technische universität
dortmund

fakultät für informatik
informatik 12

Synthese Eingebetteter Systeme

Wintersemester 2012/13

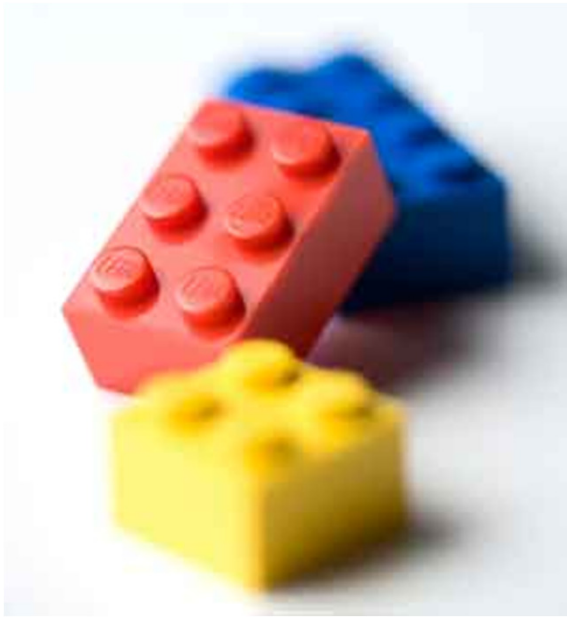
13 – Synthese: Systolische Arrays

Michael Engel
Informatik 12
TU Dortmund

2012/12/17

Synthese: Systolische Arrays

- Granularität paralleler Systeme
- Systolische Felder
- Systolische Instruktionsfelder



Granularität paralleler Systeme

- "Klassische" Hardwaresynthese erlaubt die Erzeugung feingranularer Parallelität
  - Aufwendig
- Basiselemente haben meist geringe Komplexität
  - Gatter, Flip-Flops, Addierer…
- Lässt sich Parallelität auch effizient mit groberer Granularität realisieren?

  Gatter → ??? → Multicores → verteilte Systeme

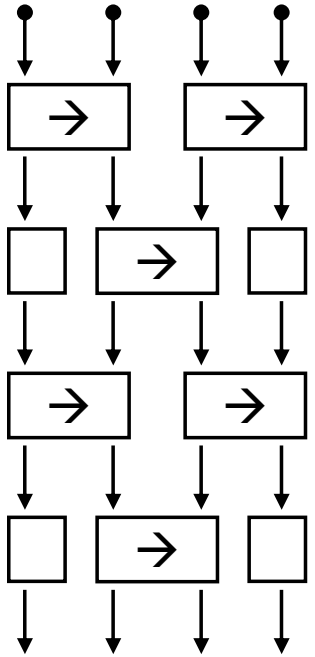
- Systolische Felder
  - Ersetzung eines Prozessors durch Array gleichartiger Verarbeitungskomponenten

Systolische Felder

- Ein systolisches Feld (*systolic array*) ist ein anwendungs-spezifisches paralleles System aus einigen wenigen einfachen Zelltypen
  - Zellen sind regulär und lokal verbunden [☞ H.T. Kung]

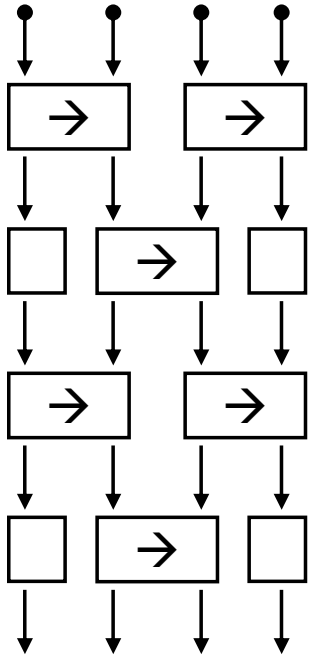
Beispiel:

- Parallele Eingabe:
- Einfache Zellen („CondSwap“, „Nop“):
- Reguläre Struktur:



Systolische Felder

- Systolische Arrays sind datenstromgetrieben
  - Verwaltung von Datenzählern
  - Gegenentwurf zur von-Neumann-Architektur
- Funktion der Zellen
  - auch “DPU” (Data Processing Unit) genannt
  - Ähnlich zu einer CPU
  - Aber: Kein Programmzähler (PC)
- *Transport triggered* (Datenstromgetrieben)
  - Zelle leitet Ergebnis direkt nach Ende der Berechnung an Folgezellen weiter

