

# FPGA Systementwurf

---

Rosbeh Etemadi

Paderborn Center for Parallel Computing  
Paderborn University

29. Mai 2007



PADERBORN  
CENTER FOR  
PARALLEL  
COMPUTING

1. FPGAs
2. Entwicklungssprache VHDL
3. Matlab/Simulink
4. Entwicklungssprache Handel-C
5. Fazit

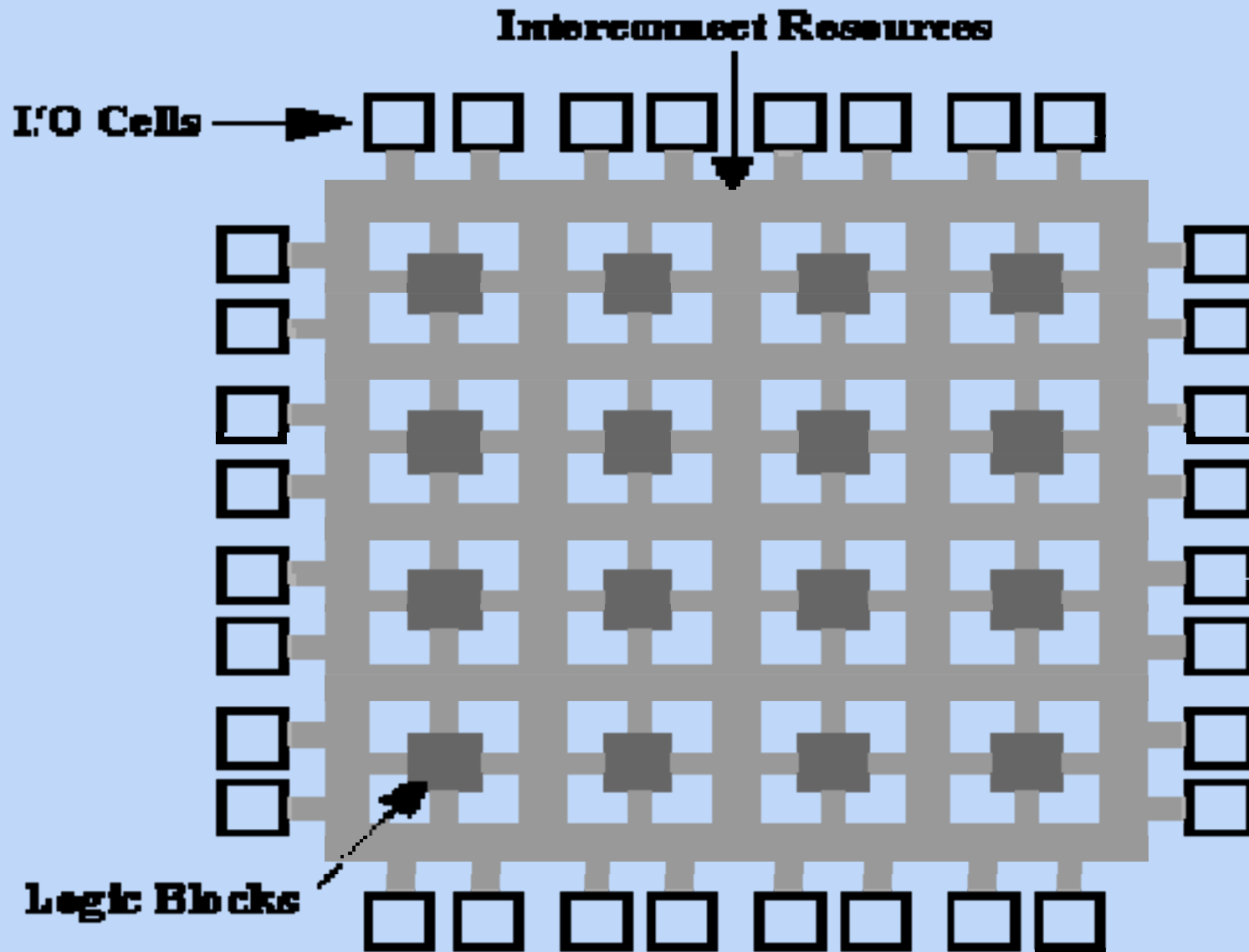
1. FPGAs
  - ▶ Grundsächliche Betrachtung
  - ▶ Was kann realisiert werden
  - ▶ Struktureller Aufbau
  - ▶ Entwicklungsablauf
2. Entwicklungssprache VHDL
3. Matlab/Simulink
4. Entwicklungssprache Handel-C
5. Fazit

