Realisierung von Algorithmen in Hardware

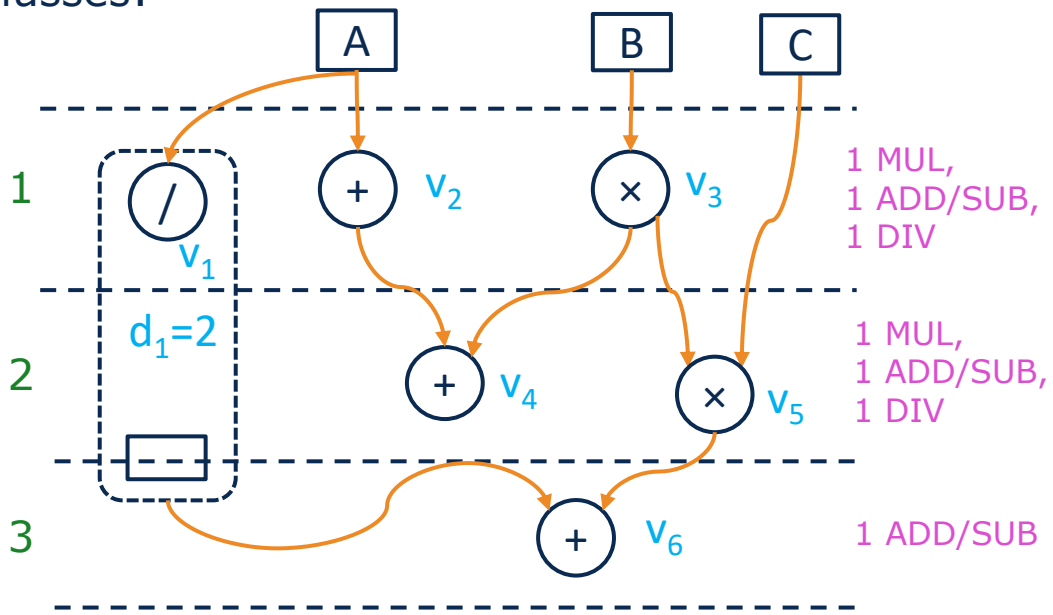
© Sebastian Höppner 2015

- Algorithmus Beschreibungsformen
  - Textform
  - Gleichungen
  - (Pseudo-) Code
  - Struktogramm
  - …

- Szenario 1:
  - Programmierung einer gegebenen Hardware **(Prozessor)**
    - Gegebener Datenpfad, Speicherarchitektur, Befehlssatz
  - **Abbilden** des Codes auf die Hardware durch Compiler
- Szenario 2:
  - Entwurf einer für den Algorithmus spezifischen Hardware (ASIC)
    - **Entwurf** von Datenpfad, Speicherarchitektur, Control-Flow

© Sebastian Höppner 2015

- Formale Darstellung des Datenflusses:



- Operationen $V=\{v_0,v_1,v_2,\ldots\}$
- Verzögerung $D=\{d_0,d_1,d_2,\ldots\}$
- Menge an Kanten $E=\{e_{ij},\ldots\}$
- $Pred_{vi}$ alle Vorgänger von v_i
- $Succ_{vi}$ alle Nachfolger von v_i

- Die „Laufzeit" einer Operation wird in Takt Zyklen angegeben
- Darstellung der **Startzeit** und der **Verzögerungszeit**
- Angabe der notwendigen **Hardware Ressourcen**

© Sebastian Höppner 2015

- Festlegen der Abarbeitungsreihenfolge (Schedule) der Operationen
- Berücksichtigung von Randbedingungen (Constraints)
  - Laufzeit (Timing Constraints)
  - Hardwareaufwand (Ressource Constraints)
- Scheduling ist ein Optimierungsproblem
  - Festlegen der Start-Zeit der Operationen
  - Einhalten der gegebenen Randbedingungen

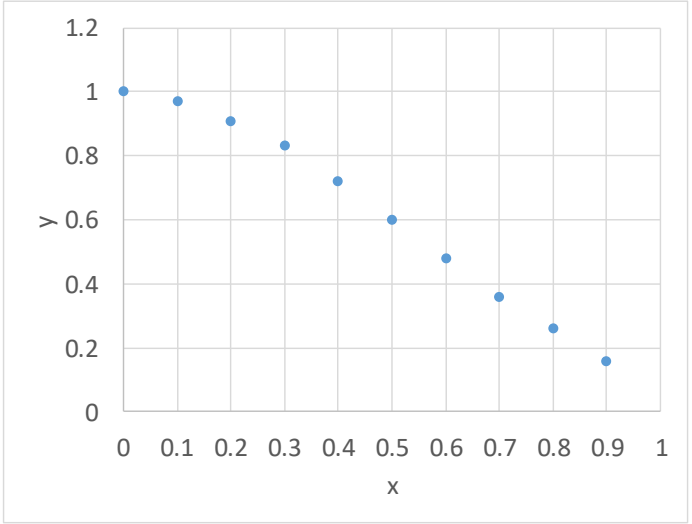
- Ergebnis des Scheduling:
  - Erstellen eines DFG
  - Daraus abgeleitet:
    - Datenpfad
      - Welche und wie viele Datenpfadbaublöcke?
      - Welche und wie viele Register?
      - Wie viele Busse?
    - Register-Transfer Folge zum Entwurf des Steuerwerks

© Sebastian Höppner 2015

Beispiel Algorithmus

- Numerische DGL Lösung: y‘‘+3xy‘+3y=0
- Euler-Vorwärts-Verfahren
- Speichern des Ergebnis im RAM an ADDR

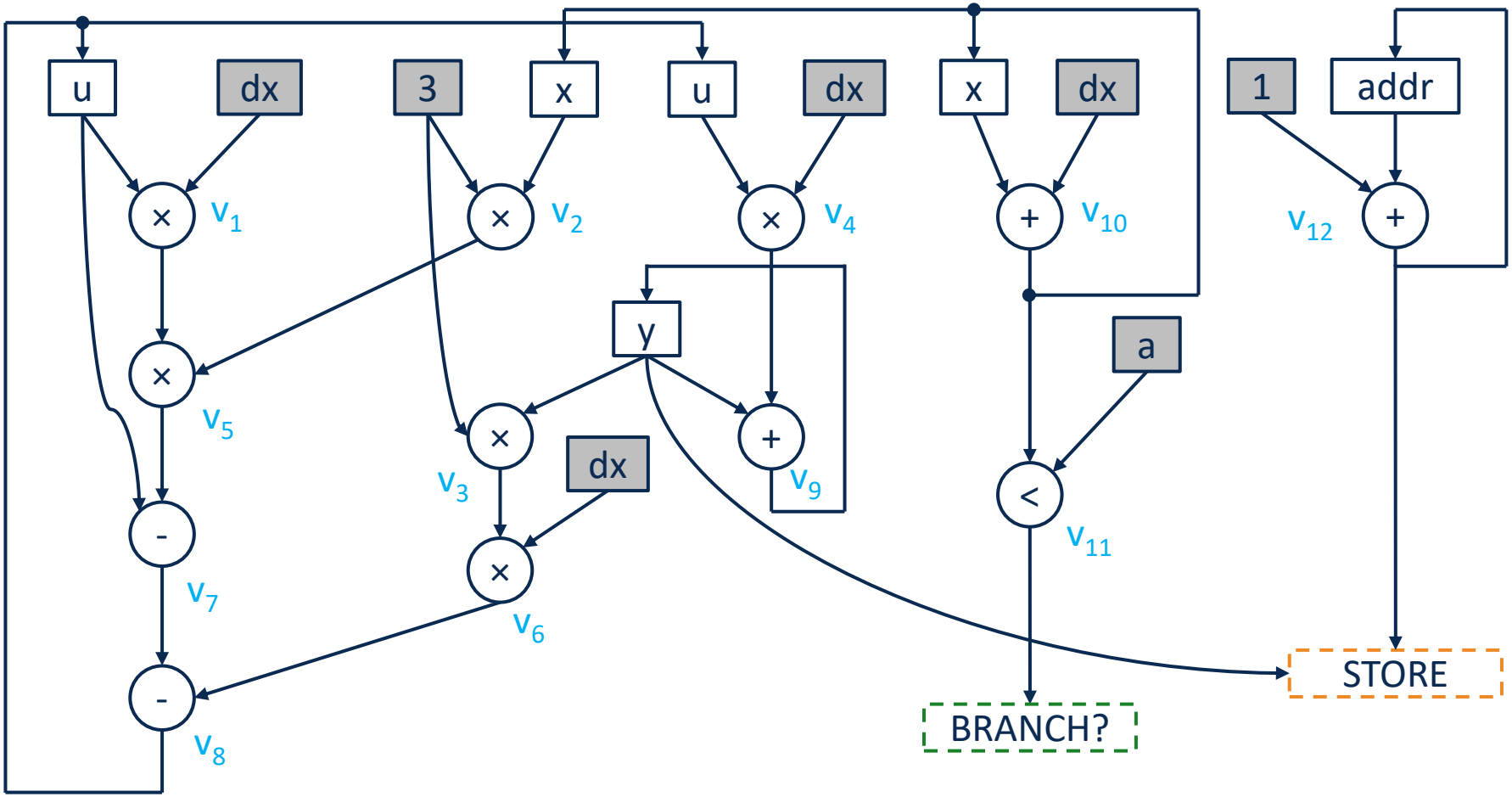
```
WHILE (x<a) DO
        x1:=x+dx;
        u1:=u-(3·x·u·dx)-(3·y·dx);
        y1:=y+(u·dx);
        x := x1; u := u1; y :=y1;
        addr := addr+1; STORE(y1,addr);
ENDWHILE
```



