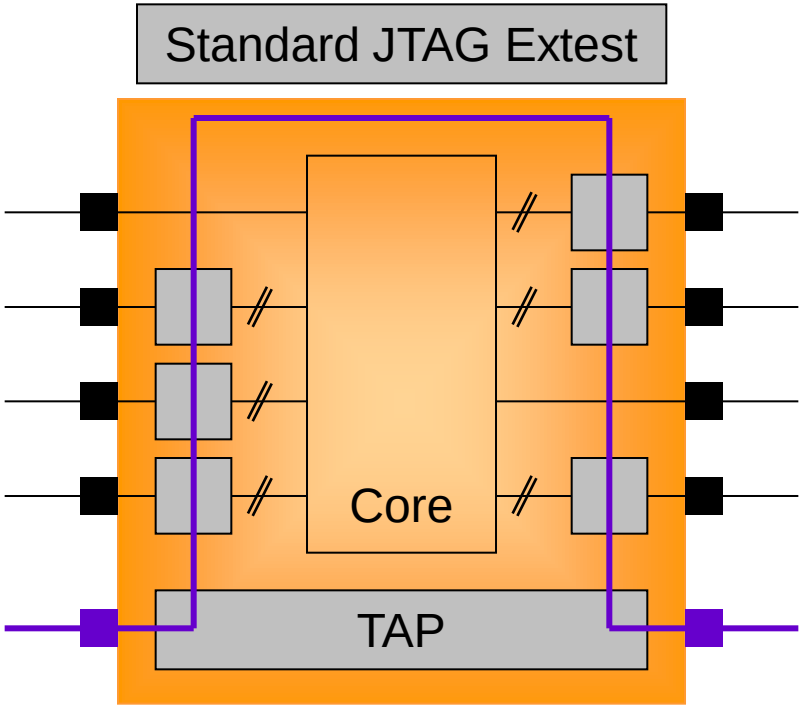
Einsatz für

Virtual JTAG

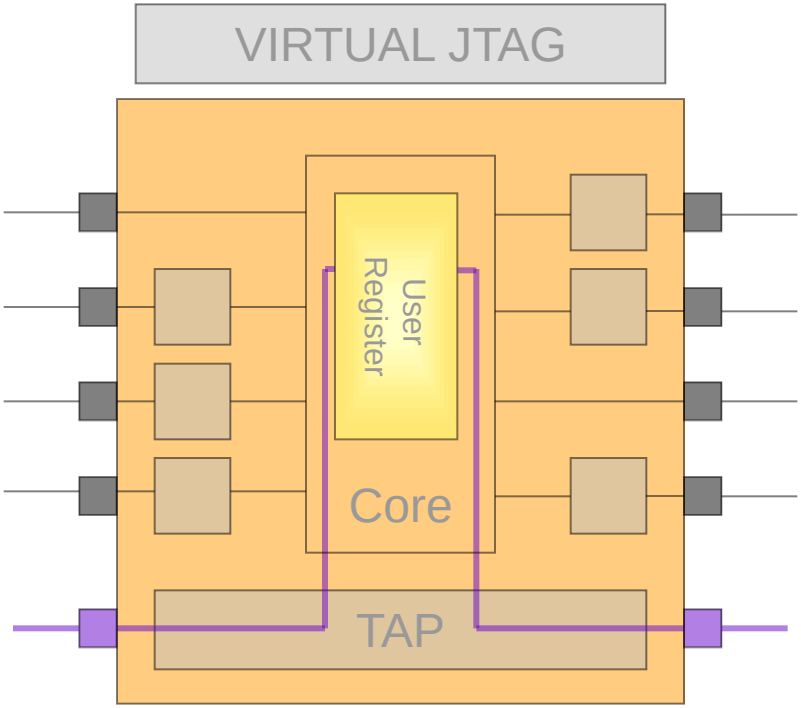
von ALTERA

Standard JTAG vs. Virtual JTAG?

Unterschied und Möglichkeiten (1)?



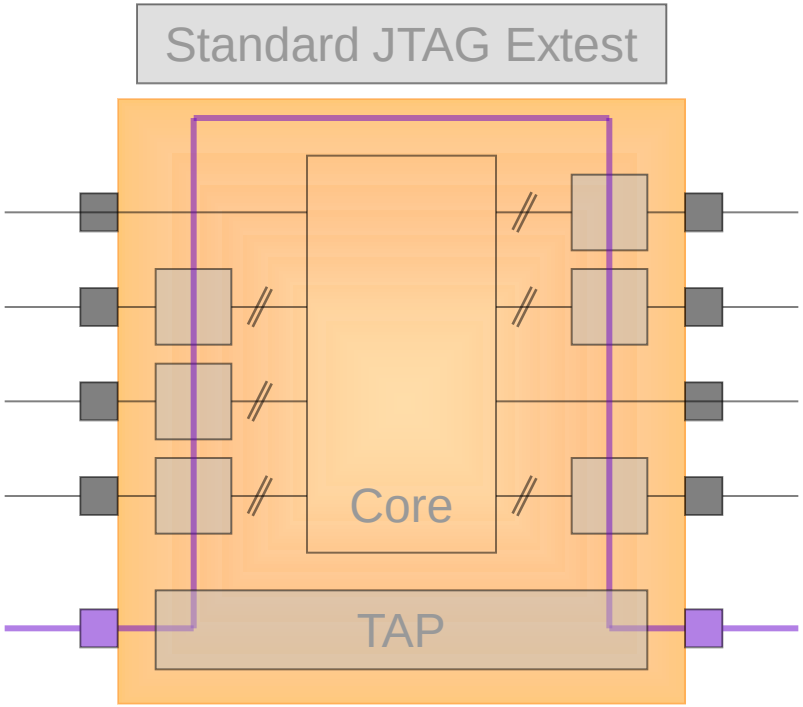
- **Core-Logik abgetrennt**
- **Aus- und Eingänge werden über BSCAN gesteuert**
- **Pins ohne BSCAN-Zelle können nicht kontrolliert werden**
- **Frequenz Pin = 1 / (TCK * BSCAN-Zellen + Overhead)**