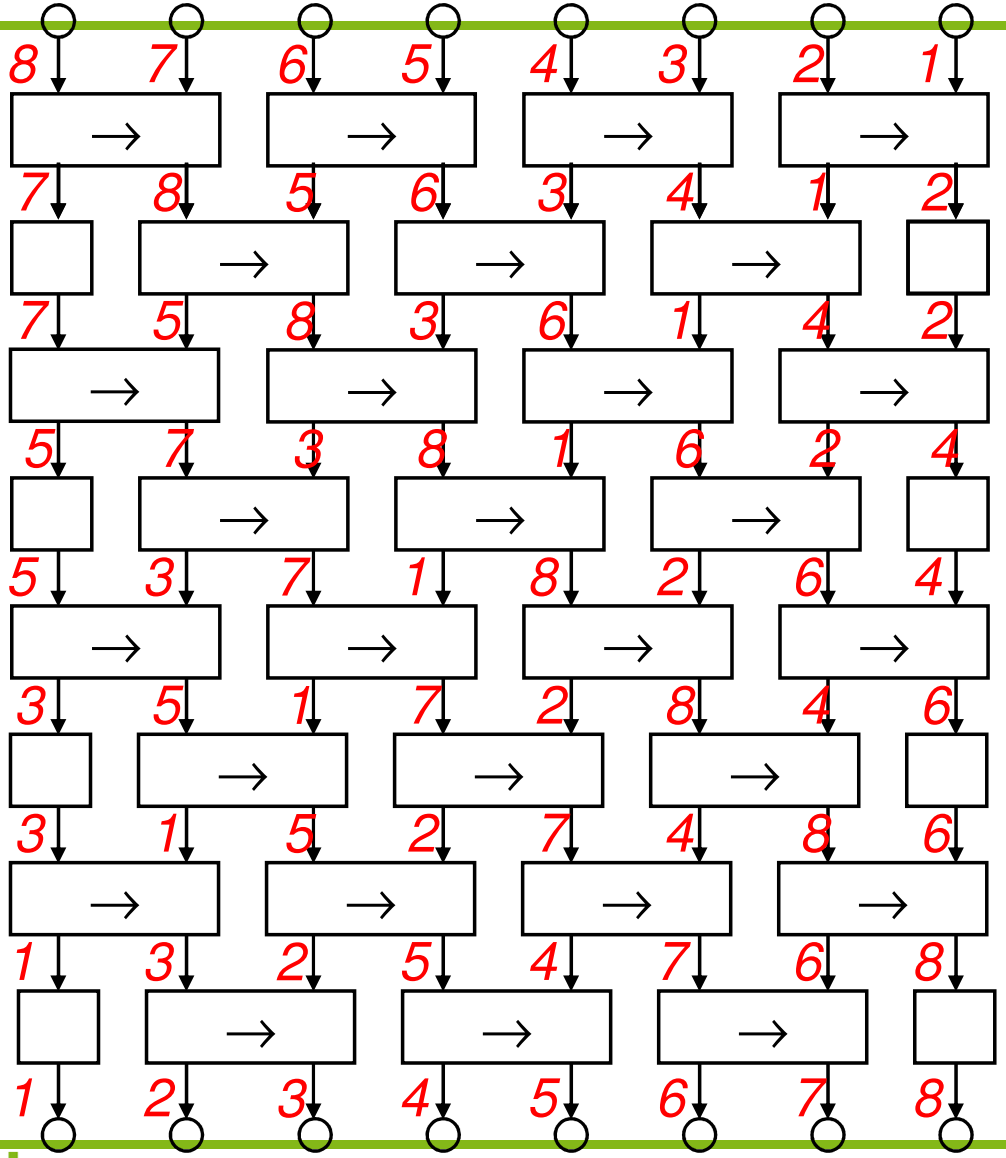


Systolische Felder

- Vergleich mit anderen parallelen Strukturen

- Systolische Felder ↔ Pipelining
  - Nicht-lineare Array-Strukturen
  - Datenfluss in mehrere Richtungen
  - Jede Zelle kann (kleine) lokale Daten/Befehlsspeicher besitzen
- Systolische Felder ↔ SIMD
  - Jede Zelle kann eine unterschiedliche Funktion ausführen

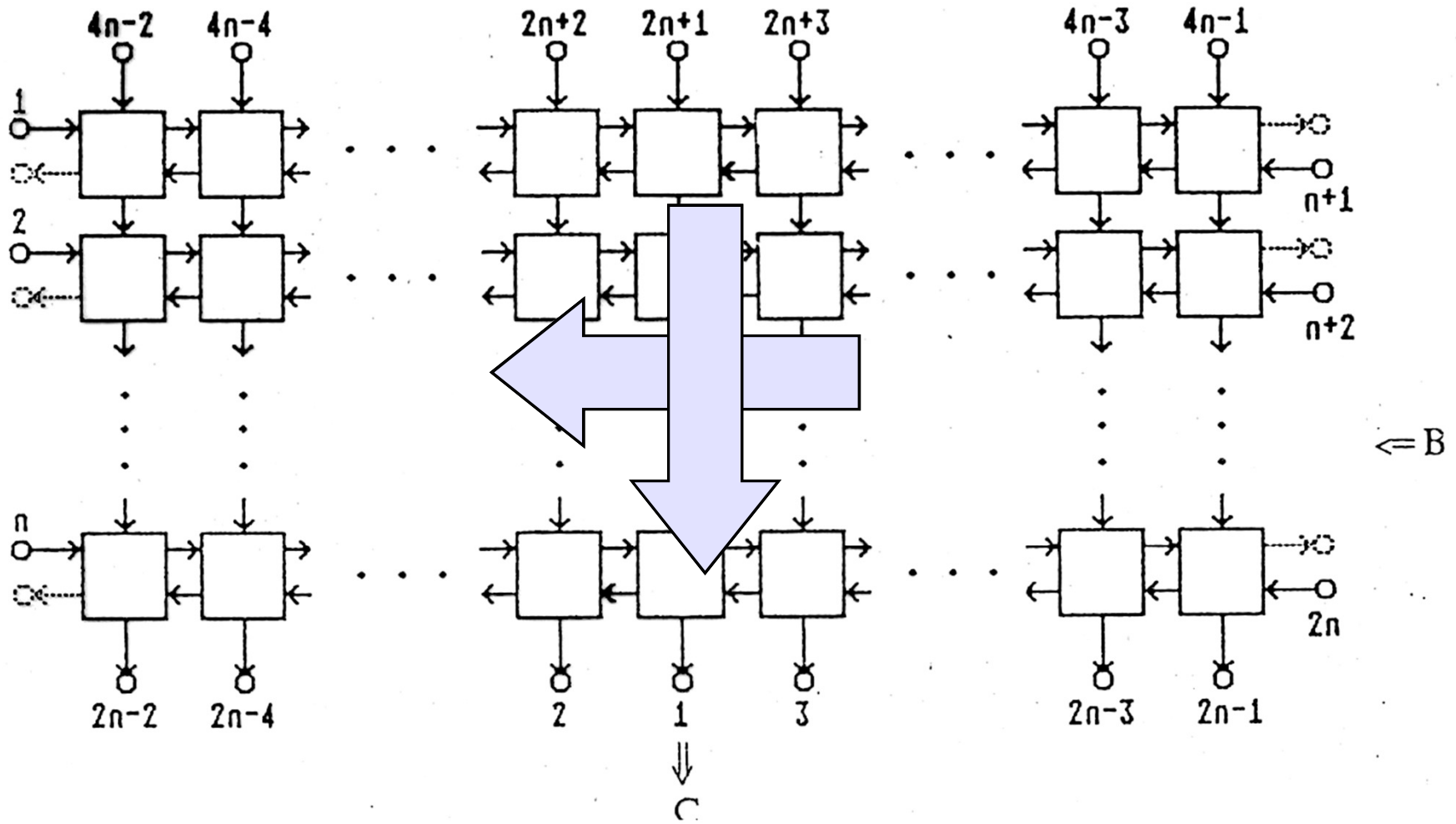
Beispiel für ein systolisches Feld: *odd-even transposition sort (OETS), n=8*



- Daten werden synchron durch ein- oder zweidimensionale Felder "gepumpt“. Dabei werden auf diesen Daten Berechnungen ausgeführt.

Prozessoren führen feste Befehle aus.

Behauptung:
systolisches Array für Matrixmultiplikation



Wie kann man systolische Arrays systematisch konstruieren?