- ▶ (**F**ield **P**rogrammable **G**ate **A**rray)
- ▶ Programmierbarer Halbleiterbaustein
- ▶ Programmierbare I/O-Blöcke z.B. für IN/OUT/Bidirektional
- ▶ Besitzen programmierbare logische Blöcke (CLB: Configurable Logic Block) und programmierbare Verbindungen zwischen diesen Blöcken
- ▶ CLB besteht aus look-up tables (LUTs)
- ▶ Werden programmiert durch Setzen von Schaltern
- ▶ Unterscheidung von Schalter Technologien
  - ▶ In einmalig brennbare Schalter und (Anti-Fuse)
  - ▶ Re-programmierbare (SRAM)

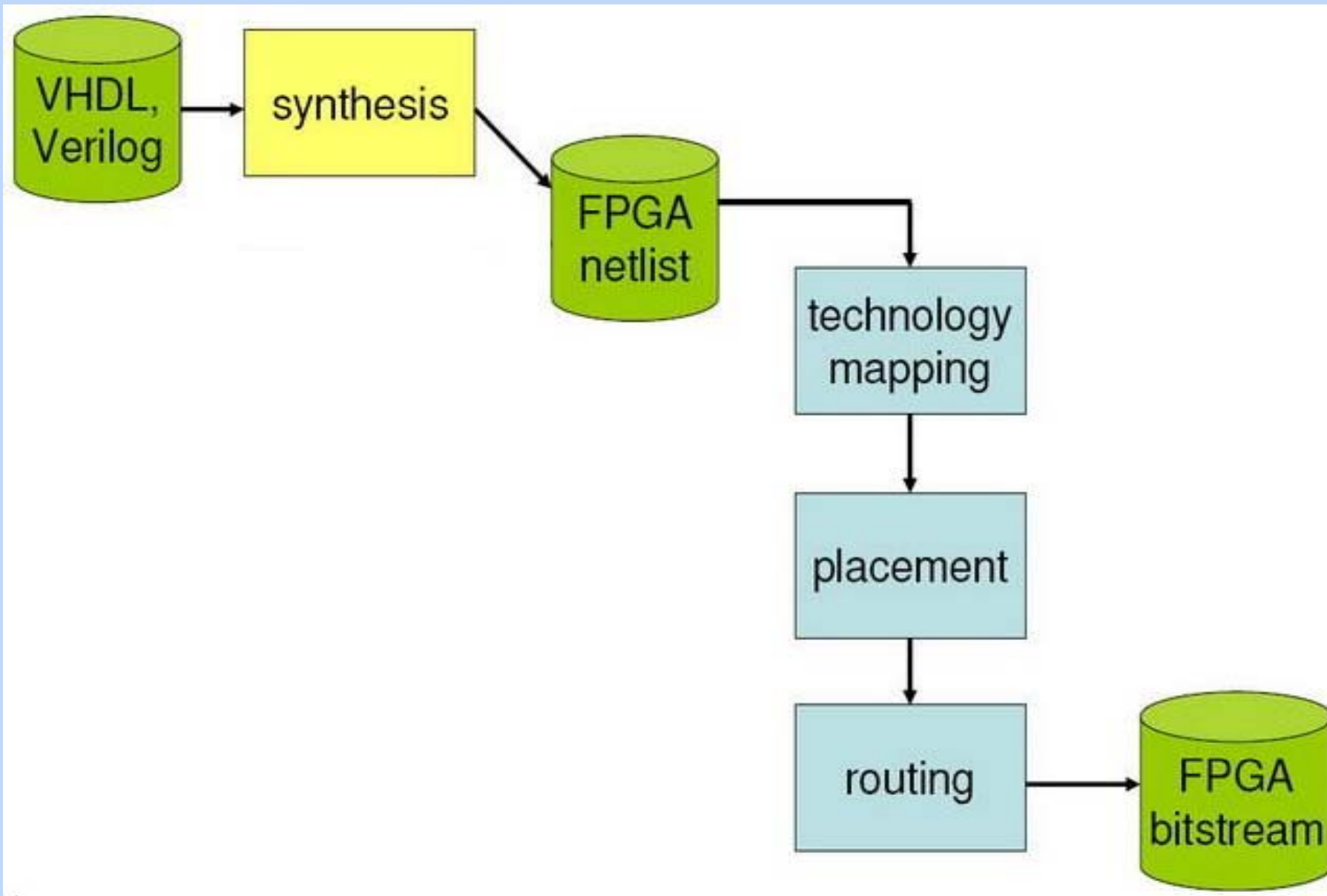
# Was wird mit FPGAs realisiert

- ▶ logische Bausteine wie AND, OR, NOT, NOR und NAND
- ▶ Komplexe Logik wie Decoder/Encoder
- ▶ Mathematische Funktionen

