Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

HARDWARE-BESCHREIBUNGSSPRACHEN

Hardwareentwurf mit VHDL

21. Oktober 2021
Revision: b941727 (2021-01-16 01:57:51 +0100)

Steffen Reith

Theoretische Informatik
Studienbereich Angewandte Informatik
Hochschule **RheinMain**

AUTOMATEN MIT DATENPFAD

Notizen

Notizen

AUTOMATEN MIT DATENPFAD

Bisher haben die betrachteten Automaten nur **Ausgabesignale erzeugt**. Nun soll dieses Konzept zur **Steuerung** von **komplexe(re)n Komponenten** verwendet werden.

Definition (Datenpfad)

Zustandsspeicher, Schaltkreise für Berechnungen und deren Verbindungen, die zusammen den Fluss und die Umwandlung von Daten ermöglichen, werden **Datenpfad** (engl. data path) genannt.

Definition (Kontrollpfad)

Ein (endlicher) Automat, der die Komponenten eines Datenpfads sinnvoll steuert und so eine zielgerichtete Berechnung ermöglicht heißt **Kontrollpfad** (engl. control path).

Notizen

DIE REGISTER-TRANSFER METHODE

Mit Hilfe von Automaten mit Datenpfad können sequentielle Schaltkreise implementiert werden, die entsprechend der **RT-Methode (Register-Transfer)** arbeiten.

Bei der RT-Methode werden Berechnungen durch die **Manipulation** und den **Transfer** von Daten zwischen Registern durchgeführt. Eine elementare Register-Transfer Instruktion hat die Form

$$r_{\text{dest}} \leftarrow f(r_{\text{src}_1}, \ldots, r_{\text{src}_n})$$

Die Funktion f wird auf die Werte der Register $r_{\text{src}_1}, \ldots, r_{\text{src}_n}$ angewendet und das Ergebnis im Register r_{dest} abgelegt.

Notizen

BEISPIELE FÜR REGISTER-TRANSFER INSTRUKTIONEN

Die folgenden Instruktionen zeigen beispielhaft die Möglichkeiten.

Transfer- und Initialisierungsoperationen:

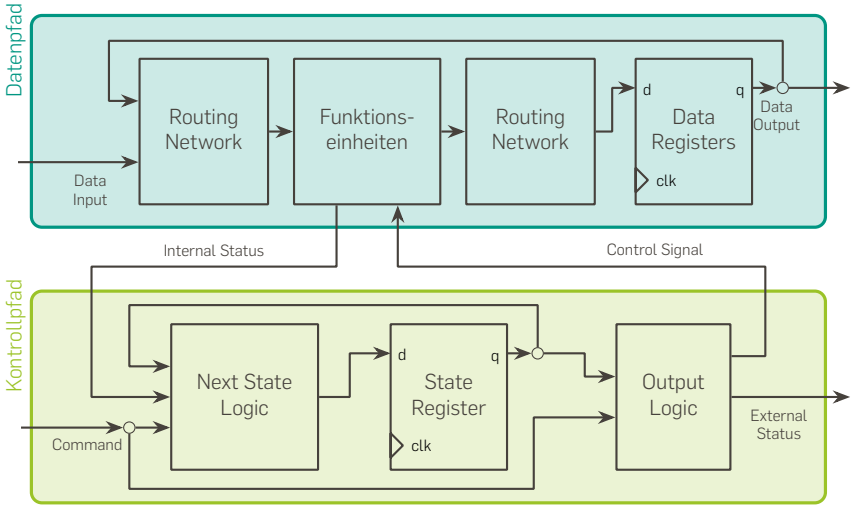
- $r_i \leftarrow 0$ (belege Register i mit dem Wert 0)
- $r_j \leftarrow r_i$ (speichere den Inhalt von Register i in Register j)
- $r_i \leftarrow r_j ? r_a : r_b$ (wenn $r_j \neq 0$ speichere den Inhalt von r_a in r_i, sonst speichere r_b in r_i)

Logische und arithmetische Operationen:

- $r_i \leftarrow r_a \,\mathrm{AND}\, r_b$ (und-Verknüpfung der Register r_a und r_b)
- $r_i \leftarrow r_i << 3$ (schiebe r_i um drei Bits nach links)
- $r_i \leftarrow r_a + r_b$ (addiere r_a mit r_b und speichere Ergebnis in r_i)
- $r_i \leftarrow r_i + 1$ (Inkrement)