Systolisches Array für Matrixmultiplikation

- Beispiel: 3x3-Matrixmultiplikation

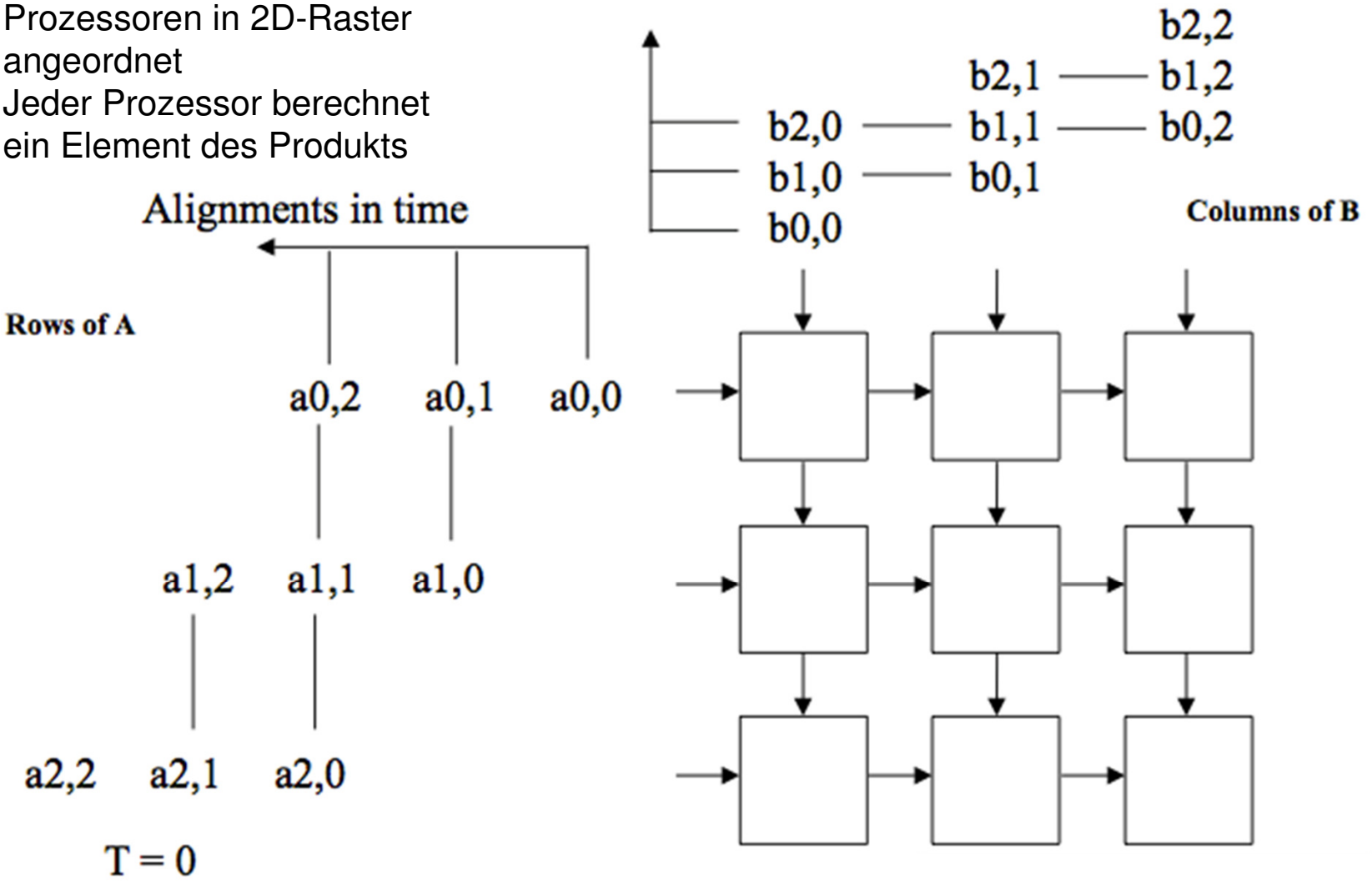
$$\begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix} \cdot \begin{pmatrix} j & m & p \\ k & n & q \\ l & o & r \end{pmatrix}$$

$$\Updownarrow$$

$$\begin{pmatrix} (aj+dk+gl) & (am+dn+go) & (ap+dq+gr) \\ (bj+ek+hl) & (bm+en+ho) & (bp+eq+hr) \\ (cj+fk+il) & (cm+fn+io) & (cp+fq+ir) \end{pmatrix}$$

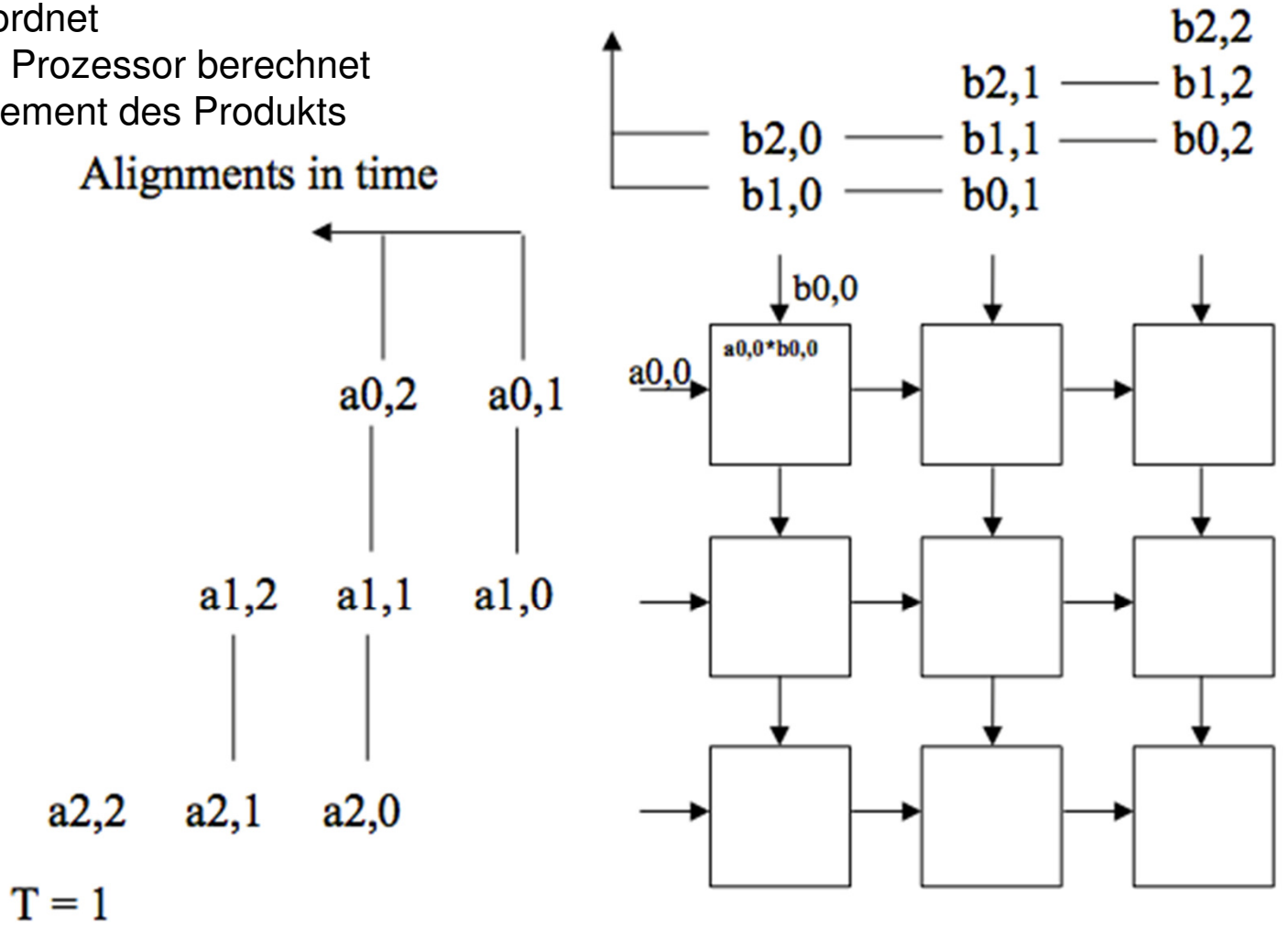
Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