# Struktureller Aufbau eines FPGAs



# Entwicklungsablauf eines FPGAs

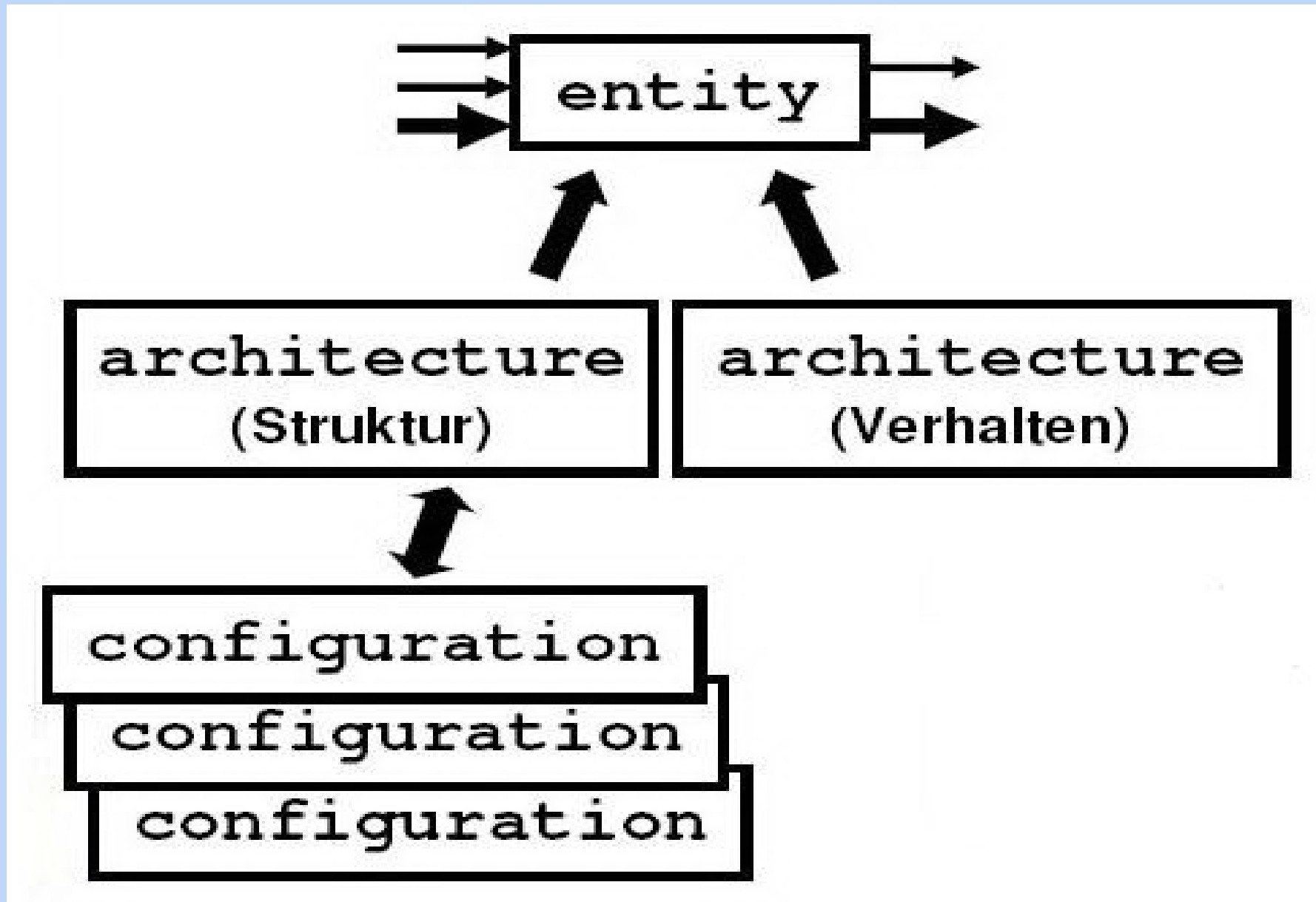


1. FPGAs
2. Entwicklungssprache VHDL
  - ▶ Grundlagen VHDL
  - ▶ VHDL Spezifikation
  - ▶ VHDL Beispielcode
  - ▶ Entwicklungsumgebung Xilinx ISE
  - ▶ Ablaufdiagramm von FPGA-Synthese mit ISE
3. Matlab/Simulink
4. Entwicklungssprache Handel-C
5. Fazit



- ▶ Definition VHDL
  - ▶ Very High Speed Integrated Circuit Hardware Description Language
  - ▶ Ist eine Hardwarebeschreibungssprache
  - ▶ Oft verwendete Sprache (in Europa)
  - ▶ Wird benutzt um komplizierte digitale Systeme zu beschreiben
- ▶ Grundkonzepte von VHDL
  - ▶ Abstraktion von spezifischer Hardware