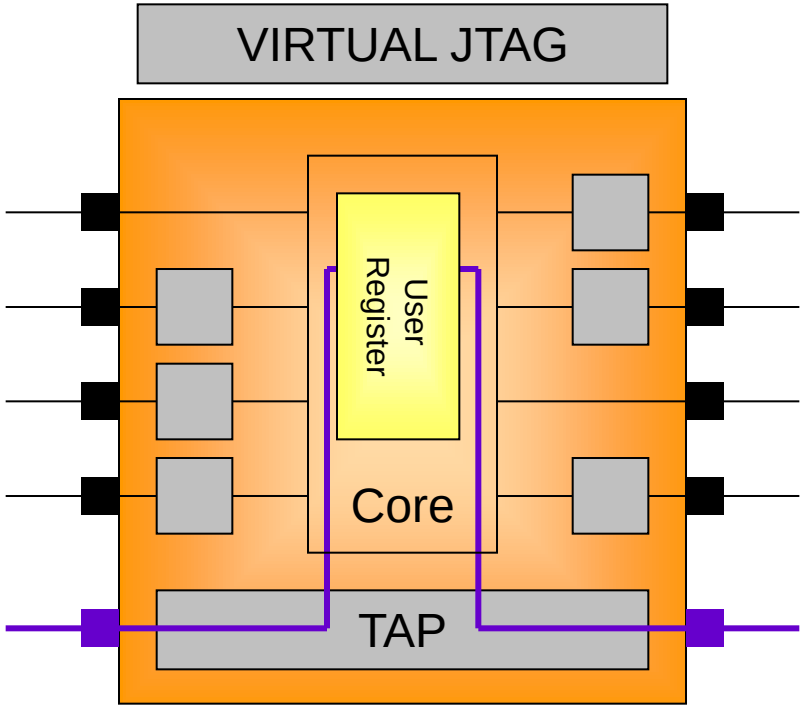
- **Direkter Zugriff auf Core-Logik möglich**
- **Aus – und Eingänge werden vom Core gesteuert**
- **Pins ohne BSCAN-Zelle können kontrolliert werden**
- **Frequenz Pin = TCK/2**

Standard JTAG vs. Virtual JTAG?

Unterschied und Möglichkeiten (2)?



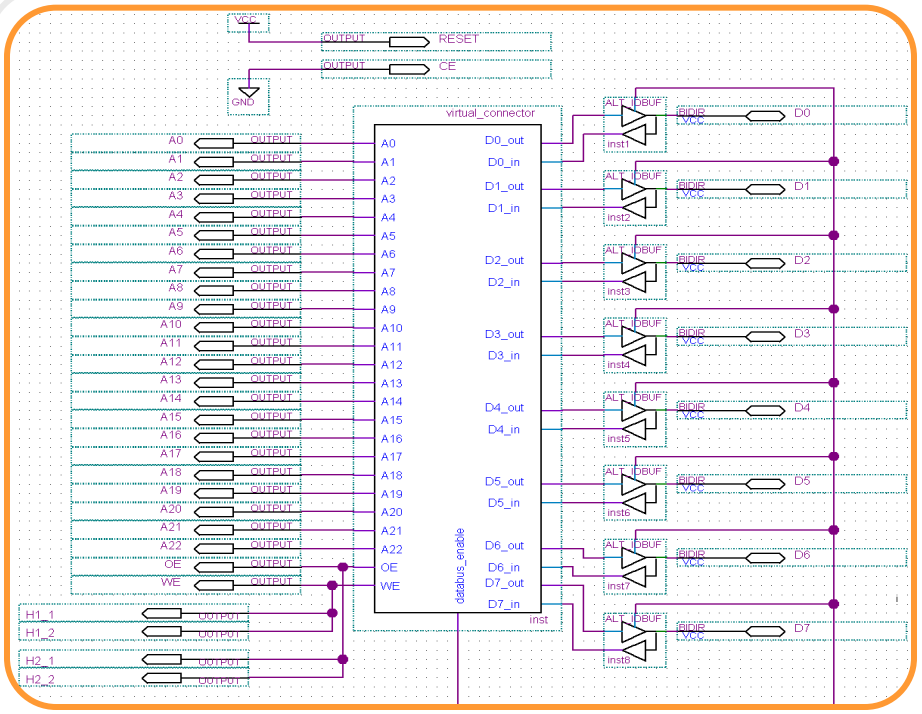
- Core-Logik abgetrennt
- Aus- und Eingänge werden über BSCAN gesteuert
- Pins ohne BSCAN-Zelle können nicht kontrolliert werden
- Frequenz Pin = 1 / (TCK * BSCAN-Zellen + Overhead)