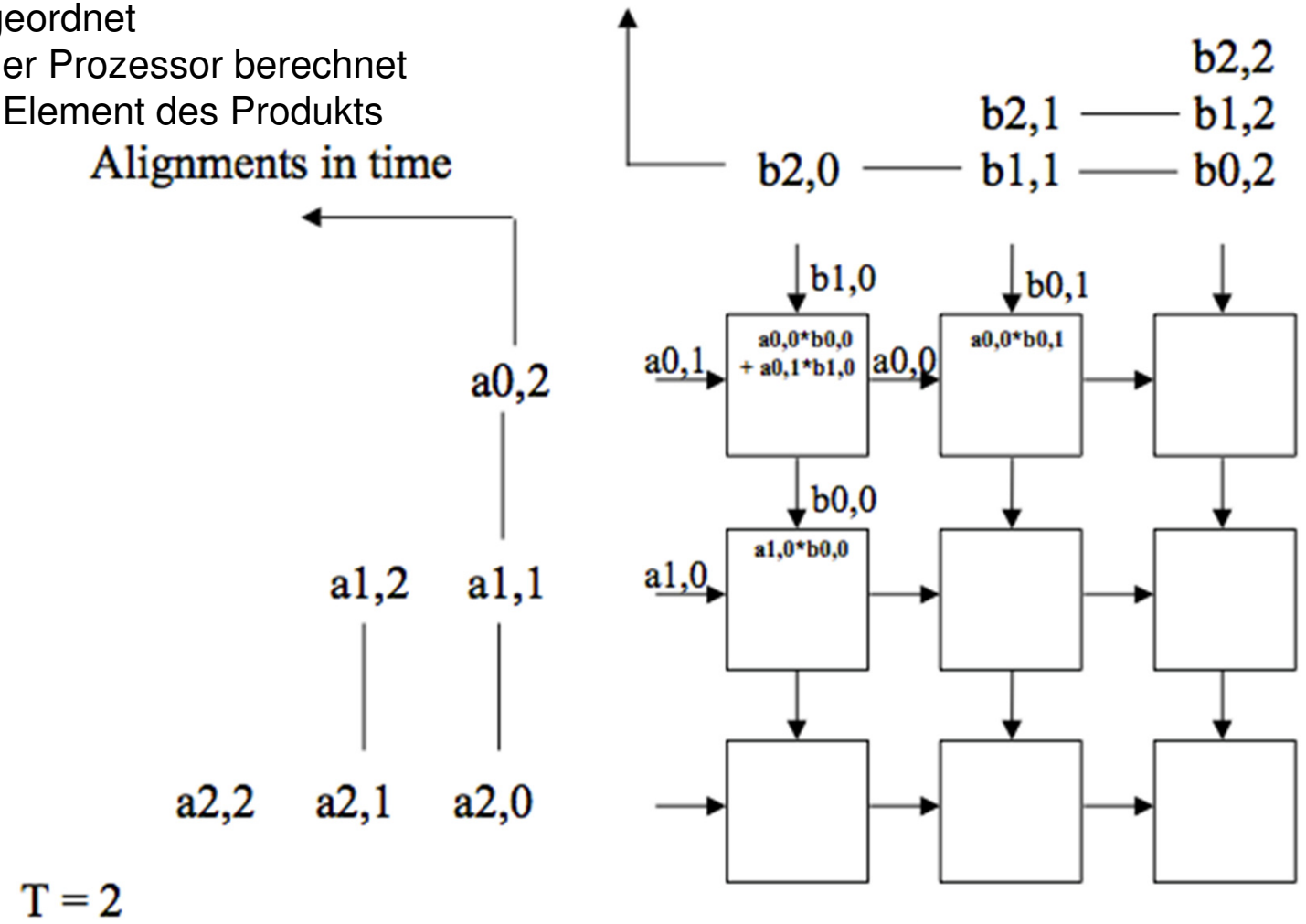
Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



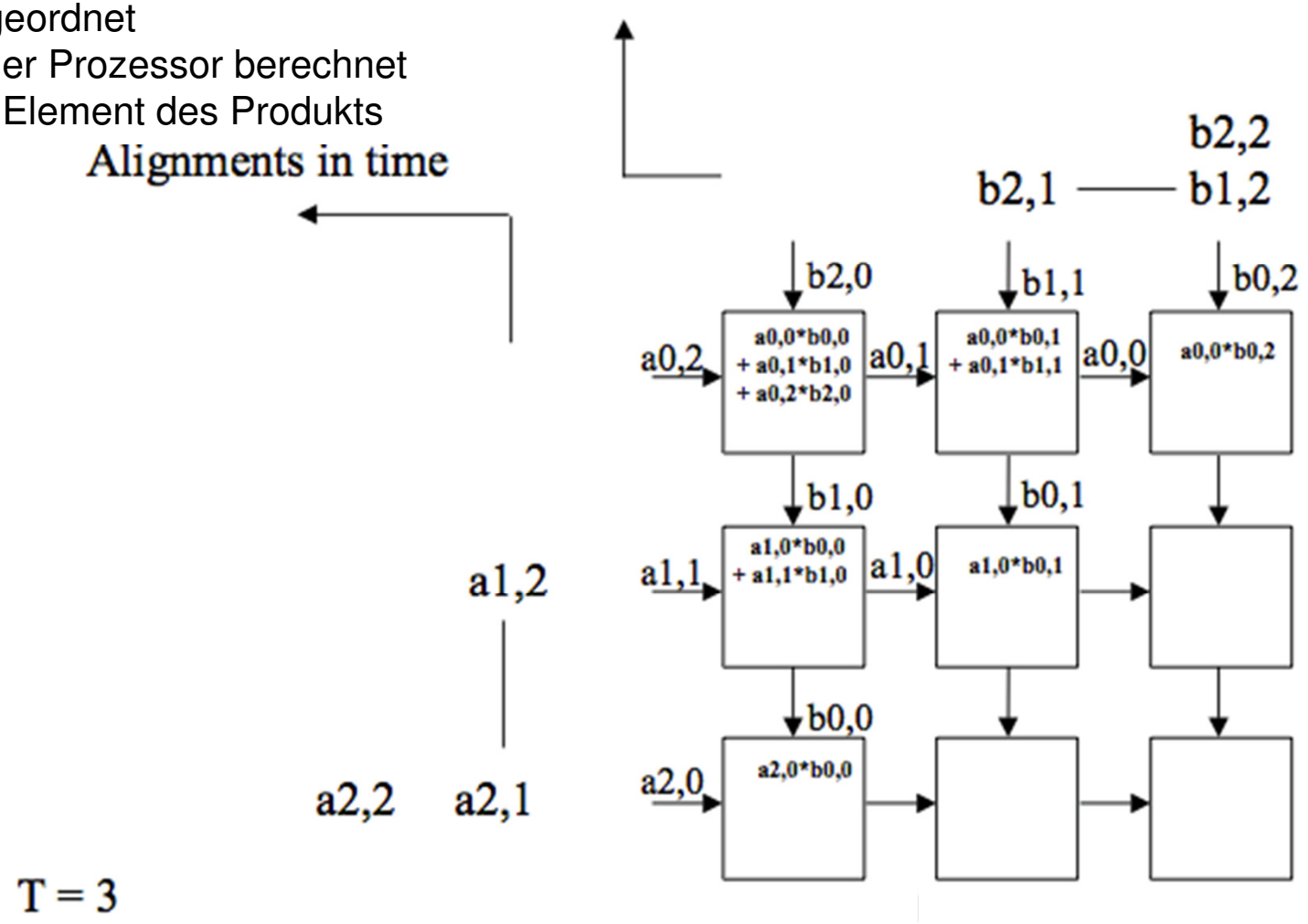
Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