  - ▶ Hierarchisch / Modular
  - ▶ Bibliothek von einzelnen Modulen

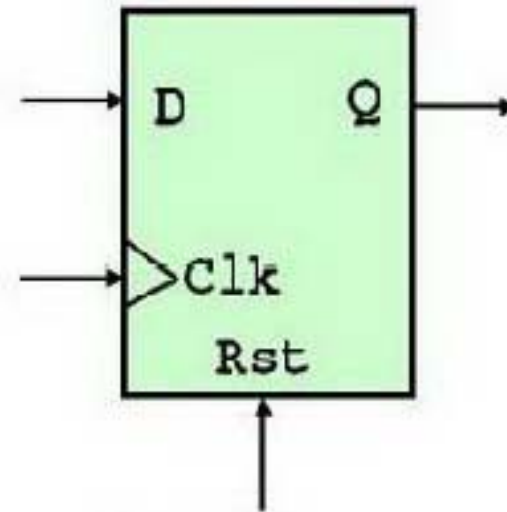
# Aufbau einer VHDL Spezifikation



# VHDL Beispielcode (D Flip-Flop)

```
entity DFF is
  port ( D, Clk, Reset : in  std_logic;
         Q              : out std_logic );
end entity DFF;
```

```
architecture Behavioral of DFF is
begin
  process (Clk, Reset)
  begin
    if (Reset = '0') then
      Q <= '0';
    elsif (Clk'event and Clk ='1') then
      Q <= D;
    end if;
  end process;
end architecture Behavioral;
```



# Entwicklungsumgebung Xilinx ISE

The screenshot displays the Xilinx ISE development environment. The main window shows a VHDL code editor with the following code:

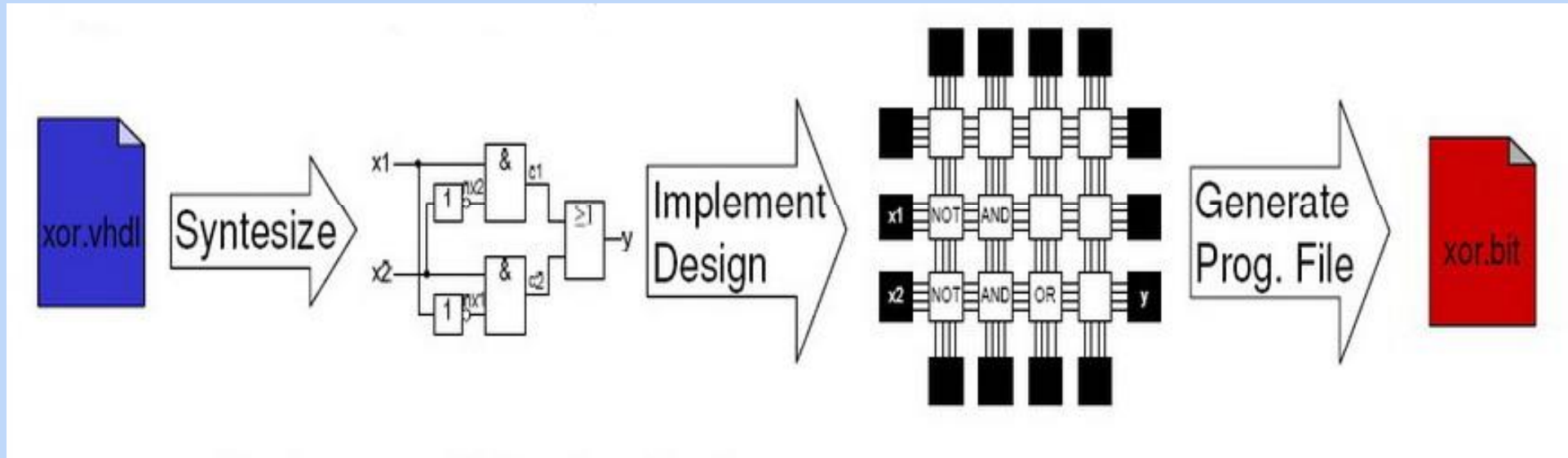
```
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 -----
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ----- Uncomment the following library declaration if instantiating
26 ----- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity counter is
31     Port ( CLOCK : in STD_LOGIC;
32           DIRECTION : in STD_LOGIC;
33           COUNT_OUT : out STD_LOGIC_VECTOR (3 downto 0));
34 end counter;
35
36 architecture Behavioral of counter is
37     signal count_int : std_logic_vector(3 downto 0) := "0000";
38     begin
39     process (CLOCK)
40     begin
41         if CLOCK='1' and CLOCK'event then
42             if DIRECTION='1' then
43                 count_int <= count_int + 1;
44             else
45                 count_int <= count_int - 1;
46             end if;
47         end if;
48     end process;
49
50     COUNT_OUT <= count_int;
51
52 end Behavioral;
```

The console window at the bottom shows the following output:

```
Number of warnings: 0
Total time: 3 secs

Process "Generate Post-Place & Route Static Timing" completed successfully
```

The status bar at the bottom indicates "Ready" and "Ln 48 Col 16 | CAPS NUM SCRS VHDL".



- ▶ Plattform-unabhängige Synthese
- ▶ Implementierung (Low-Level Synthese) für den speziellen FPGA
- ▶ Erzeugung des Bit-Files für den FPGA

1. FPGAs
2. Entwicklungssprache VHDL
3. Matlab/Simulink
  - ▶ Was ist Matlab
  - ▶ Was ist Simulink
  - ▶ Entwicklungsumgebung Matlab/Simulink
  - ▶ Simulink Beispielmodell
4. Entwicklungssprache Handel-C
5. Fazit

- ▶ Ist eine kommerzielle plattformunabhängige Software der Firma The MathWorks
- ▶ Wir verwendet um mathematische Probleme zu lösen und Ergebnisse grafisch darzustellen
- ▶ verfügt über einen Satz von speziellen Befehlen
- ▶ Speziell entwickelt zur numerischen Berechnung von Problemen
- ▶ Probleme Gebiet sind linearen Algebra, der numerischen Analysis sowie Simulation

- ▶ Simulink stellt eine spezielle Toolbox von Matlab dar
- ▶ Simulink erweitert Matlab zur Modellierung und Simulation von Systemen
- ▶ Eine graphische Zusammenstellung von Modellen die vorgefertigt oder selbst zu definierend sind
- ▶ Speziell von Xilinx eine Bibliothek von Bausteinen für Hardware Entwicklung
- ▶ Mit dem System Generator Hardwarebeschreibungssprache erzeugen