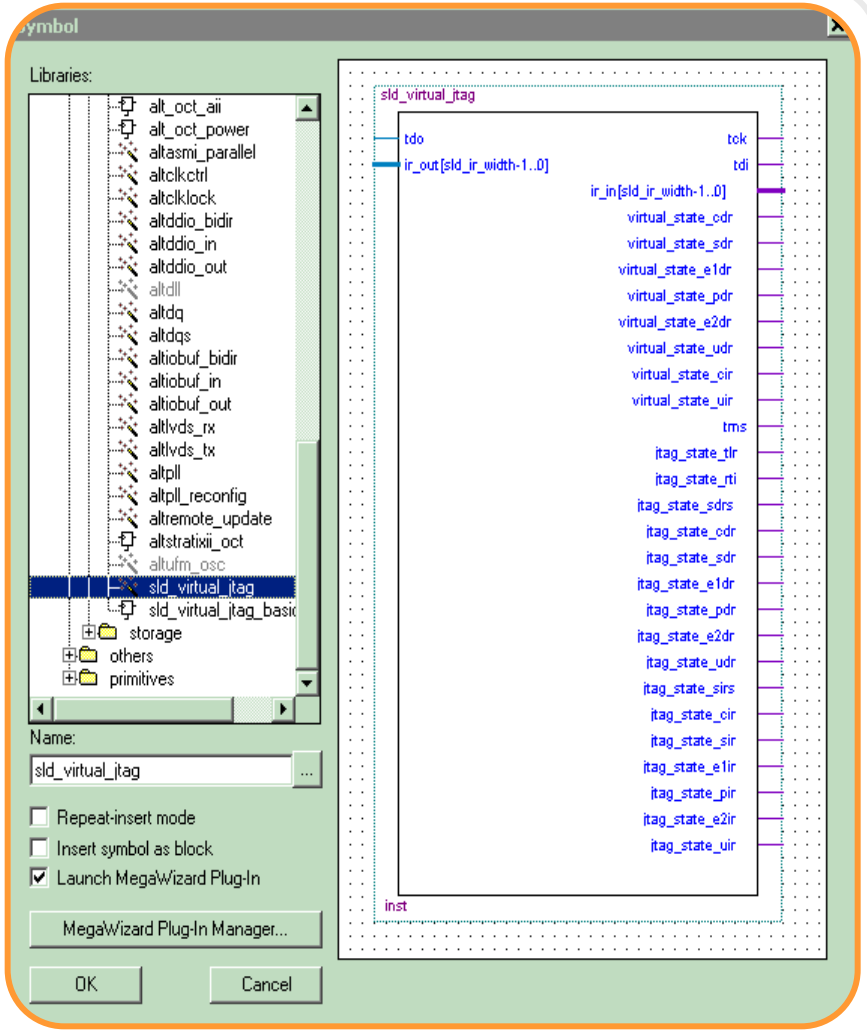


- **Direkter Zugriff auf Core-Logik möglich**
- **Aus – und Eingänge werden vom Core gesteuert**
- **Pins ohne BSCAN-Zelle können kontrolliert werden**
- **Frequenz Pin = TCK/2**

Umsetzung/Erstellung (1)