Quelle: Scheduling Algorithms for High-Level Synthesis, Zoltan Baruch

© Sebastian Höppner 2015

Beispiel Algorithmus Datenfluss

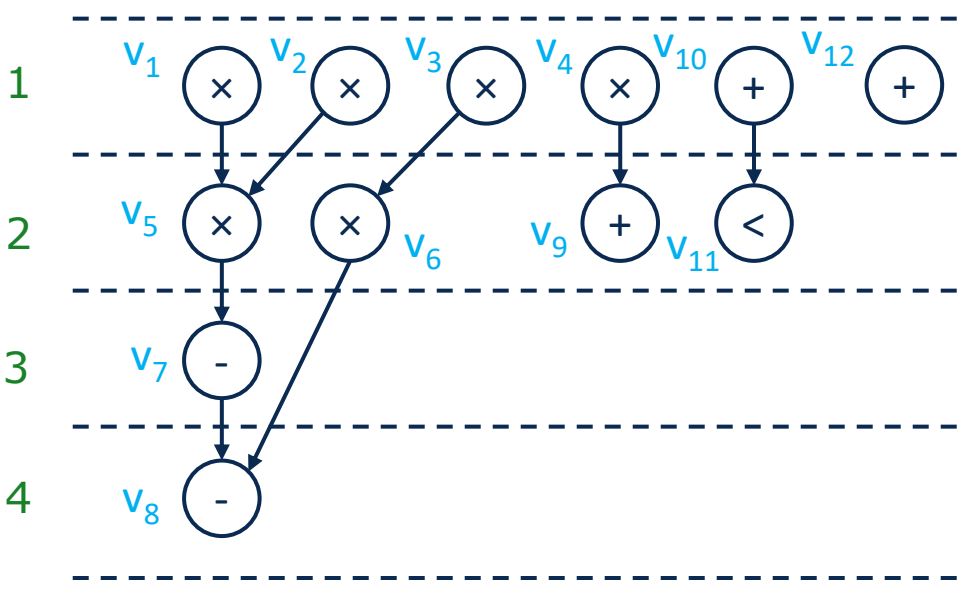


Quelle: Scheduling Algorithms for High-Level Synthesis, Zoltan Baruch

© Sebastian Höppner 2015

As Soon As Possible (ASAP) Scheduling

- Scheduling Algorithmus:
  - Wiederhole für alle Knoten v_i
    - Auswahl eines v_i dessen **Vorgänger** alle zugewiesen sind
    - Terminiere v_i → $\mathbf{t_i = max\ (t_j + d_j)}$ für alle j aus $Pred_{vi}$ (→ frühester Zeitpunkt)
  - Bis alle v_i zugewiesen sind



4 MUL, 2 ADD/SUB

2 MUL, 2 ADD/SUB

1 ADD/SUB

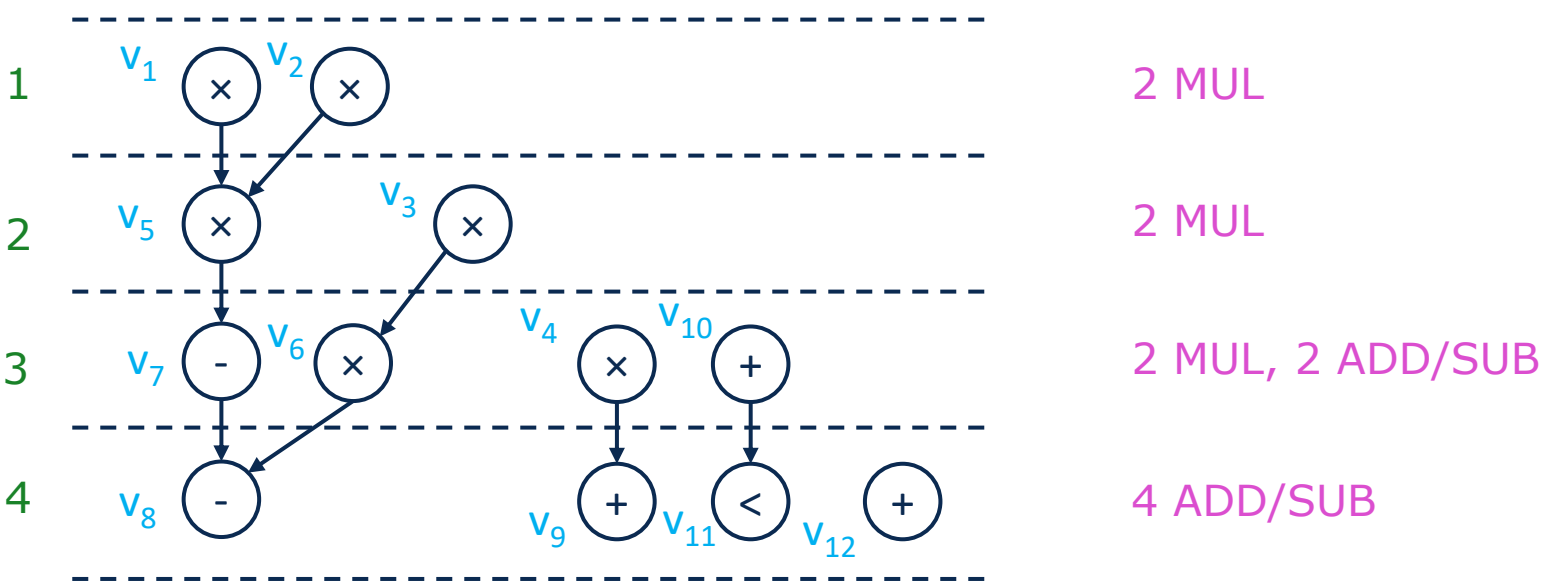
1 ADD/SUB

→ 4 MUL, 2 ADD/SUB

© Sebastian Höppner 2015

As Late As Possible (ALAP) Scheduling

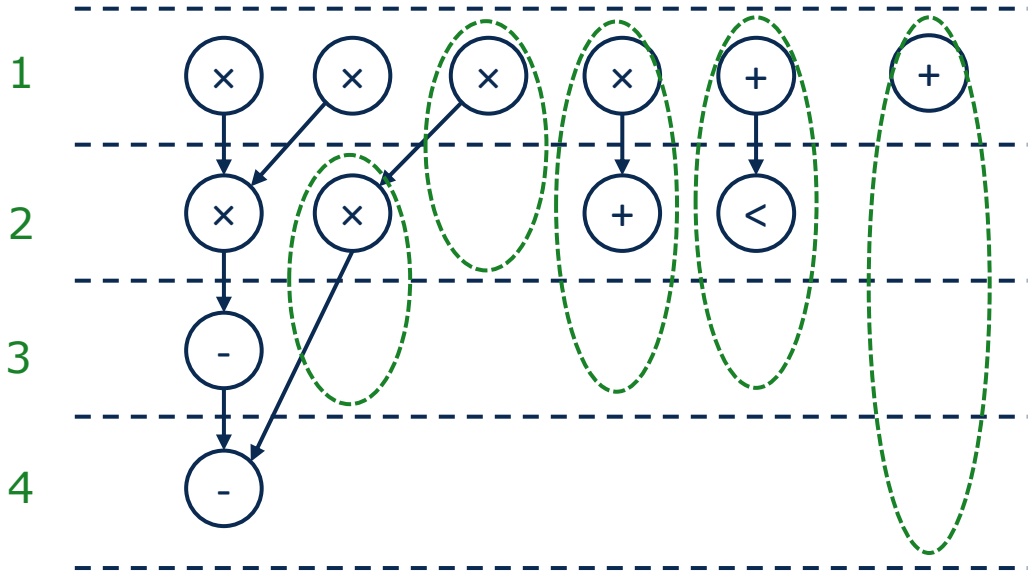
- Scheduling Algorithmus :
  - Wiederhole für alle Knoten v_i
    - Auswahl eines v_i dessen **Nachfolger** alle zugewiesen sind
    - Terminiere v_i → $\mathbf{t_i = min\ (t_j - d_i)}$ für alle j aus $Succ_{vi}$ (→ spätester Zeitpunkt)
  - Bis alle v_i zugewiesen sind



→ 2 MUL, 4 ADD/SUB

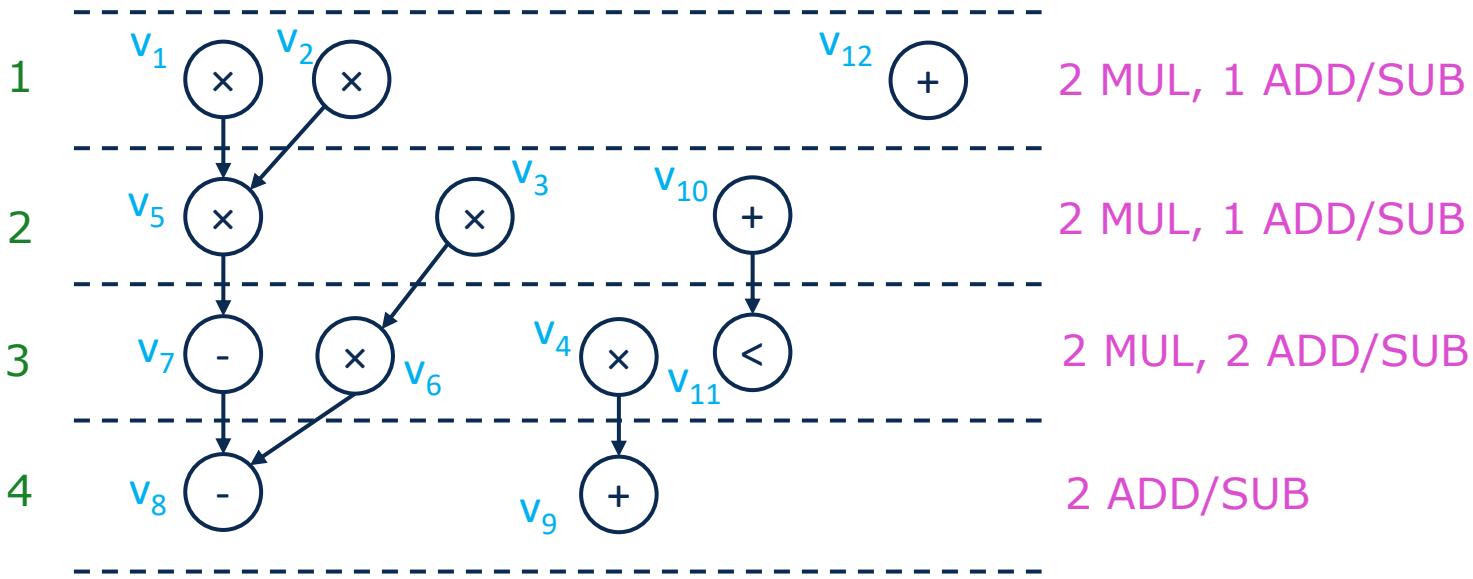
© Sebastian Höppner 2015

- Freiheitsgrad des Startzeitpunktes von Operationen ohne Auswirkung auf die Latenz des DFG