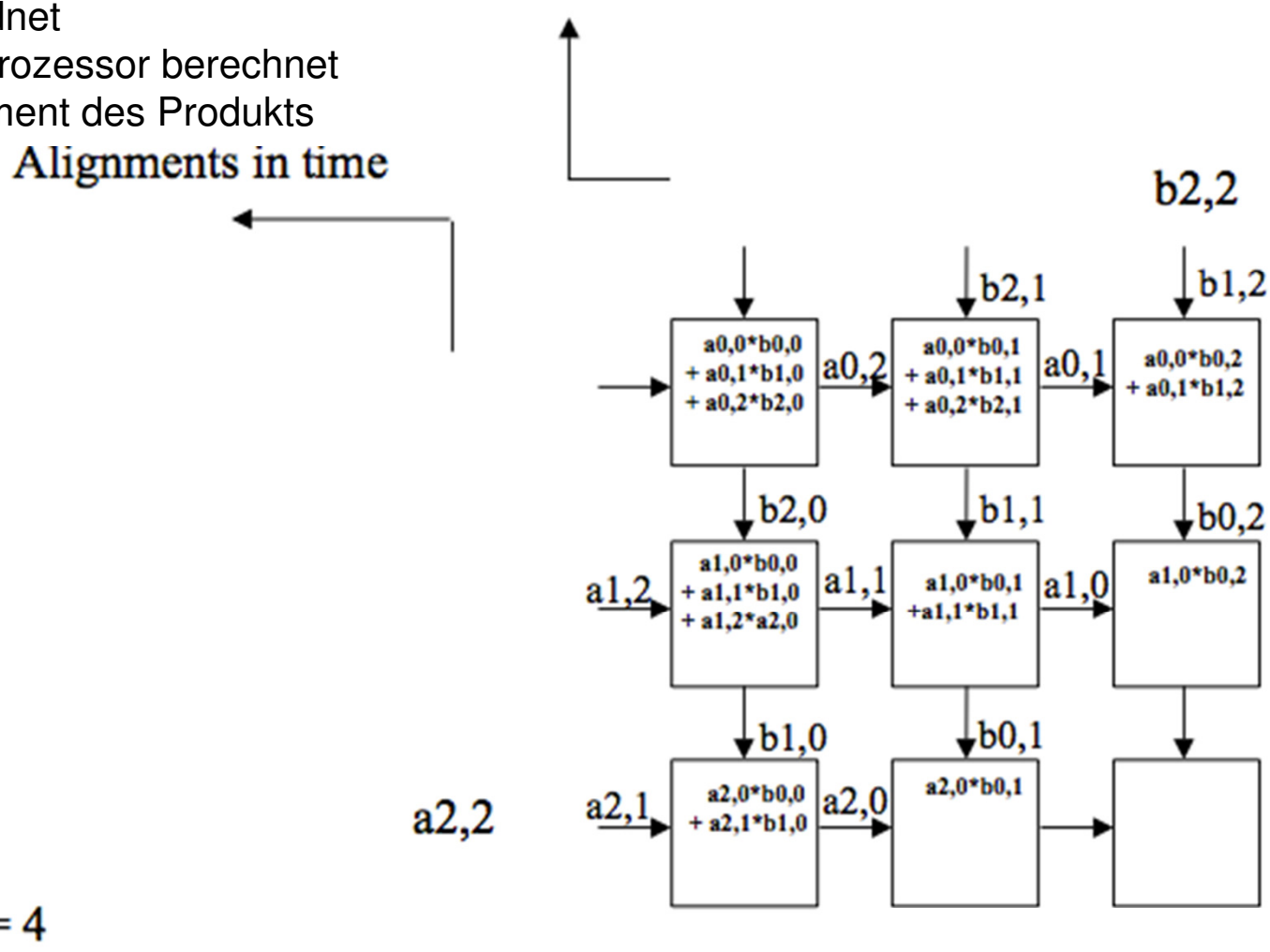
Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