FPGA-Design Erstellung



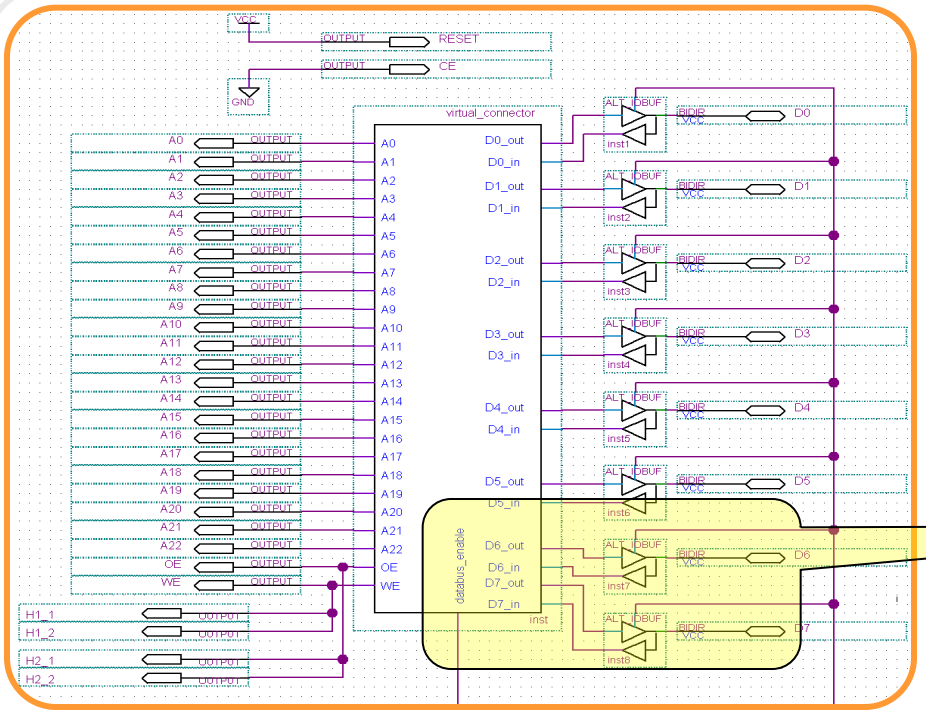


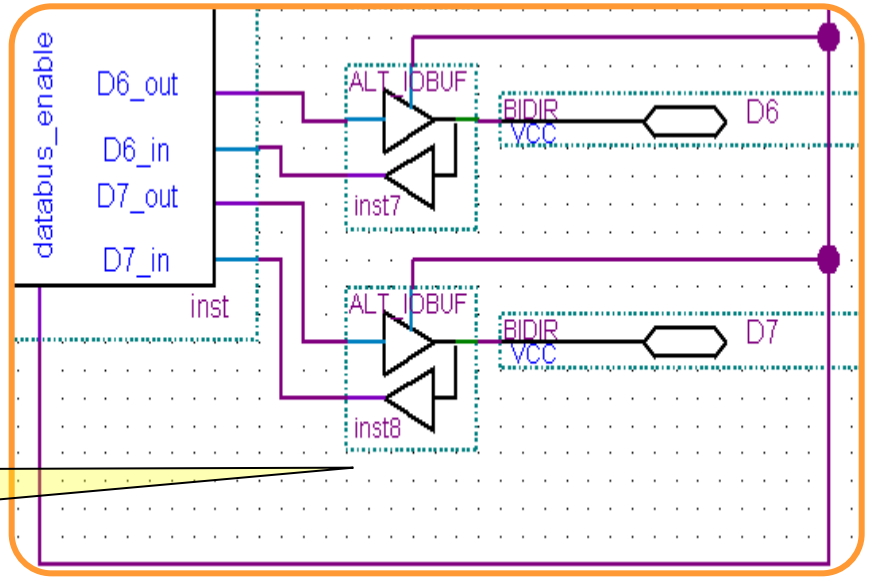
Megafunction / sld_virtual_jtag aus Altera Library

ermöglicht den direkten Zugriff auf den Core des Bauelementes über TDI/TDO/TMS/TCK/TRST

Umsetzung/Erstellung (2)