© Sebastian Höppner 2015

- Scheduling für
  - gegebene Hardware Ressourcen (Chipfläche)
  - Extremfall: minimale Hardware (Module, Busse)
- Nutzung der Mobility (keine Erhöhung der Latenz):

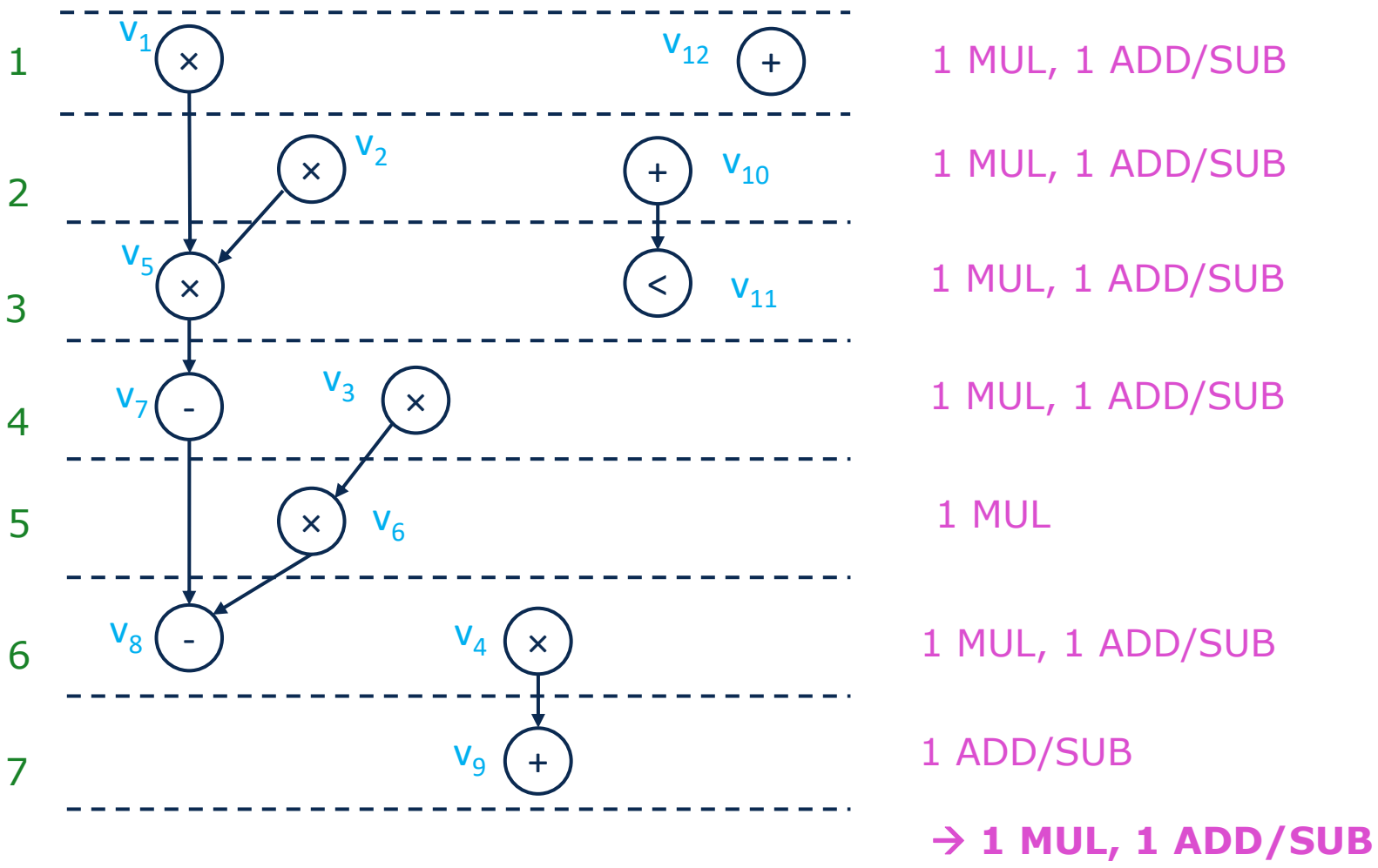


→ 2 MUL, 2 ADD/SUB

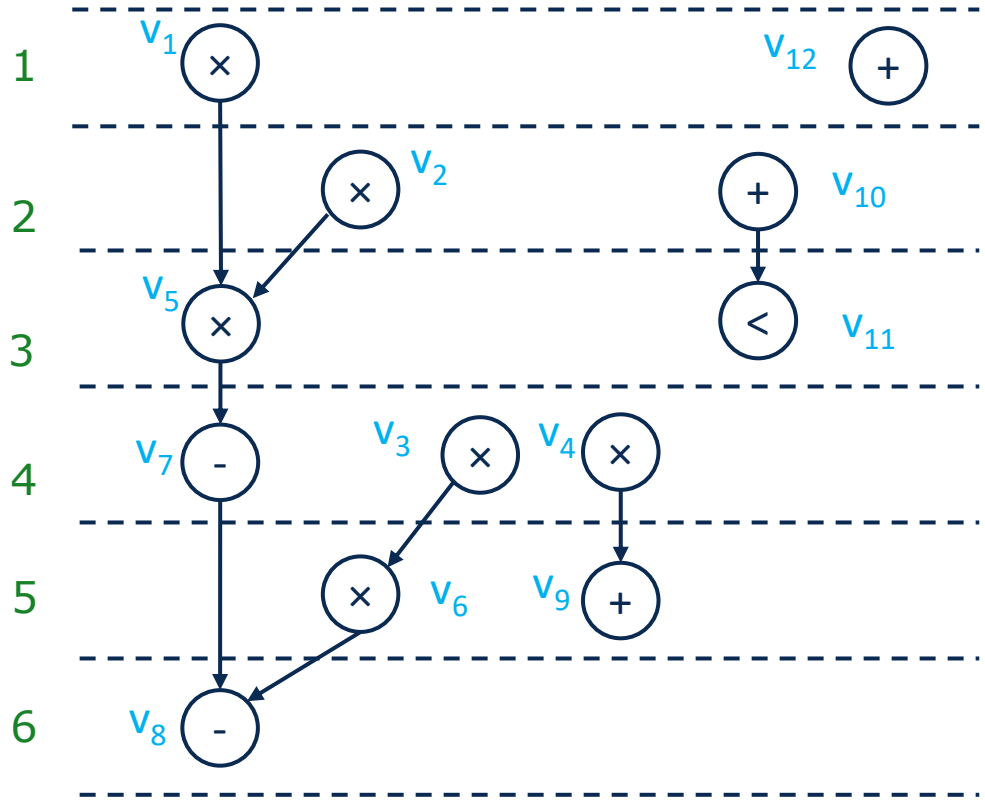
© Sebastian Höppner 2015

Beispiel: Scheduling für minimale Hardware

- Einfügen zusätzlicher Takte → Erhöhung der Latenz



- Scheduling für gegebene Abarbeitungszeit
- Anwendung z.B. digitale Signalprozessoren (DSP) mit Echtzeit Anforderung → Ergebnis muss nach gegebener Taktzahl vorliegen
- Beispiel: Minimale Hardware bei 6 Takten Latenz