# Entwicklungsumgebung Matlab/Simulink

The screenshot displays the Simulink development environment. On the left, the Simulink Library Browser is open, showing a tree view of various toolboxes. The 'Xilinx Blockset' is expanded, and the 'Control Logic' sub-category is selected. A list of blocks is shown on the right, including 'Black Box', 'Constant', 'Counter', 'Dual Port RAM', 'EDK Processor', 'Expression', 'FIFO', 'Inverter', 'Logical', 'MCode', 'Mux', 'PicoBlaze Microcontroller', 'Register', 'Relational', 'ROM', 'Shift', 'Single Port RAM', and 'Slice'. The 'Logical' block is highlighted.

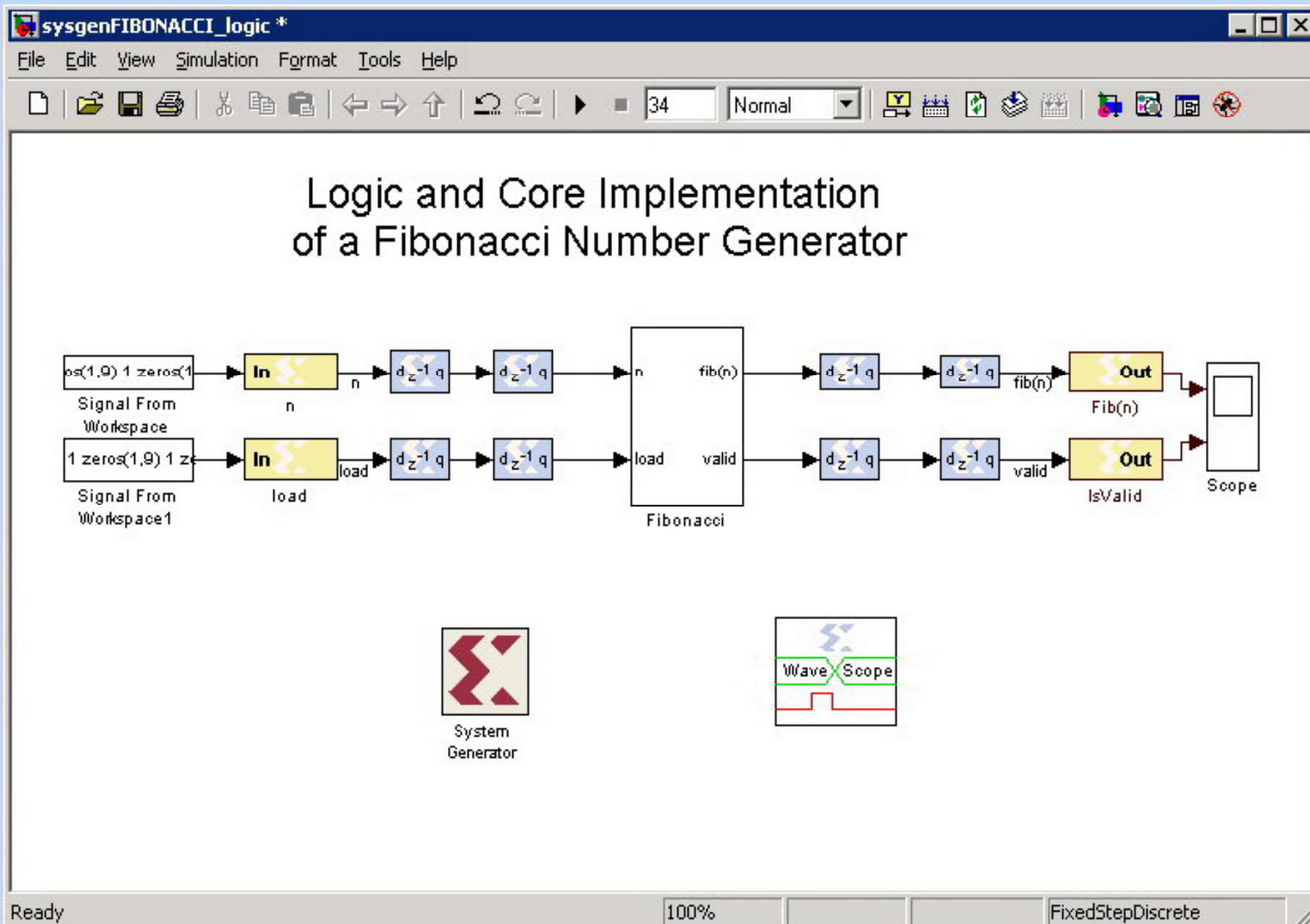
The main workspace shows a Simulink block diagram. The diagram consists of the following blocks and connections:

- State-Space** block with equations  $x' = Ax + Bu$  and  $y = Cx + Du$ .
- First Order Hold** block.
- Ramp** block.
- Gateway In** block.
- Gateway In1** block.
- Logical** block (the selected block from the library browser).
- Gateway Out** block.
- Integrator** block.

The signal flow is as follows: The output of the State-Space block goes to the First Order Hold block. The output of the First Order Hold block goes to the Ramp block. The output of the Ramp block goes to the Gateway In block. The output of the Gateway In block goes to the Logical block. The output of the Logical block goes to the Gateway Out block. The output of the Gateway Out block goes to the Integrator block. The output of the Integrator block goes back to the State-Space block.

The status bar at the bottom of the Simulink window shows 'Ready', '100%', and 'ode45'. Below the status bar, the text 'for entity counter1\_tbw, architecture testbench\_arch' is visible.

# Simulink Beispielmodell



1. FPGAs
2. Entwicklungssprache VHDL
3. Matlab/Simulink
4. Entwicklungssprache Handel-C
  - ▶ Was ist Handel-C
  - ▶ Warum wurde Handel-C entwickelt
  - ▶ Beispiel Erweiterungen
  - ▶ Handel-C Beispielcode
  - ▶ Handel-C Entwurfsablauf
5. Fazit

- ▶ Entwicklung von Handel-C begann Mitte der 90er an Oxford-University
- ▶ Ist eine Programmiersprache, die dazu dient, hoch abstrahierte Algorithmen direkt in Hardware zu übersetzen
- ▶ Ist von C abgeleitete Programmiersprache zur Spezifikation von Signalverarbeitungsroutinen

# Warum wurde Handel-C entwickelt

- ▶ Entwicklung von Programmen für FPGAs Vereinfachen
- ▶ Wiederverwendung in C entwickelter Algorithmen
- ▶ Richtet sich eher an Software- als Hardwareentwickler
- ▶ Muss daher aller Hardware-relevanten Probleme verstecken

- ▶ Es gibt nur ein Datentyp
- ▶ Konstrukte wie PAR oder Signalverzögerung
- ▶ CHANIN und CHANOUT für input/output
- ▶ Es existieren keine Pointer in Handel-C