FPGA-Design Erstellung





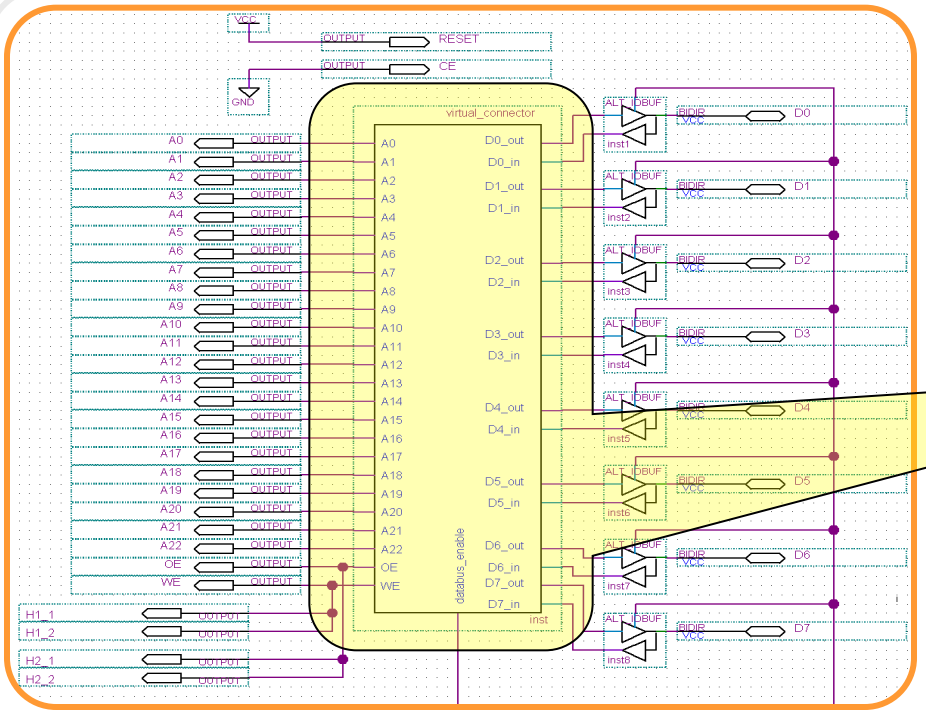
Nachbildung einer BSCAN-Zelle

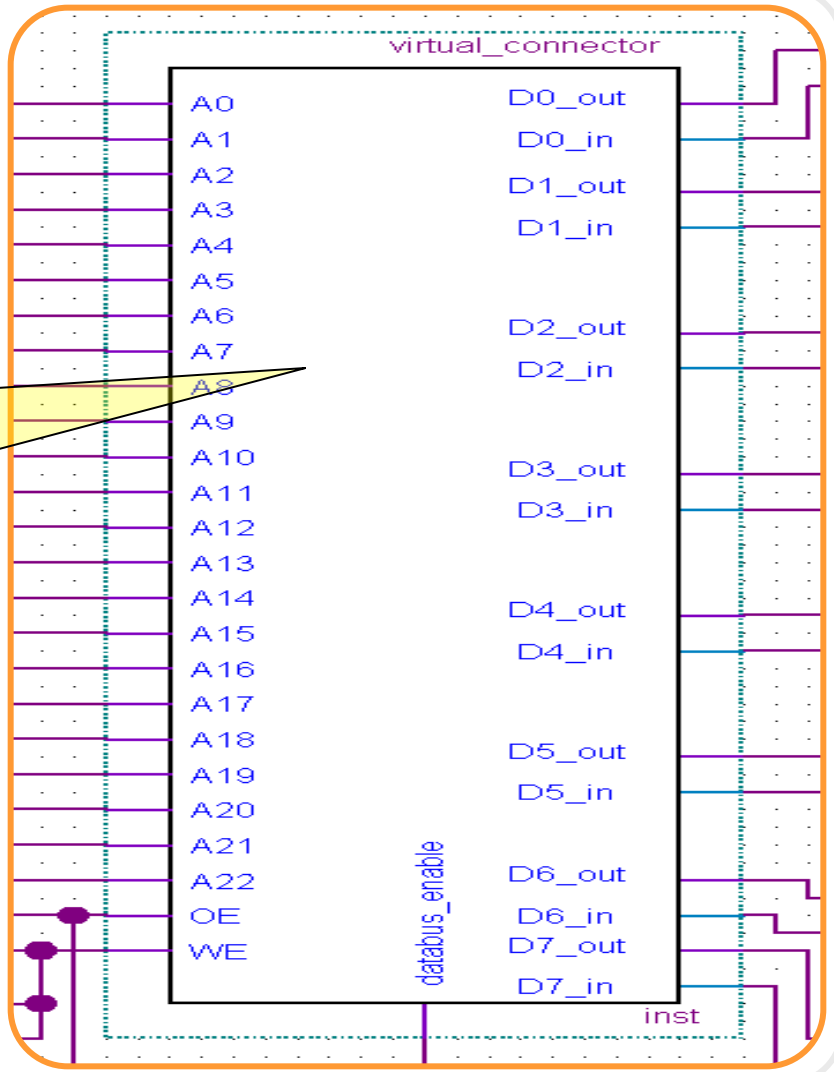
OUT und IN wurden im FPGA-Code getrennt beschrieben und über einen IO-Buffer geführt.

Ermöglicht das gleichzeitige Treiben und Messen der Pins

Umsetzung/Erstellung (3)

FPGA-Design Erstellung





Virtual Connector

Anwender spezifischer FPGA-Code zur Ansteuerung des Flashs.
Verilog HDL / VHDL ……

Umsetzung/Erstellung (4)

FPGA-Design Erstellung

```
1  module virtual_connector (A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,
2                            OE,WE,
3                            databus_enable,
4                            D0_in,  D1_in,  D2_in,  D3_in,  D4_in,  D5_in,  D6_in,  D7_in,
5                            D0_out, D1_out, D2_out, D3_out, D4_out, D5_out, D6_out, D7_out);
6
7  output      A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A17,A18,A19,A20,A21,A22;
8  output      D0_out,D1_out,D2_out,D3_out,D4_out,D5_out,D6_out,D7_out;
9  input       D0_in, D1_in, D2_in, D3_in, D4_in, D5_in, D6_in, D7_in;
10 output      databus_enable;
11 output      WE,OE;
```

Beschreibung der Aus- und Eingänge

```
68 reg         [2607:0] shift_register_2608;      //look table
69 //*************************************************************
```

Daten-word 1	Daten-word 2		Daten-word 319	Daten-word 320	End-Adresse	Start-Adresse	Ready bit	Start bit
D0..D7	D0..D7	....	D0..D7	D0..D7	A0..A22	A0..A22	1 bit	1 bit
0 7	8 15	....	2544 2551	2552 2559	2560 2582	2583 2605	2606	2607

Ein Virtual_**DR**SHIFT schreibt in dieses Register

```
79
80 wire READ_IDCODE               = ~ir_in[2] &&  ~ir_in[1] && ~ir_in[0]; // 000   --> READ_IDCODE
81 wire READ_FLASH_320_ADRESS     = ~ir_in[2] &&  ~ir_in[1] &&  ir_in[0]; // 001   --> READ_FLASH  (read max 320 addresses in one cylce)
82 wire ERASE_FLASH               = ~ir_in[2] &&   ir_in[1] && ~ir_in[0]; // 010   --> erase_FLASH
83 wire WRITE_FLASH_320_ADRESSES  = ~ir_in[2] &&   ir_in[1] &&  ir_in[0]; // 011   --> write_FLASH (write max 320 adresses in one cycle)
```