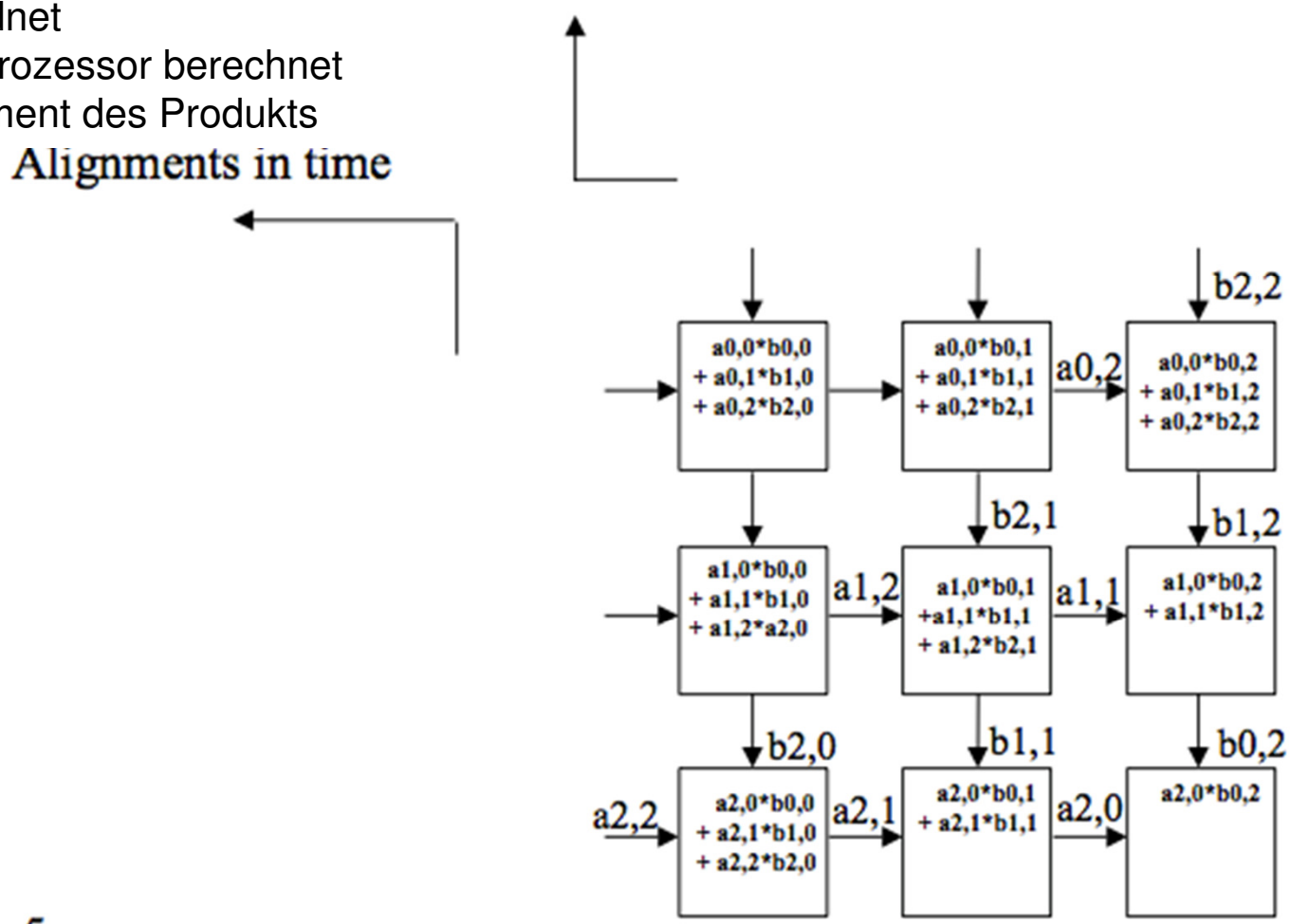
Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



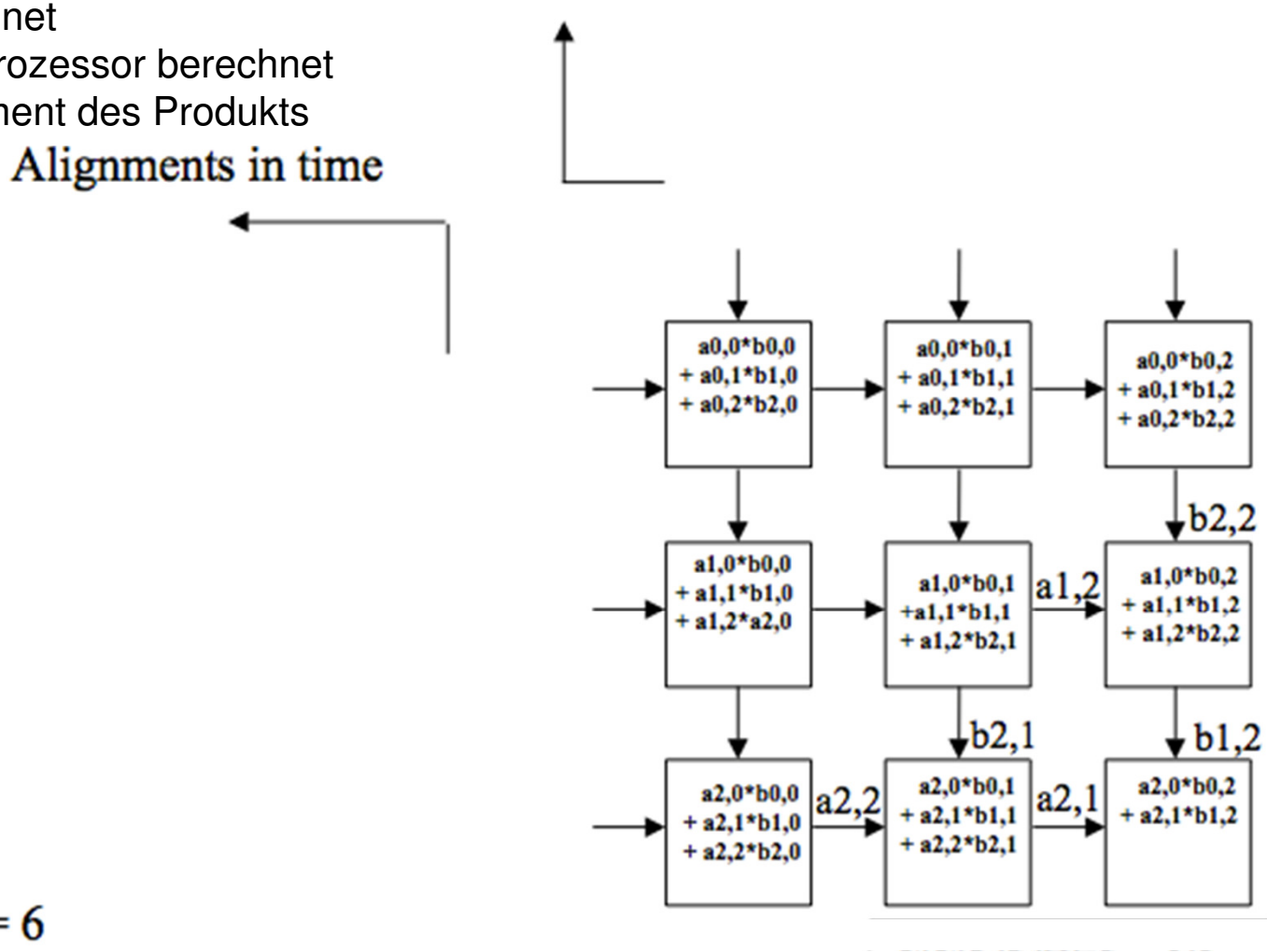
Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