1 MUL, 1 ADD/SUB

1 MUL, 1 ADD/SUB

1 MUL, 1 ADD/SUB

2 MUL, 1 ADD/SUB

1 MUL, 1 ADD/SUB

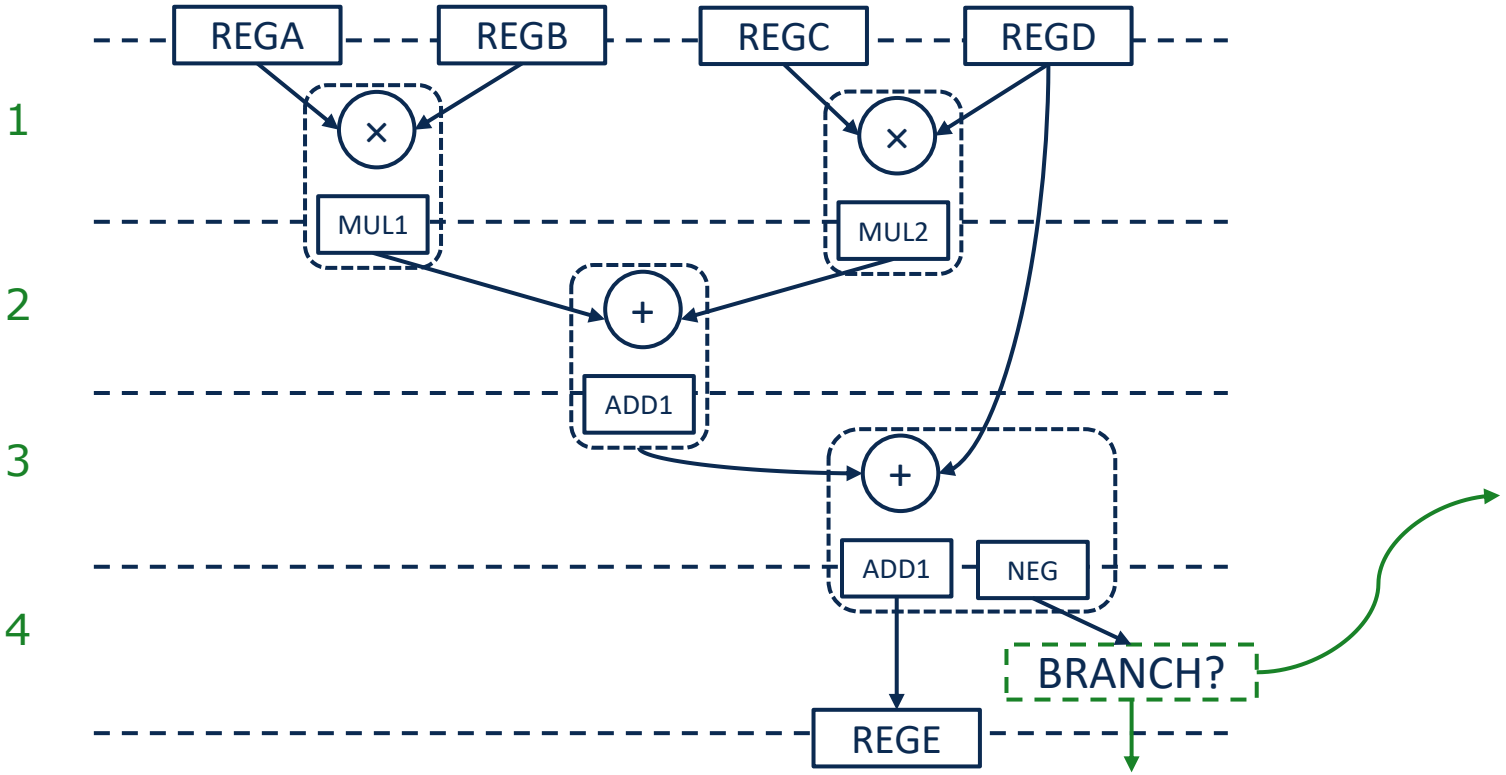
1 ADD/SUB

→ 2 MUL, 1 ADD/SUB

© Sebastian Höppner 2015

Register und Verzweigungen im DFG

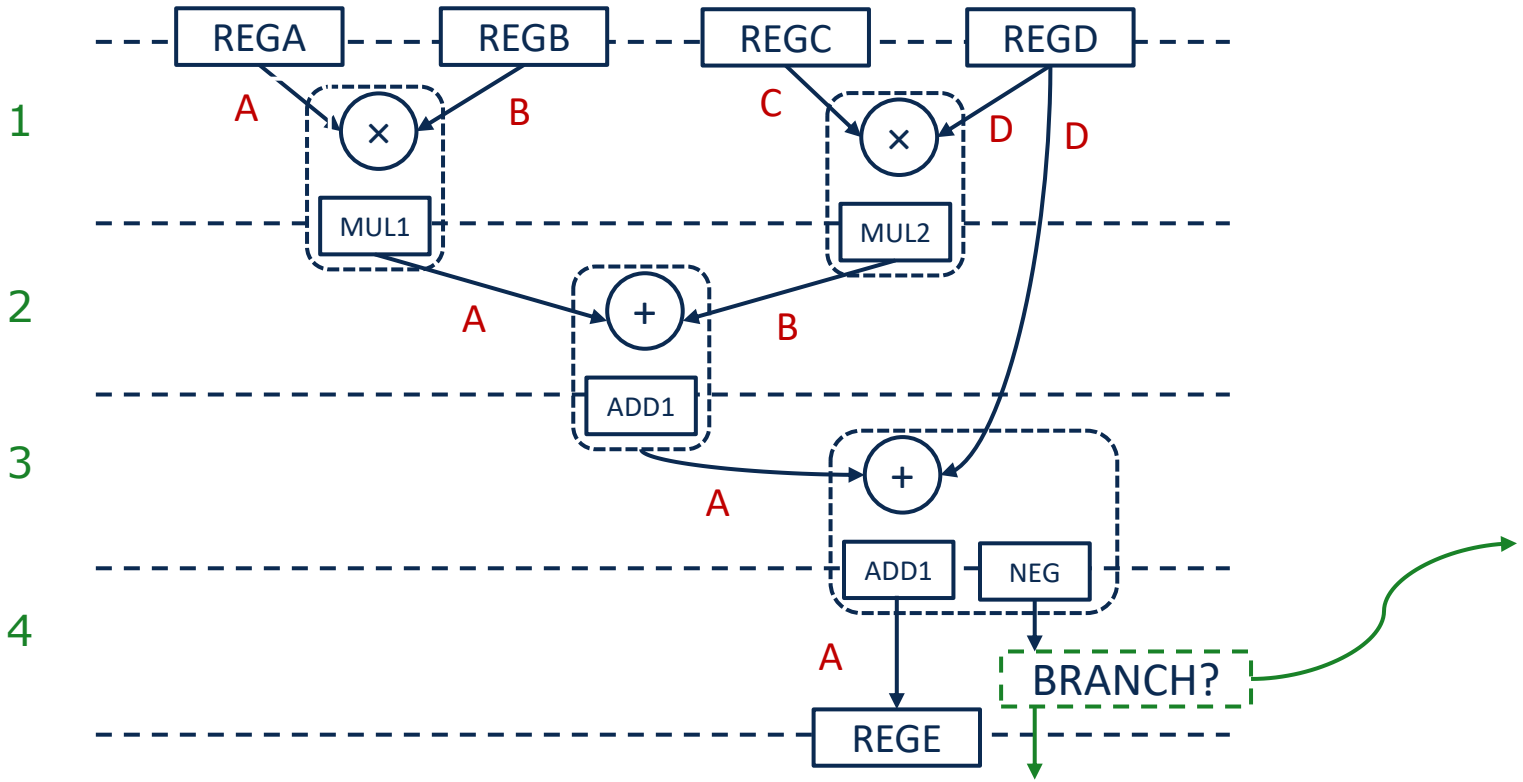
- Detaillierter DFG mit expliziter Darstellen von Registern
  - Visualisierung der Speicherung von Daten → Register Hardware
  - Grundlage des Entwurfs von Pipelines
- Explizite Darstellung von Verzweigungen



© Sebastian Höppner 2015

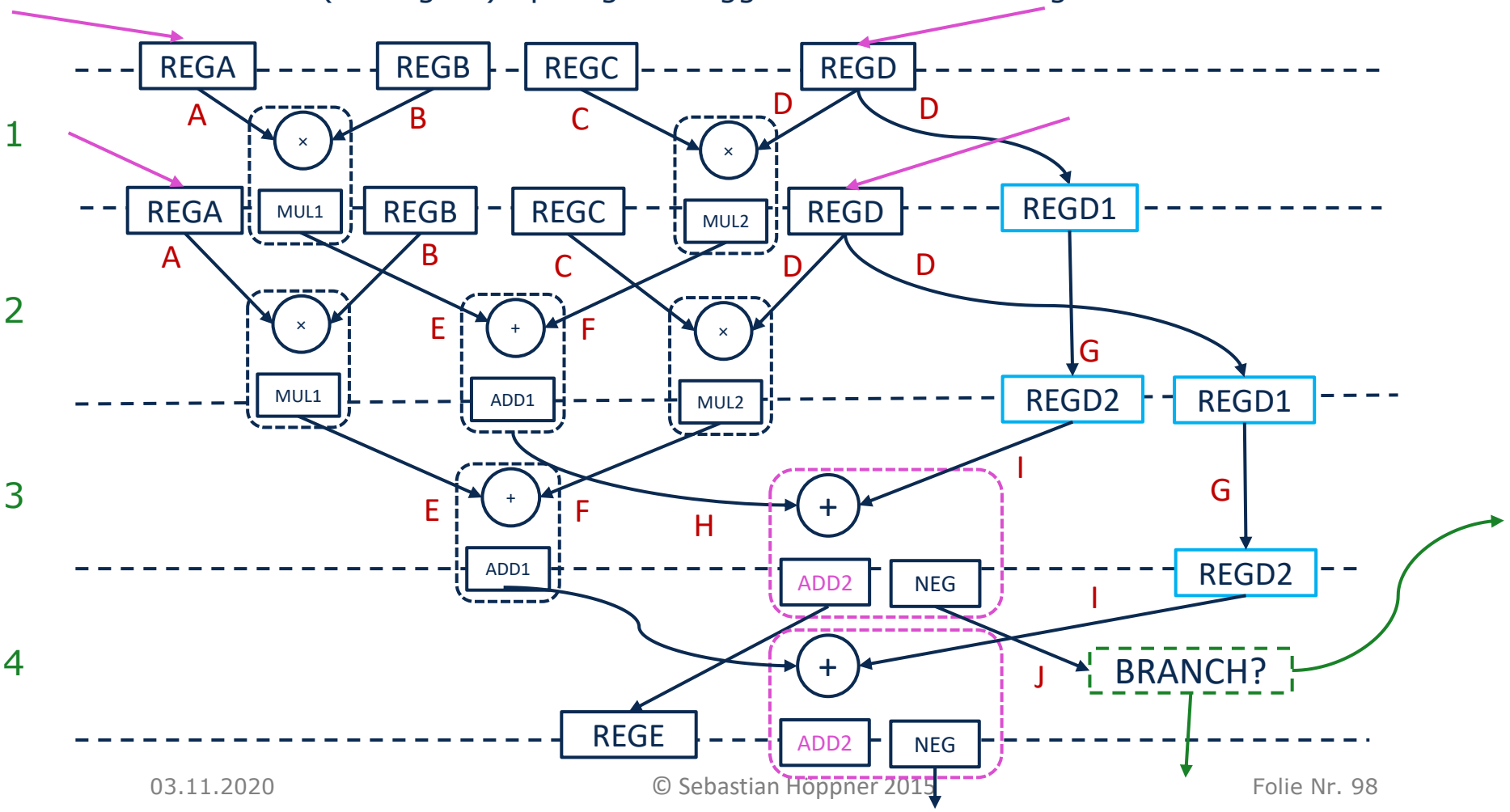
Darstellung von Datenbussen im DFG

- Zuweisung von Datenbussen an Kanten des DFG
- → Bestimmung der Anzahl der notwendigen Datenbusse