Ein Virtual_**IR**SHIFT schreibt in dieses Register

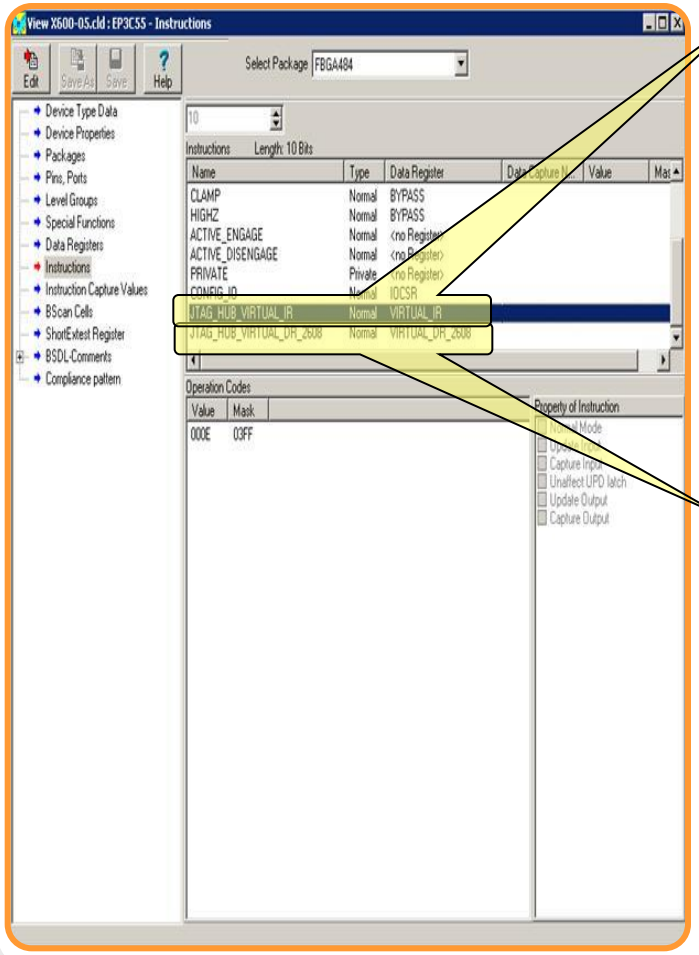
Umsetzung/Erstellung (4)