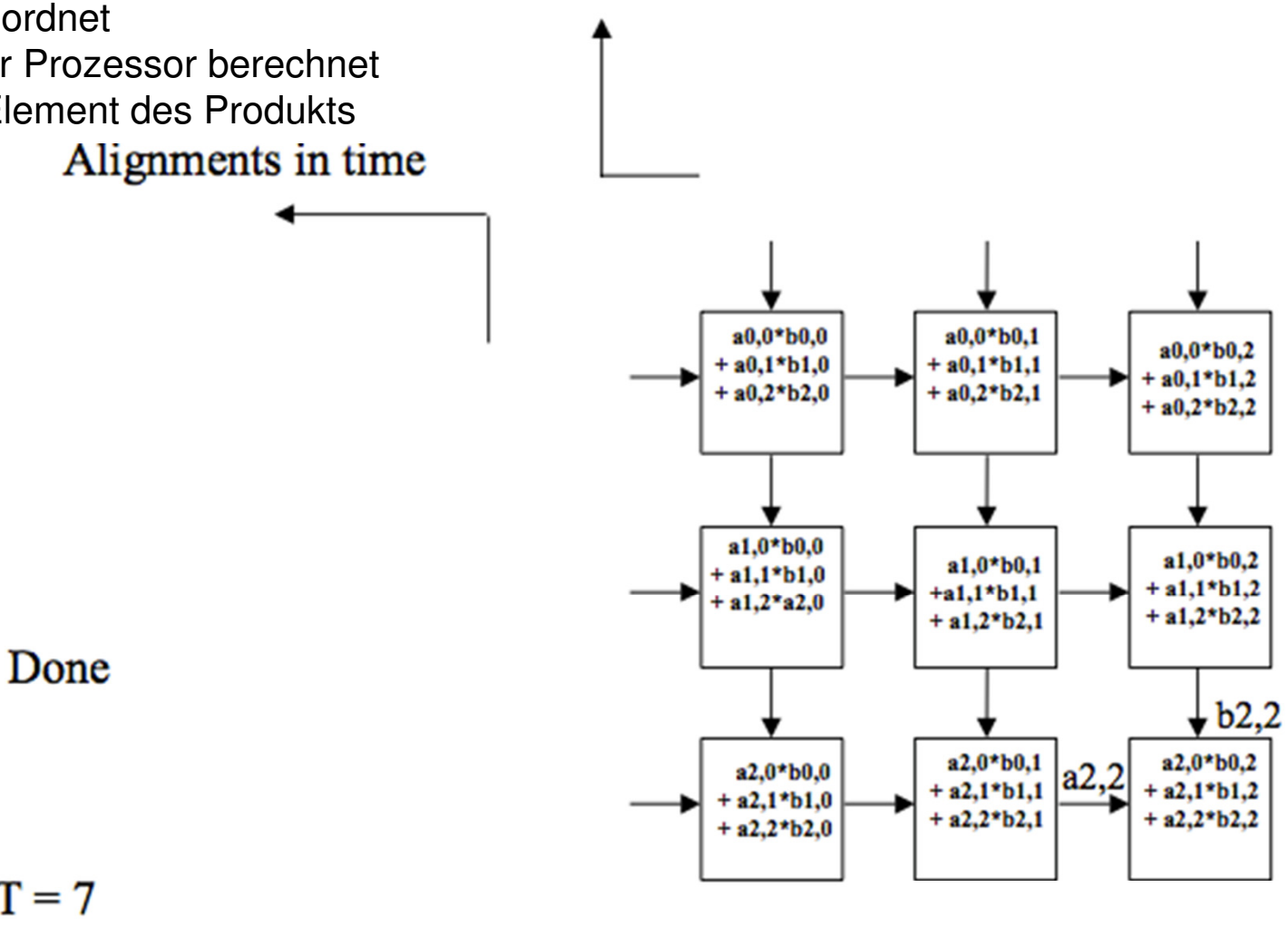
Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



Systolisches Array für Matrixmultiplikation

- Prozessoren in 2D-Raster angeordnet
- Jeder Prozessor berechnet ein Element des Produkts



Ansatz zur Konstruktion: Vorgabe von Rekursionsformeln

- Beispiele:

1. $$n! = \begin{cases} 1, & \text{falls } n = 1 \\ n * (n-1)!, & \text{falls } n \neq 1 \end{cases}$$

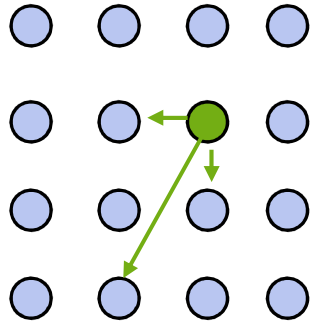
2. $$He_{n+1}(x) = x * He_n(x) - n * He_{n-1}(x)$$

☞ P. Quinton: Automatic Synthesis of Systolic Arrays from Recurrent Equations, Ann. Symp. On Computer Architecture, 1984, S. 209 ff

Voraussetzung der Methode von Quinton

- Rekursionsgleichungen der Form

$$U_1(z) = f(U_1(z-m_1),...,U_p(z-m_p))$$
$$U_2(z) = U_2(z-m_2)$$
...
$$U_p(z) = U_p(z-m_p) \text{ mit } z \in D \subseteq \mathbb{Z}^n$$



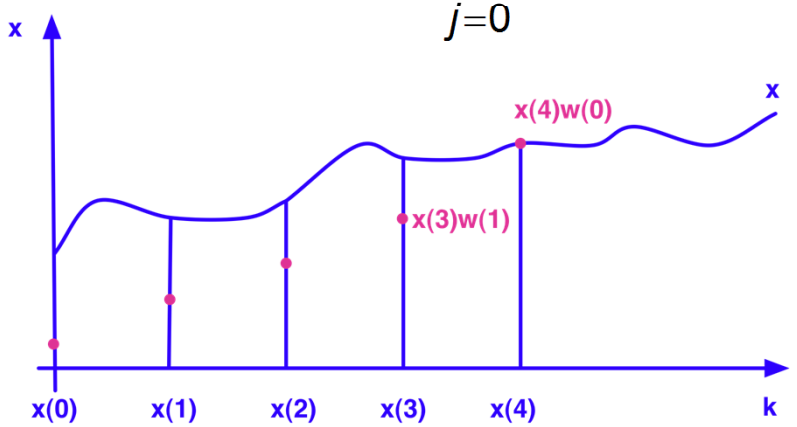
Raum D von Berechnungen

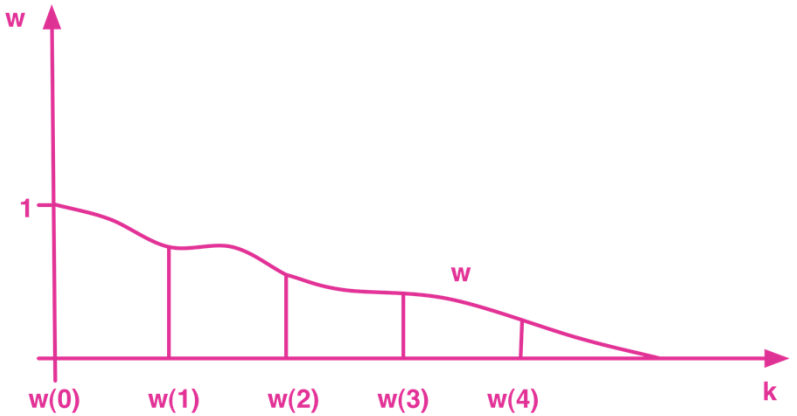
Die Vektoren m_j mit $j \in \{1..p\}$ heißen Abhängigkeitsvektoren
Die Knoten aus dem Gebiet $D \subseteq \mathbb{Z}^n$ zusammen mit den Kanten $M = \{m_1, \ldots, m_p\}$ bilden den Abhängigkeitsgraphen $G = (D,M)$.
$x \in D$ heißt abhängig von $y \in D$ falls $\exists\, i : x = y - m_i$.