© Sebastian Höppner 2015

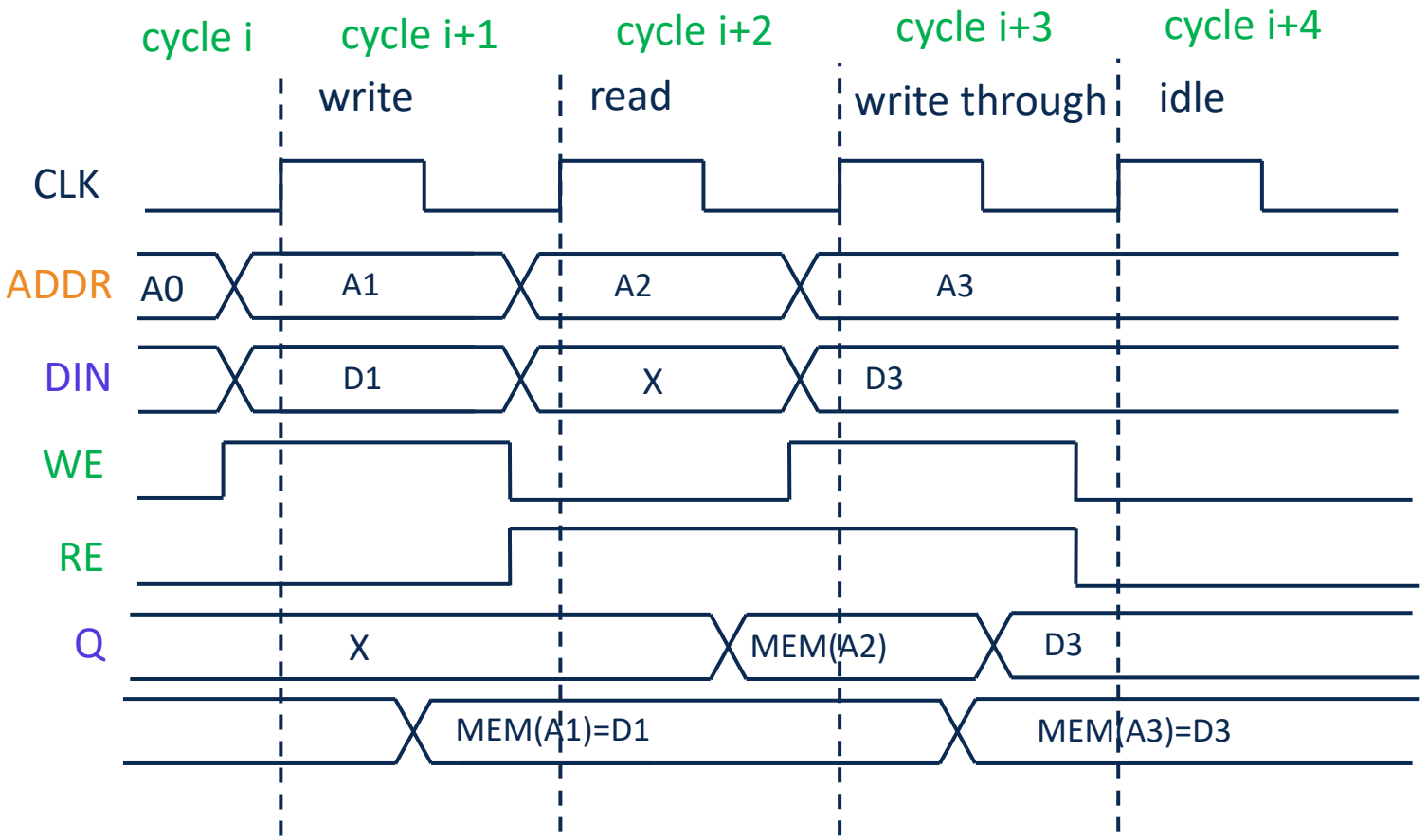
- Aufbau von Pipelines durch bei Bedarf Hinzufügen von:
  - Registern, Datenpfadbaublöcken, Bussen
- Durchsatz von 1 Ergebnis pro Taktzyklus bei N Takten Latenz
- Beachten von (bedingten) Sprüngen! → ggf. Verwerfen von Ergebnissen



© Sebastian Hoppner 2015

Darstellung von Speicherzugriffen im DFG

- Sequentieller SRAM Zugriff

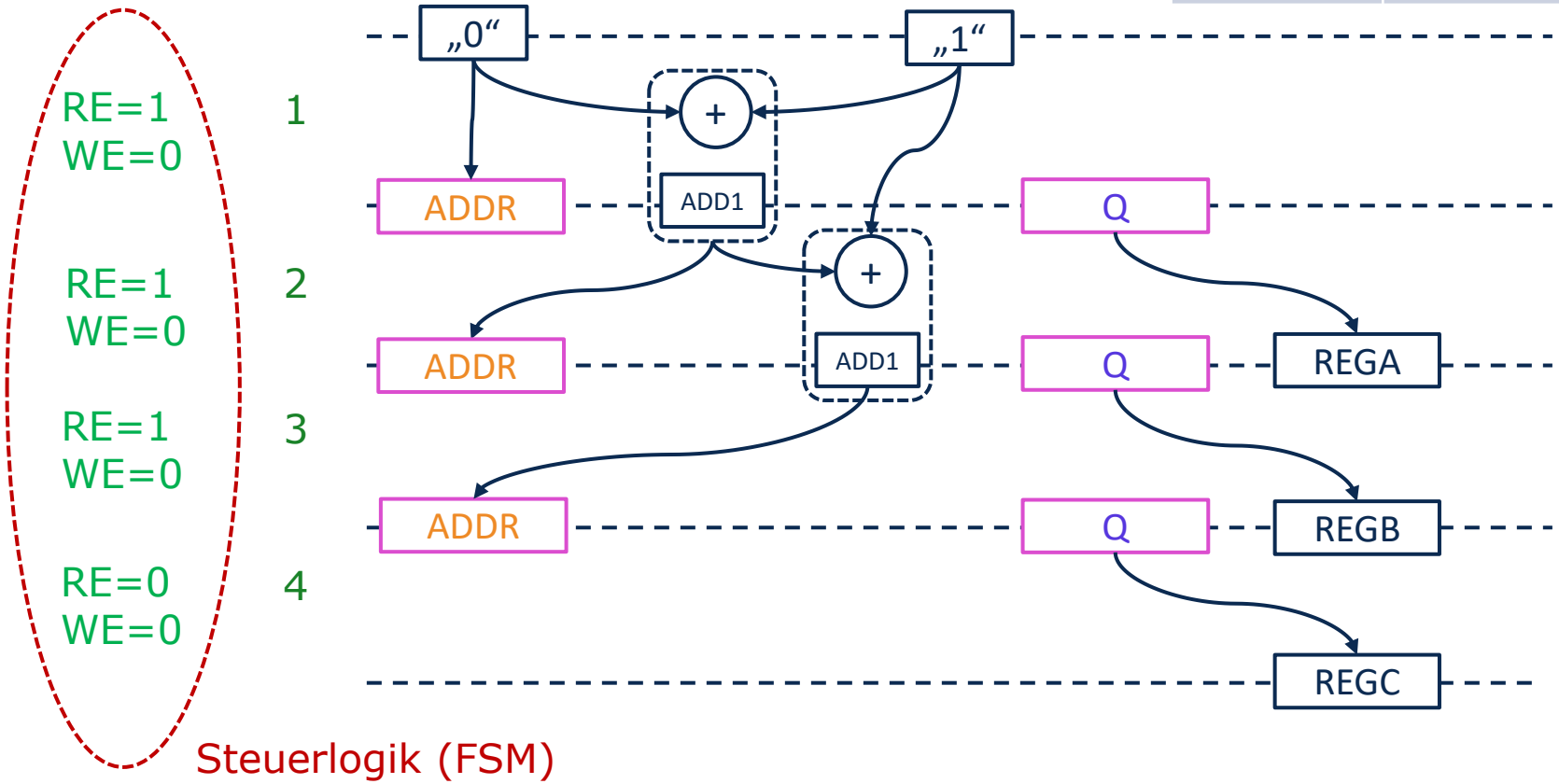


© Sebastian Höppner 2015

Darstellung von Speicherzugriffen- Lesen vom SRAM

- Berechnen der Adresse durch Fixed-Point ALU
- Betrachtung der SRAM Signale wie Register
- Beispiel:

Adresse	Aktion
0	Lese nach REGA
1	Lese nach REGB
2	Lese nach REGC

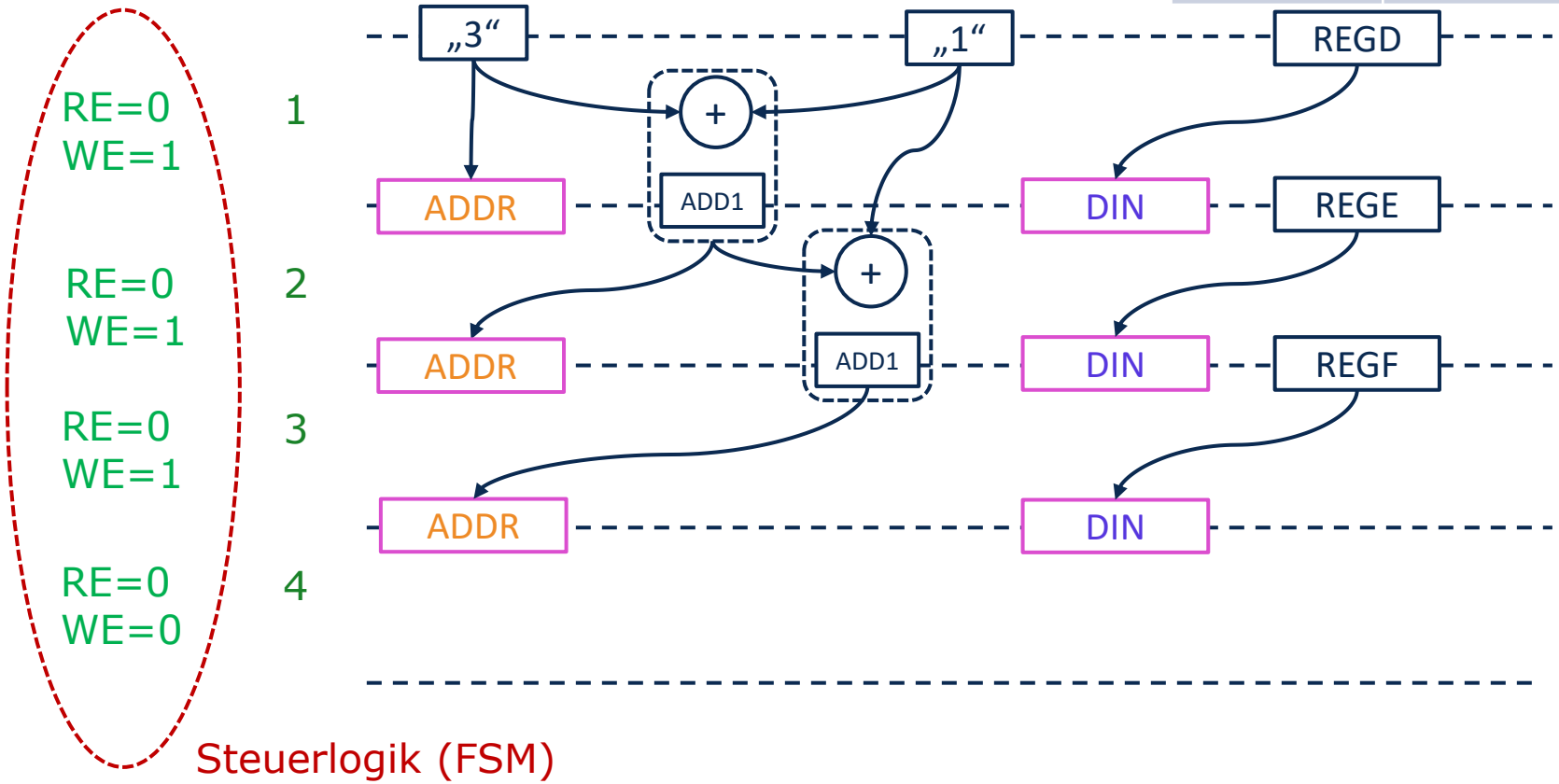


© Sebastian Höppner 2015

Darstellung von Speicherzugriffen- Schreiben auf SRAM

- Berechnen der Adresse durch Fixed-Point ALU
- Betrachtung der SRAM Signale wie Register
- Beispiel:

Adresse	Aktion
3	Schreibe von REGD
4	Schreibe von REGE
5	Schreibe von REGF

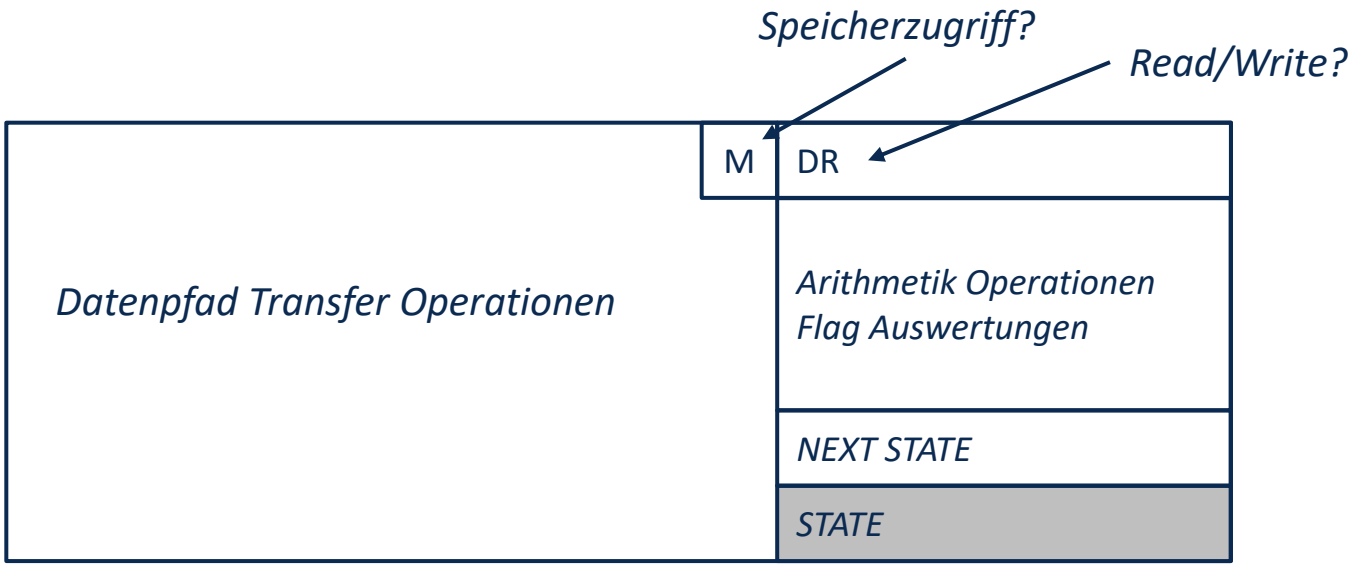


© Sebastian Höppner 2015

- Konstruktion des Datenpfades aus einem DFG
- Bestimmen der Anzahl von
  - Modulen
  - Registern
  - Busse/Datenverbindungen
- Entwurf des Datenpfades mit Bussystem (z.B.Multiplexer)
  → siehe Abschnitt Datenpfade/Busse

© Sebastian Höppner 2015

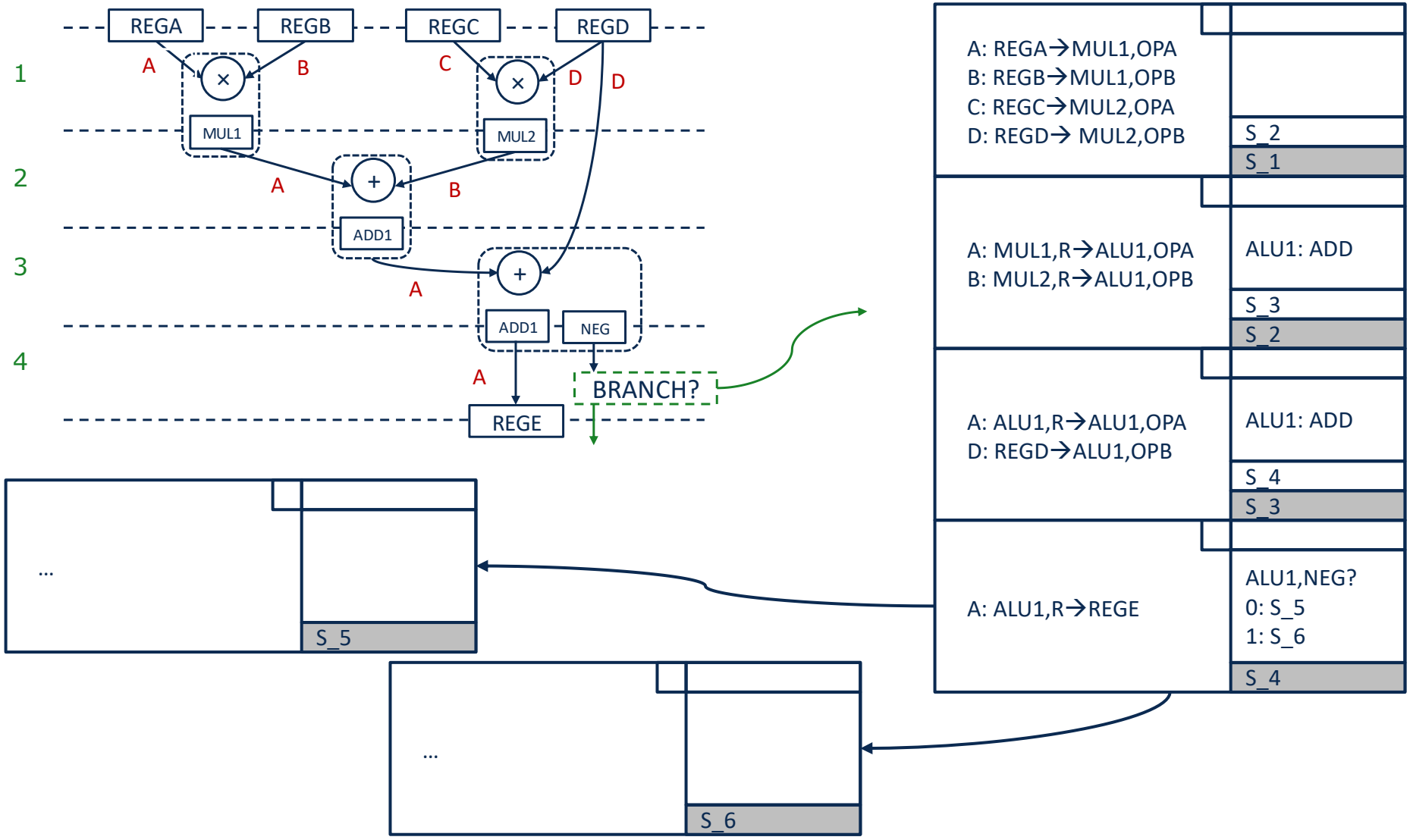
- Darstellung der Abläufe in Register Transfer Notation



- Ein Register-Transfer Block beschreibt **einen** Taktzyklus

© Sebastian Höppner 2015

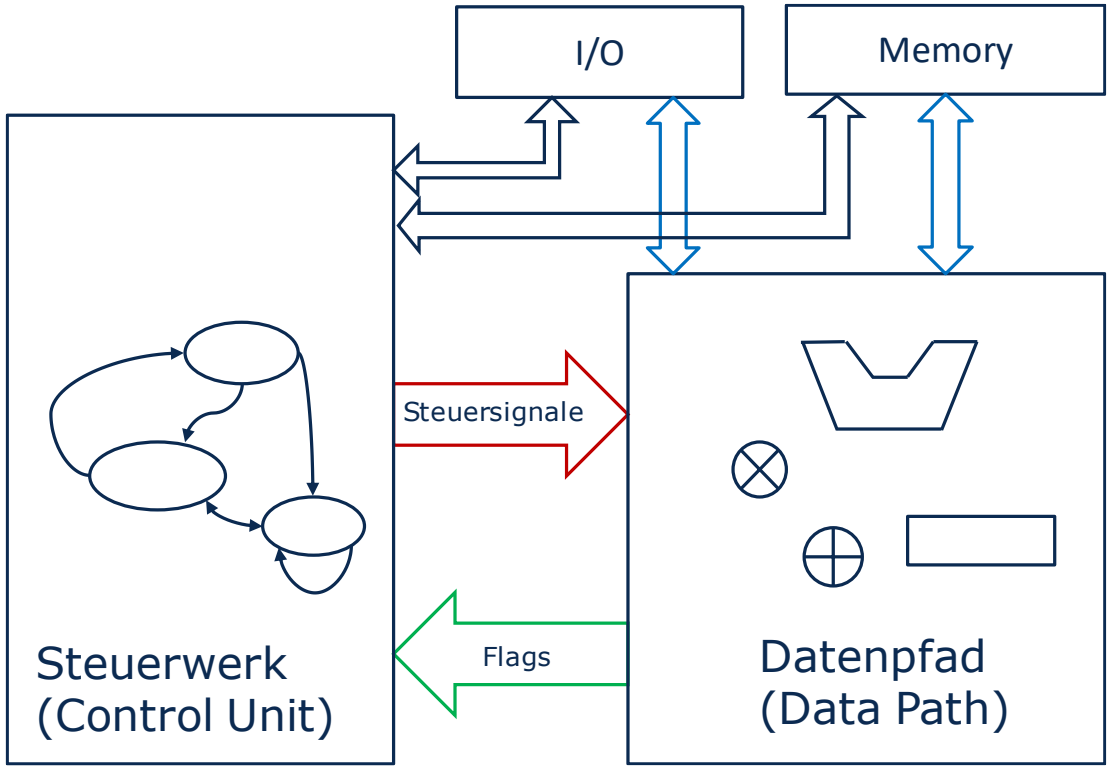
Beispiel RT-Folge



- Die RT-Darstellung legt die Zustände und Zustandsübergänge der FSM fest
  - Bedingungen: Flags
- Die RT-Darstellung legt die Signale → Ausgangslogik der FSM
  - Schalten von Bussen (Tristate-Treiber, Multiplexer)
  - Konfigurieren von Datenpfadbaublöcken (ADD/SUB)
  - …
- → siehe Abschnitt FSM
- → Details und ausführliches Beispiel siehe Praktikumsanleitung