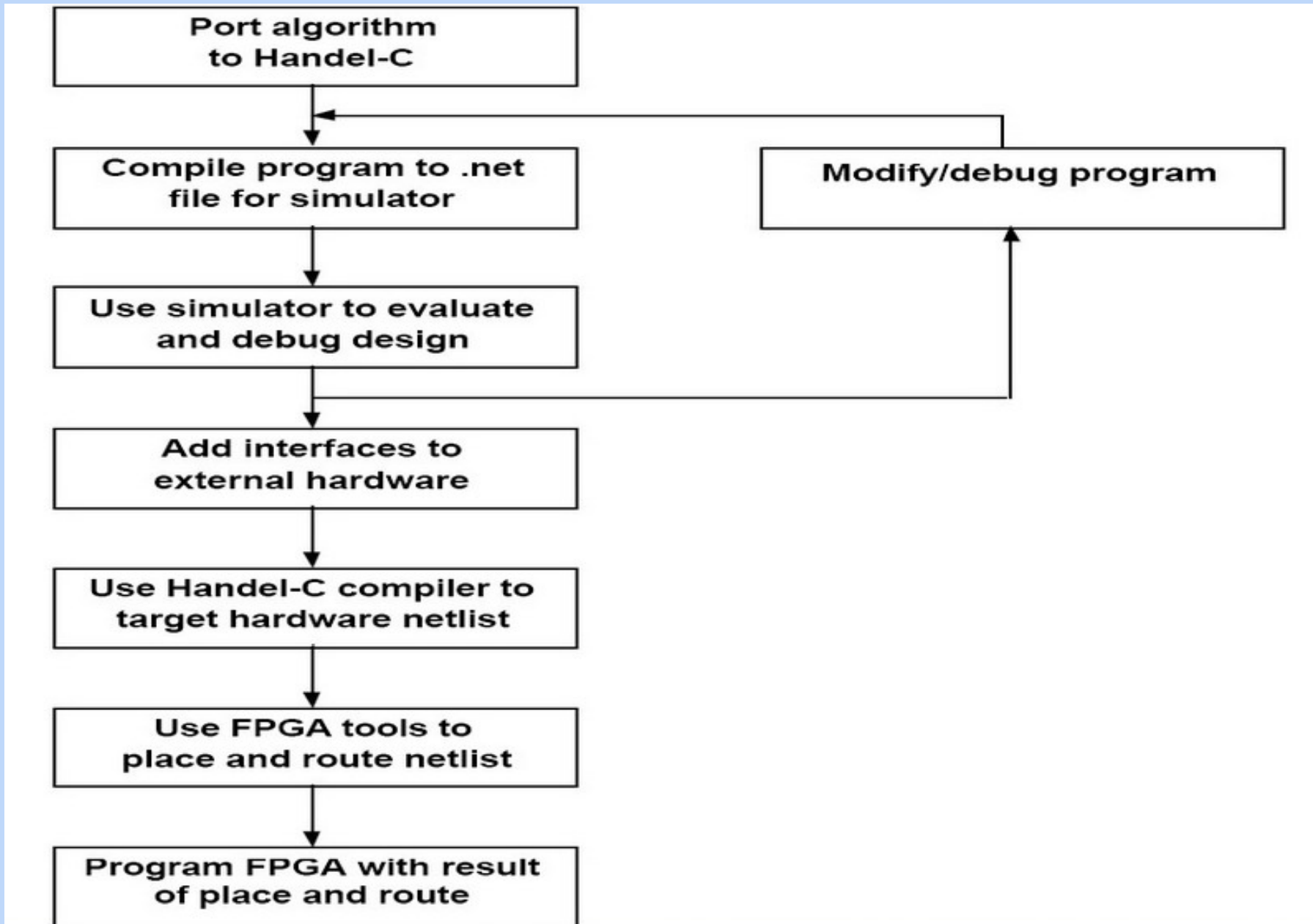
# Handel-C Beispielcode (Summierer)

```
void main(void) {
    unsigned int 16 sum;
    unsigned int 8 data;
    chanin input;
    chanout output;

    sum = 0;
    do {
        input ? data;
        sum = sum + (0 @ data);
    } while (data != 0);

    output ! sum;
}
```

# Handel-C Entwurfsablauf





|                       | VHDL                               | Matlab/Simulink  | Handel-C   |
|-----------------------|------------------------------------|--|--|
| <b>Vorteile</b>       | Beschreibt Hardware detailliert    | Code ist meist eher synthesefähig                          | Benötigt keine hohes Hardwareverständnis                   |
|                       | Ermöglicht hohe Optimierung        | Sehr übersichtlich da Schaltungen Visuell angezeigt werden | Schnelle Entwicklung bei kleinen Projekten möglich         |
|                       |                                    | Ermöglicht schnelles Prototyping                           |  |
| <b>Nachteile</b>      | Erfordert Hardwareverständnis      | Muss nicht optimalster VHDL Code sein                      | Bei großen Projekten sind meist Netzlisten nicht umsetzbar |
|                       | Code ist nicht immer synthesefähig | Fehlerbehandlung kann schnell kompliziert werden           | Nachträglich Optimierung sehr aufwendig                    |
| <b>Modelumsetzung</b> | Programmcode                       | Grafisch   | Programmcode   |