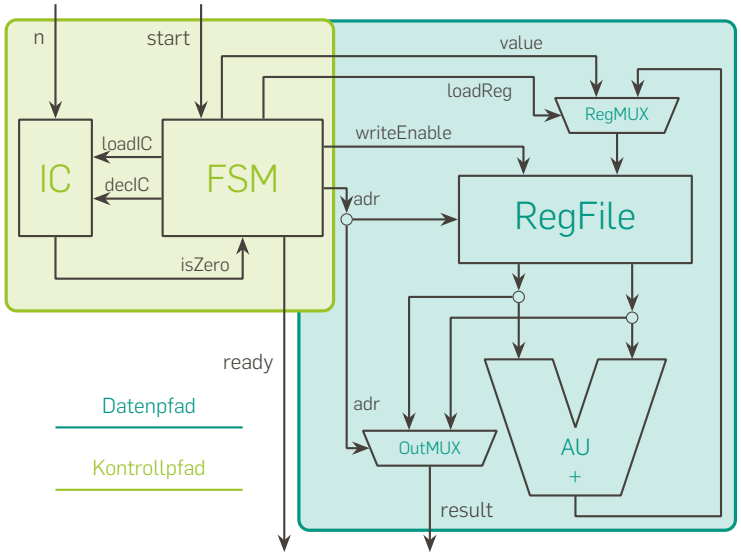
Notizen

AUTOMAT MIT DATENPFAD - GRUNDLEGENDE STRUKTUR



Notizen

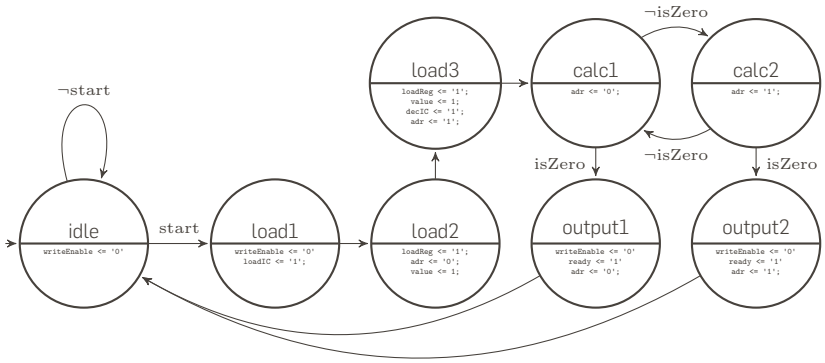
BEISPIEL: EIN SCHALTKREIS FÜR FIBONACCI-ZAHLEN



Notizen

BEISPIEL: EIN SCHALTKREIS FÜR FIBONACCI-ZAHLEN (II)

Defaultwerte: `loadIC <= '0'`, `value <= (others => '0')`, `adr <= '0'`, `loadReg <= '0'`, `decIC <= '0'`, `ready <= '0'` und `writeEnable <= '1'`.



Zusätzlich: Die Moore-Übergänge ¬isZero belegen `decIC <= '1'` bzw. die isZero Übergänge belegen `writeEnable <= '0'`.

Notizen

SCHNITTSTELLEN DES DATENPFADS

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  library FibLib;
5  use FibLib.FTypes.all;
6
7  entity DataPath is
8
9    port (reset       : in  std_logic;
10         clk         : in  std_logic;
11         value       : in  word_t;
12         loadReg     : in  std_logic;
13         adr         : in  std_logic;
14         writeEnable : in  std_logic;
15         output      : out word_t);
16
17 end DataPath;
```


Notizen

DIE ARITHMETISCHE EINHEIT

Die arithmetische Einheit stellt eine **extrem** vereinfachte Version einer ALU dar.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  library FibLib;
6  use FibLib.FTypes.all;
7
8  entity AU is
9
10   port (portA  : in  word_t;
11         portB  : in  word_t;
12         result : out word_t);
13
14 end AU;
15
16 architecture Behavioral of AU is
17 begin
18   result <= std_logic_vector(unsigned(portA) + unsigned(portB));
19 end architecture;
```


Notizen

SCHNITTSTELLEN DES KONTROLLPFADS

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  library FibLib;
5  use FibLib.FTypes.all;
6
7  entity ControlPath is
8
9    port (reset       : in  std_logic;
10         clk         : in  std_logic;
11         n           : in  word_t;
12         value       : out word_t;
13         adr         : out std_logic;
14         loadReg     : out std_logic;
15         writeEnable : out std_logic;
16         start       : in  std_logic;
17         ready       : out std_logic);
18
19 end ControlPath;
```


Notizen

ERZEUGUNG DER STEUERSIGNALE

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  library FibLib;
5  use FibLib.FTypes.all;
6
7  entity FSM is
8
9    port (reset       : in  std_logic;
10         clk         : in  std_logic;
11         loadIC      : out std_logic;
12         decIC       : out std_logic;
13         value       : out word_t;
14         adr         : out std_logic;
15         loadReg     : out std_logic;
16         writeEnable : out std_logic;
17         start       : in  std_logic;
18         isZero      : in  std_logic;
19         ready       : out std_logic);
20
21 end FSM;
```


Notizen

ERZEUGUNG DER STEUERSIGNALE (II)

Es wird das übliche Design-Pattern für Moore/Mealy-Automaten verwendet:

```
1  architecture Behavioral of FSM is
2
3    type state_t is (idle,load1,load2,load3,calc1,calc2,output1,
4                     output2,crash);
5    signal state_reg, state_next : state_t;
6
7  begin
8
9    state_handler : process (reset, clk)
10   begin
11
12     if (reset = '1') then
13       state_reg <= idle;
14     elsif (rising_edge(clk)) then
15       state_reg <= state_next;
16     end if;
17
18   end process;
```


Notizen

ERZEUGUNG DER STEUERSIGNALE (III)

```
1   transition : process(state_reg, start, isZero)
2    begin
3      -- Set defaults
4      state_next <= state_reg; writeEnable <= '1';
5      loadIC      <= '0';      value       <= (others => '0');
6      adr         <= '0';      loadReg     <= '0';
7      decIC       <= '0';      ready       <= '0';
8
9      case (state_reg) is     -- Handle all possible states
10
11       when idle =>
12
13         writeEnable <= '0';  -- Do not write to the register file
14
15         if (start = '1') then -- Check for start signal
16           state_next <= load1;
17         else
18           state_next <= idle; -- Wait for start
19         end if;
```


Notizen