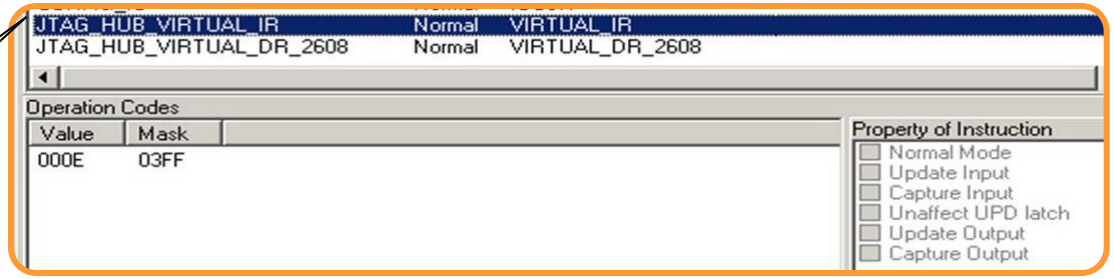
Beispiel Erase Flash - detail

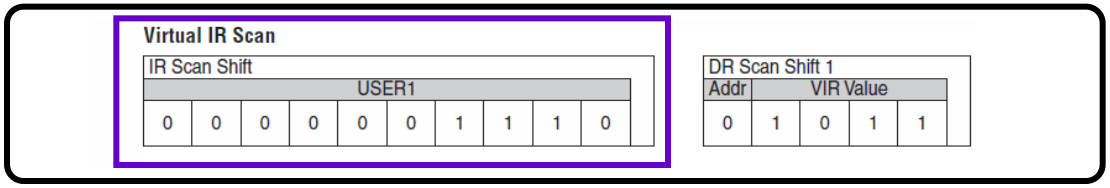
```
// ERASE_FLASH
 else if (ERASE_FLASH && cdr)       // capture dr
    begin
       shift_register_2608[2606] = ready;
    end
 else if (ERASE_FLASH && sdr )  //shift dr
    begin
       shift_register_2608 = {tdi, shift_register_2608[2607:1]};
    end
 else if (ERASE_FLASH && udr && start_command_2608)  //update dr
    begin
         begin
           ready = 0;                                                  // set ready = 0 --> run job
           run_task_nr = 0;                                            // set start run 0
         end
   end
 else if (ERASE_FLASH && ~ready)  //next line from job on next tck (runidle)
   begin
       run_task_nr = run_task_nr +1;

        if (run_task_nr == 1)  begin adress_reg <= 23'h0555; data_out_reg <= 8'hF0; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 2)  begin adress_reg <= 23'h0555; data_out_reg <= 8'hF0; oe_reg <= 1'b1;  we_reg <= 1'b0; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 3)  begin adress_reg <= 23'h0555; data_out_reg <= 8'hF0; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end

        if (run_task_nr == 4)  begin adress_reg <= 23'h0AAA; data_out_reg <= 8'hAA; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 5)  begin adress_reg <= 23'h0AAA; data_out_reg <= 8'hAA; oe_reg <= 1'b1;  we_reg <= 1'b0; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 6)  begin adress_reg <= 23'h0AAA; data_out_reg <= 8'hAA; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end

        if (run_task_nr == 7)  begin adress_reg <= 23'h0555; data_out_reg <= 8'h55; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 8)  begin adress_reg <= 23'h0555; data_out_reg <= 8'h55; oe_reg <= 1'b1;  we_reg <= 1'b0; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 9)  begin adress_reg <= 23'h0555; data_out_reg <= 8'h55; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end

        if (run_task_nr == 10) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'h80; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 11) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'h80; oe_reg <= 1'b1;  we_reg <= 1'b0; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 12) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'h80; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end

        if (run_task_nr == 13) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'hAA; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 14) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'hAA; oe_reg <= 1'b1;  we_reg <= 1'b0; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 15) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'hAA; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end

        if (run_task_nr == 16) begin adress_reg <= 23'h0555; data_out_reg <= 8'h55; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 17) begin adress_reg <= 23'h0555; data_out_reg <= 8'h55; oe_reg <= 1'b1;  we_reg <= 1'b0; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 18) begin adress_reg <= 23'h0555; data_out_reg <= 8'h55; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end

        if (run_task_nr == 19) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'h10; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 20) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'h10; oe_reg <= 1'b1;  we_reg <= 1'b0; databus_enable_reg <= 1'b1; end
        if (run_task_nr == 21) begin adress_reg <= 23'h0AAA; data_out_reg <= 8'h10; oe_reg <= 1'b1;  we_reg <= 1'b1; databus_enable_reg <= 1'b1; end

        if (run_task_nr == 22) begin adress_reg <= 23'h0000;                       oe_reg <= 1'b0;  we_reg <= 1'b1; databus_enable_reg <= 1'b0; end

        if (run_task_nr > 24)
           begin
               temp_register_8 = {D7_in,D6_in,D5_in,D4_in,D3_in,D2_in,D1_in,D0_in} ;
               if (temp_register_8 == 8'hFF)
                   begin
                       ready = 1; // finish all task done
                   end
           end
   end
```

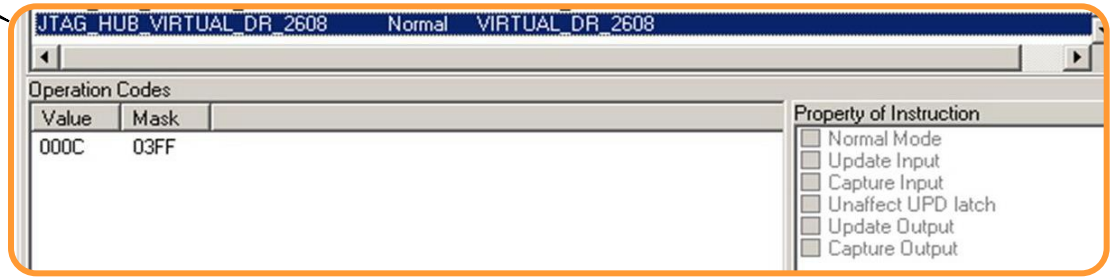
Umsetzung/Erstellung (5)
Einbindung in CASCON