© Sebastian Höppner 2015

- Vorstellung des Datenflussgraphen, Scheduling und Optimierung
  - ASAP, ALAP, Ressource Constraints und Timing Constraints
  - Pipelining
- Register Transfer Folge → Ableiten von Datenpfad und FSM



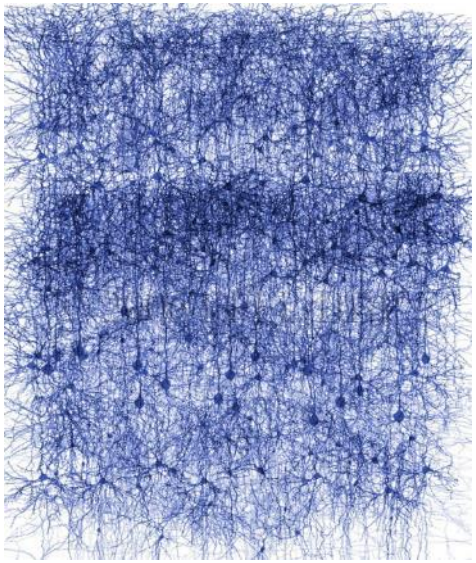
© Sebastian Höppner 2015

Beispiel: Exponentialfunktion für Neuromorphe Hardware

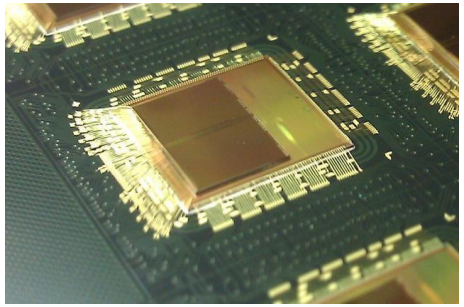
- Neuromorphe Hardware für das Human Brain Project
  - Gehirnsimulation
  - Technische Anwendungen, z.B. Robotik
- Herausforderung:
  - Simulation von einer **großen Anzahl** Neuronen und Synapsen in **biologischer Echtzeit**
- Entwicklung eines Many-Core Computers mit > 4 Millionen ARM Prozessoren (SpiNNaker 2)
  - Architekturentwicklung → University of Manchester
  - Chip-Design → TU Dresden

Quelle: Human Brain Project

Human Brain Project



http://bluebrain.epfl.ch

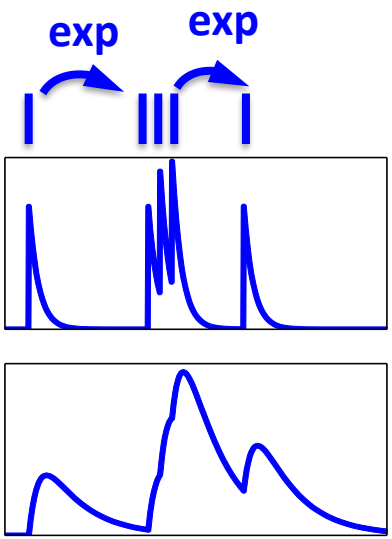


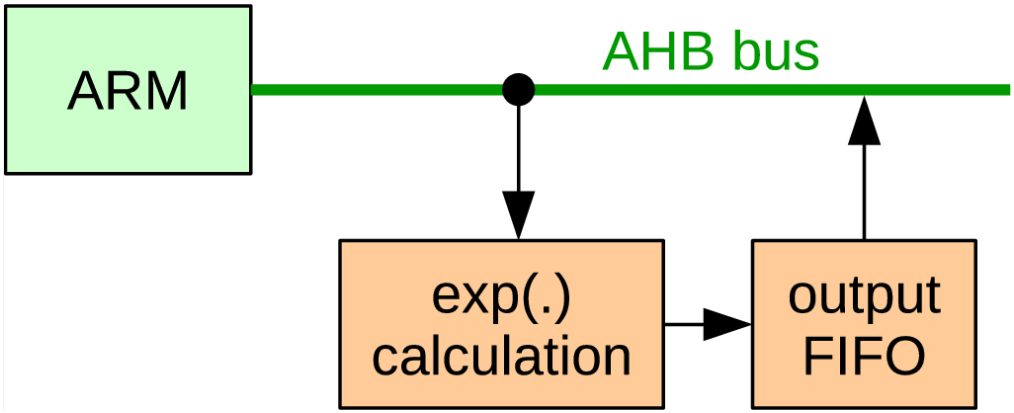
SpiNNaker 1 Chip

© Sebastian Höppner 2015

Beispiel: Exponentialfunktion für Neuromorphe Hardware

- Berechnung der Dynamiken von Neuronen und Synapsen
- Dabei häufig verwendet: $\mathbf{e^x}$
- Zahlenformat: 32-Bit, s16.15 fixed-point
- Prozessor Kern (ARM Cortex M4)
  - Bisher **exp()** in Software realisiert (≈30 Takte)
  - → Hardware Beschleuniger für **exp()**,
- Anbindung als AHB Slave an den Prozessor

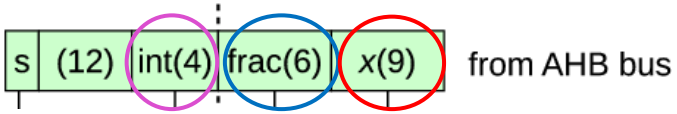


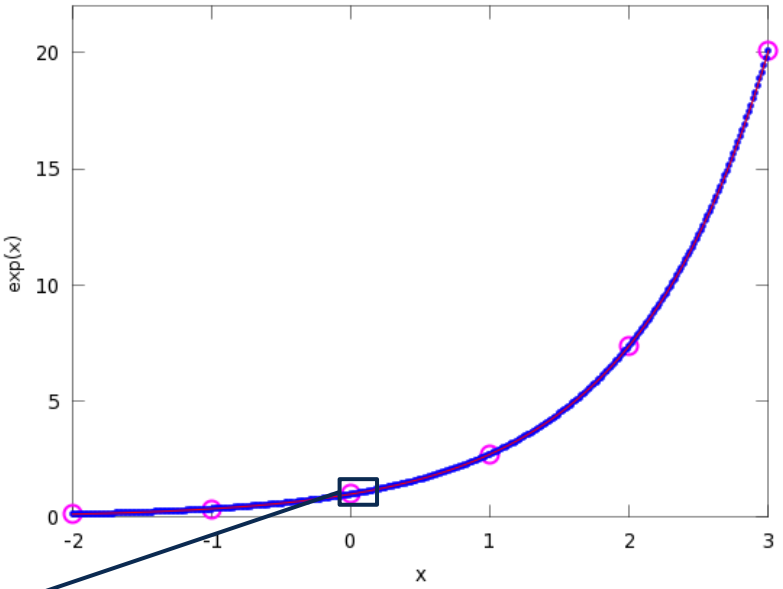


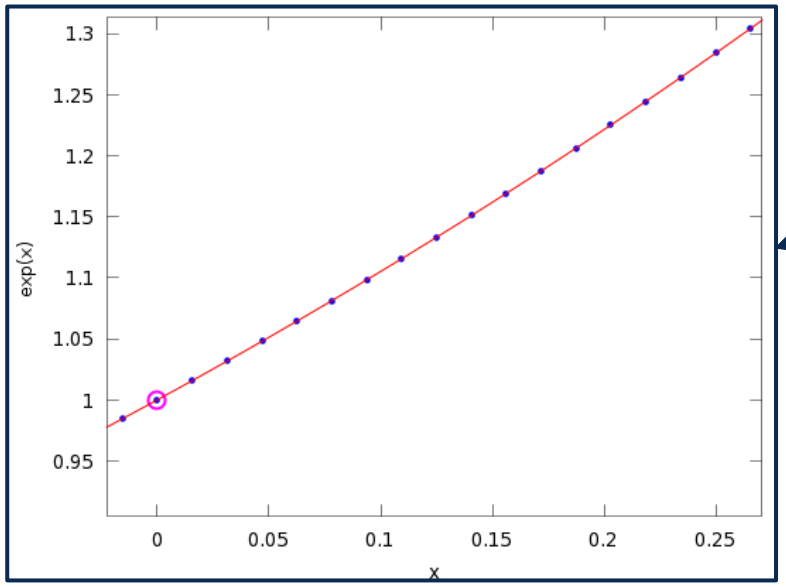
© Sebastian Höppner 2015

Beispiel: Exponentialfunktion für Neuromorphe Hardware

- Aufteilung des Argumentes der Exponentialfunktion
- $e^x = e^{(xint+xfrac+xlsb)}$
- Lookup-Tabellen für ganzzahligen Teil (int) und fraktionalen Teil (frac)
- Polynomapproximation 4. Grades für LSBs
- Genauigkeit <1LSB für 32-Bit, s16.15 fixed-point Zahlenformat