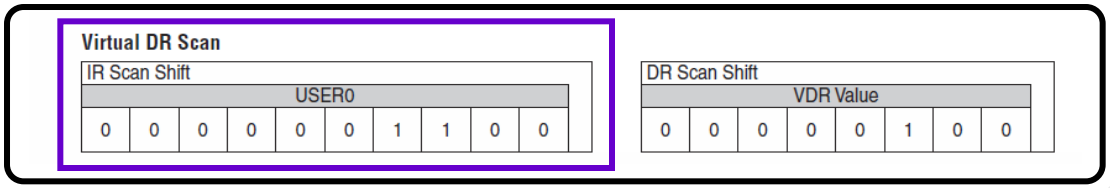
Instructions in Cascon Library einfügen





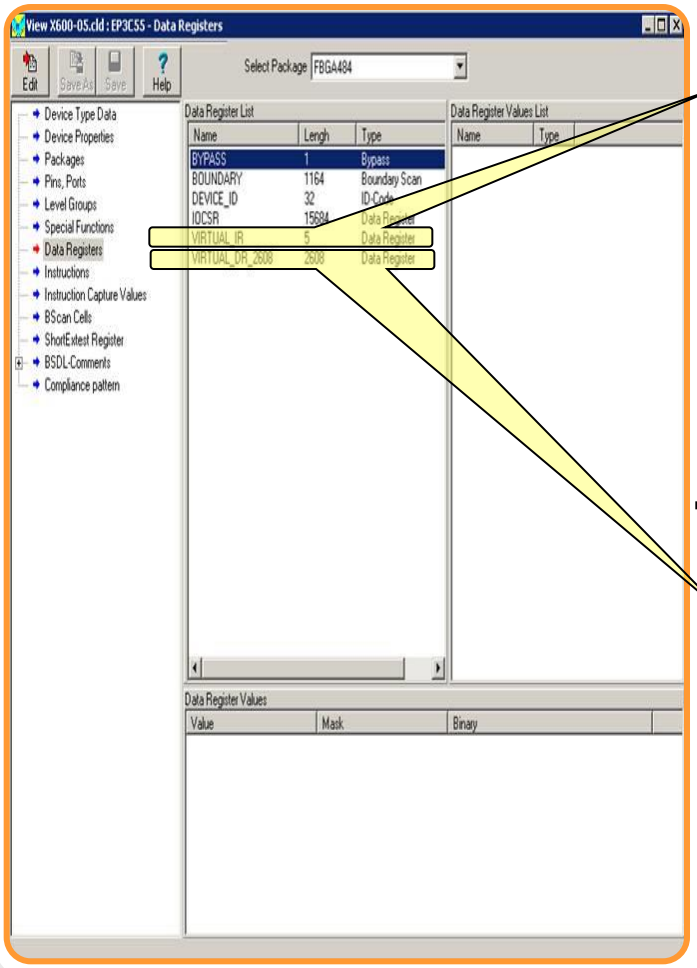






Umsetzung/Erstellung (6)
Einbindung in CASCON

Data Register in Cascon Library einfügen



VIRTUAL_IR 5 Data Register

IR Formel = (Länge IR-Register) + Adresse
IR Beispiel = (3) 4 + 1

[Wenn IR kleiner 4, muss 4 verwendet werden]

Virtual IR Scan

IR Scan Shift									
USER1									
0	0	0	0	0	0	1	1	1	0

DR Scan Shift 1				
Addr	VIR Value			
0	1	0	1	1

VIRTUAL_DR_2608 2608 Data Register

DR Formel = (Länge DR-Register)
DR Beispiel = (0-2067) 2608

Virtual DR Scan

IR Scan Shift									
USER0									
0	0	0	0	0	0	1	1	0	0

DR Scan Shift							
VDR Value							
0	0	0	0	0	1	0	0

Umsetzung/Erstellung (7)
Aufruf in CASCON

Ansteuerung des FPGA Codes aus Caslan

```
ldi D3, JTAG_HUB_VIRTUAL_IR;
irshift;
……
ld D3, VIRTUAL_IR, temp_var_5;
drshift;
```

Laden einer Instruktion in den FPGA

```
ldi D3, JTAG_HUB_VIRTUAL_DR_2608;
irshift;
……..
ld D3:VIRTUAL_DR_2608,temp_var_2608;
drshift;
```

Ein Register des FPGA‘s beschreiben

```
ldi D3, JTAG_HUB_VIRTUAL_DR_2608;
irshift;
……..
drshift;
ld temp_var_2608, D3:VIRTUAL_DR_2608;
```

Ein Register des FPGA‘s lesen

Umsetzung/Erstellung (8)

Beispiel Erase Flash - detail

```
PROC READ_STATUS_AND_WAIT_FINISH_2608_for_erase;
  begin
      timeout := 0;
     CLOCK RUNIDLE, 25;
     wait 2000;
     ld D3:VIRTUAL_DR_2608,0;drshift;
     ld SHIFT_REGISTER_2608, D3:VIRTUAL_DR_2608/m;
     ror SHIFT_REGISTER_2608,2606;
     ld ready, SHIFT_REGISTER_2608;
     while (!ready) do
         write ('!');
         wait 2000;
         CLOCK RUNIDLE, 25
         drshift;
         ld SHIFT_REGISTER_2608, D3:VIRTUAL_DR_2608/m;
         ror SHIFT_REGISTER_2608,2606;
         ld ready, SHIFT_REGISTER_2608;
         timeout := timeout +1;
         if timeout > 60 then
               writeln ('ERROR = TIMEOUT');
               STOP 300;
         end;
     end;
  end;
```

```
PROC ERASE;
    begin
        ldi D3, JTAG_HUB_VIRTUAL_IR;
        irshift;
        ld VIRTUAL_IR, 11010b;
        drshift;
        ldi D3, JTAG_HUB_VIRTUAL_DR_2608;
        irshift;
        ------------------------------------------------------------------
        and temp_var_2608, 0;
        or  temp_var_2608, 1;
        rol temp_var_2608,  2607;
        ----------------------------------------------------------
        ld D3:VIRTUAL_DR_2608,temp_var_2608;
        drshift;
        CALL READ_STATUS_AND_WAIT_FINISH_2608_for_erase;
        ----------------------------------------------------------
        writeln ('');
    end;
```

Fragen und Diskussion

jetzt aber auch später

Nokia Siemens Networks GmbH @ Co. KG
Werner-von-Siemens-Strasse 2-6
76646 Bruchsal

Tobias Dammert
Test Engineer
tobias.dammert@nsn.com
Phone +49 7251 73 3592

Lars Knüppel
Test Engineer
lars.knueppel@nsn.com
Phone +49 7251 73 4696