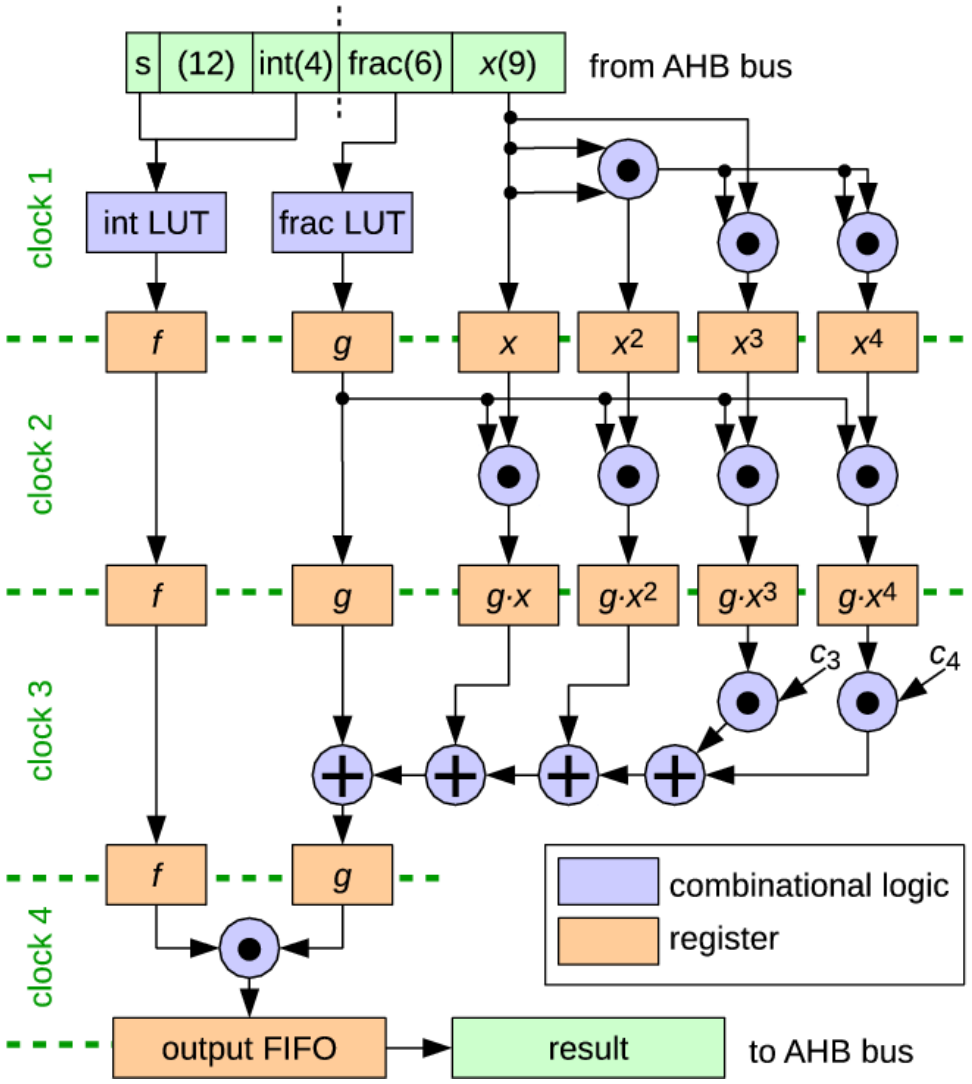




© Sebastian Höppner 2015

Beispiel: Exponentialfunktion für Neuromorphe Hardware

- Pipeline mit 4 Stufen
- FIFO am Ausgang (32 Werte)
- Realisierung in 28nm CMOS
- >500MHz Taktfrequenz möglich



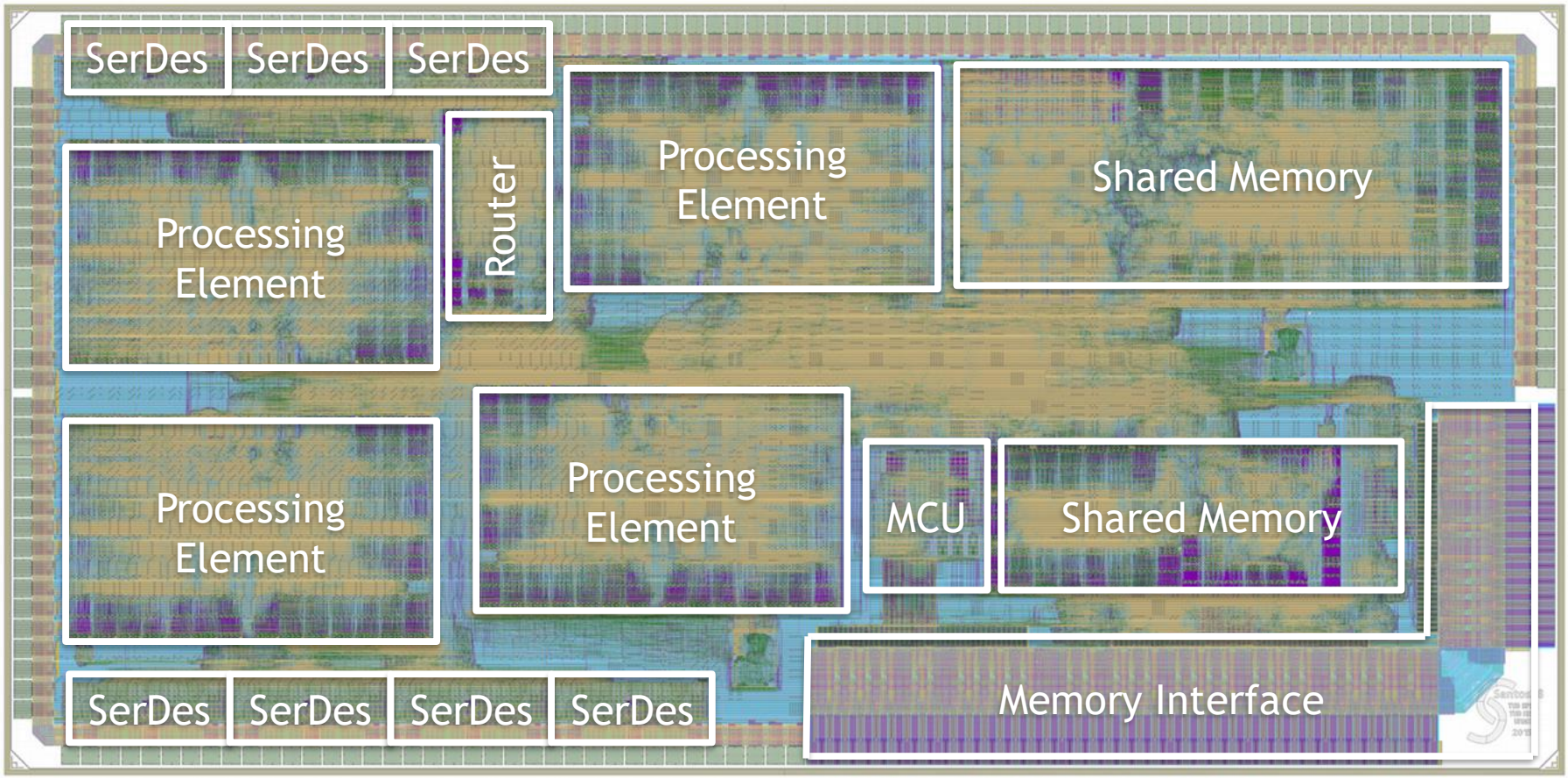
© Sebastian Höppner 2015

Beispiel: Exponentialfunktion für Neuromorphe Hardware

Measure	exp accelerator	software exp
Throughput @500MHz	250Mexp/s	5.3Mexp/s
Time per exp, pipelined	2clks/exp	95clks/exp
Latency	6clks/exp	95clks/exp
Energy per exp (nominal)	0.44nJ/exp	25nJ/exp
Energy per exp (0.7V/154MHz)	0.21nJ/exp	12nJ/exp
Total area	$10800\mu m^2$	-

J. Partzsch et al., "A fixed point exponential function accelerator for a neuromorphic many-core system," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, 2017, pp. 1-4. doi: 10.1109/ISCAS.2017.8050528

Beispiel: Exponentialfunktion für Neuromorphe Hardware



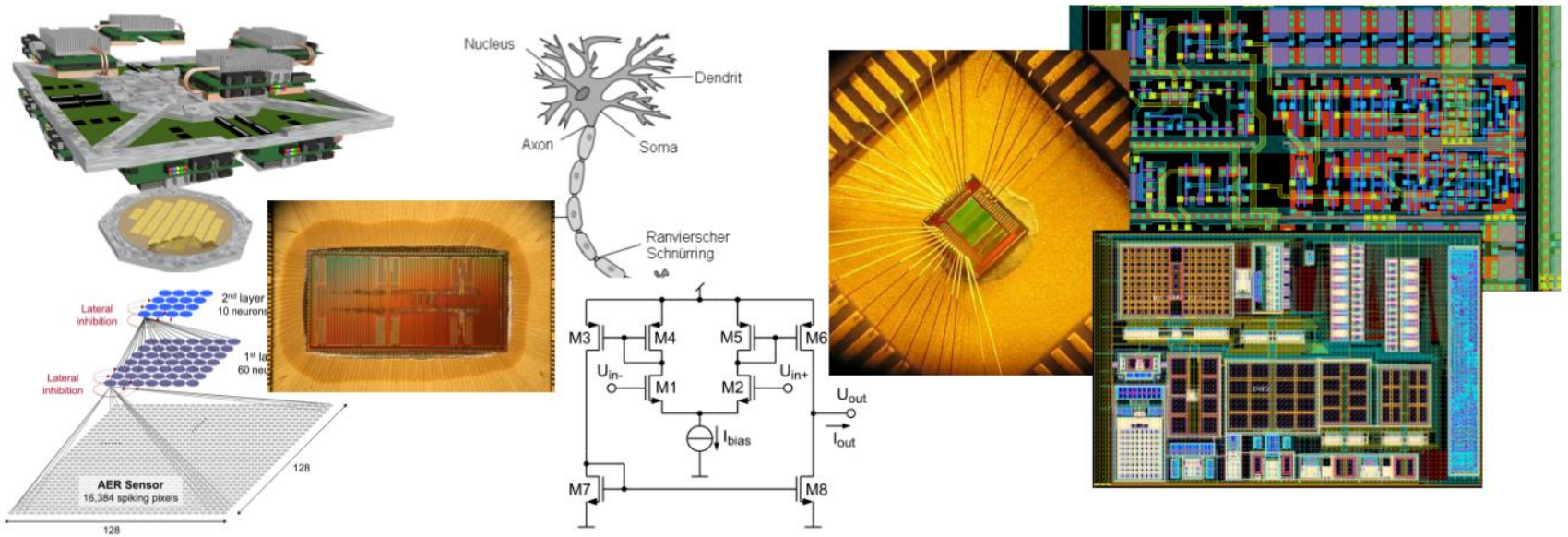
- Testchip: Santos
- 28nm SLP CMOS, 18mm²
- Komponenten Testchip für Neuromorphen Supercomputer
- 4 Processing Elements, Speicherinterface, schnelle Serielle I/Os

© Sebastian Höppner 2015

Modul Neuromorphic VLSI-Systems

Neuromorphe Systeme

Analoger CMOS-Schaltungsentwurf



- Grundlagen zu neuronalen Netzen und ihrer technischen Realisierung
- Integrierte analoge Schaltungen: Entwurf, Simulation, Verifikation
- Praktischer Schaltungsentwurf und Layout mit Cadence

Human Brain Project

© Sebastian Höppner 2015