Beispiel: Faltung

Faltung eines Vektors mit einem Vektor von Gewichten

$$\forall i \geq 0 : y(i) = \sum_{j=0}^{K} w(j) * x(i-j)$$





Vergleich von aktueller Form und Ziel

Ziel:

$$U_1(z) = f(U_1(z-m_1),...,U_p(z-m_p))$$
$$U_2(z) = U_2(z-m_2)$$
...
$$U_p(z) = U_p(z-m_p) \text{ mit } z \in D \subseteq \mathbb{Z}^n$$

Jetzt:

$$y(i) = \sum_{j=0}^{K} w(j) * x(i-j)$$

Erweiterung von $y(i)$ zu $Y(i,k)$

Def.: $$\forall 0 \le k \le K : Y(i,k) = \sum_{j=0}^{k} w(j) * x(i-j)$$

$$\Rightarrow \forall i : \forall 0 \le k \le K : \boxed{Y(i,k) = Y(i,k-1) + w(k) * x(i-k)}$$

mit $\forall i : Y(i,-1) = 0, \; Y(i,K) = y(i)$ ☞ $D \subseteq \mathbb{Z}^2$

2-dimensionale Beschreibung von w und x:

Def.: $$\forall k : W(-1,k) = w(k)$$

$$\forall i \ge 0 \; \forall k : \boxed{W(i,k) = W(i-1,k)}$$

Def.: $$\forall i \ge 0 : X(i,-1) = x(i)$$

$$\forall i \ge 0 \; \forall 0 \le k \le K : \boxed{X(i,k) = X(i-1,k-1)}$$

$$\forall k, 0 \le k \le K : X(-1,k-1) = 0$$

2-dimensionale Form der Rekursionsgleichungen

$\Rightarrow \forall i \geq 0, \forall k, 0 \leq k \leq K:$

$$Y(i,k)=Y(i,k\text{-}1)+W(i\text{-}1,k)*X(i\text{-}1,k\text{-}1)$$

$$W(i,k)=W(i\text{-}1,k)$$

$$X(i,k)=X(i\text{-}1,k\text{-}1)$$

$m_1=(0,1)$
$m_2=(1,0)$
$m_3=(1,1)$

Mit

$\forall i \geq 0:$ $\quad Y(i,\text{-}1)=0$

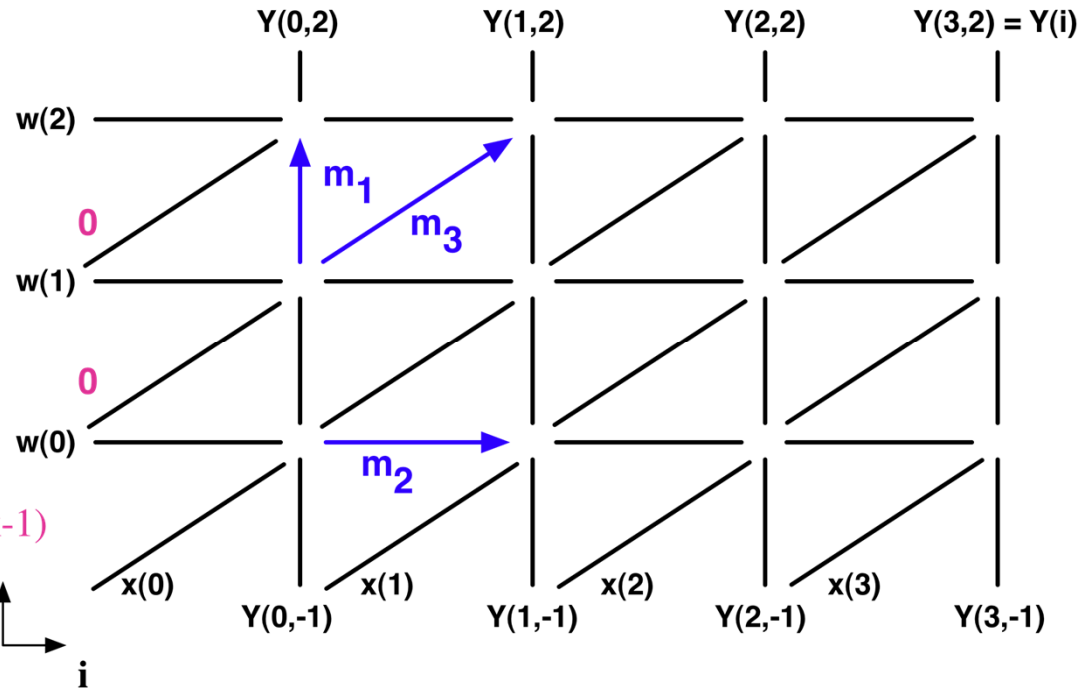
$\forall i \geq 0:$ $\quad X(i\text{-}1,\text{-}1)=x(i)$

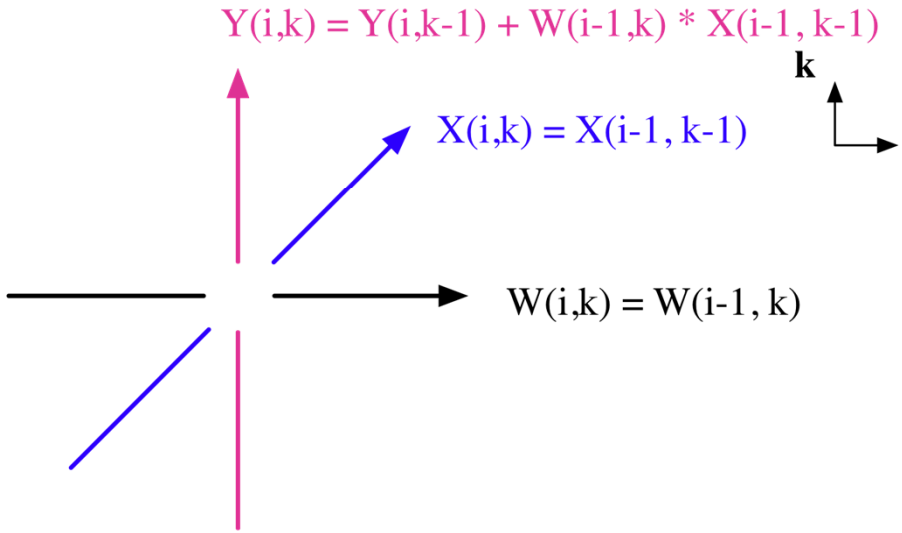
$\forall k, 0 \leq k \leq K:$ $\quad W(\text{-}1,k)=w(k)$

$\forall k, 0 \leq k \leq K:$ $\quad X(\text{-}1,k\text{-}1)=0$

Abhängigkeitsgraph für das Faltungsbeispiel

- Mit $m_1=(0,1)$, $m_2=(1,0)$, $m_3=(1,1)$ ergibt sich der Abhängig-keitsgraph.

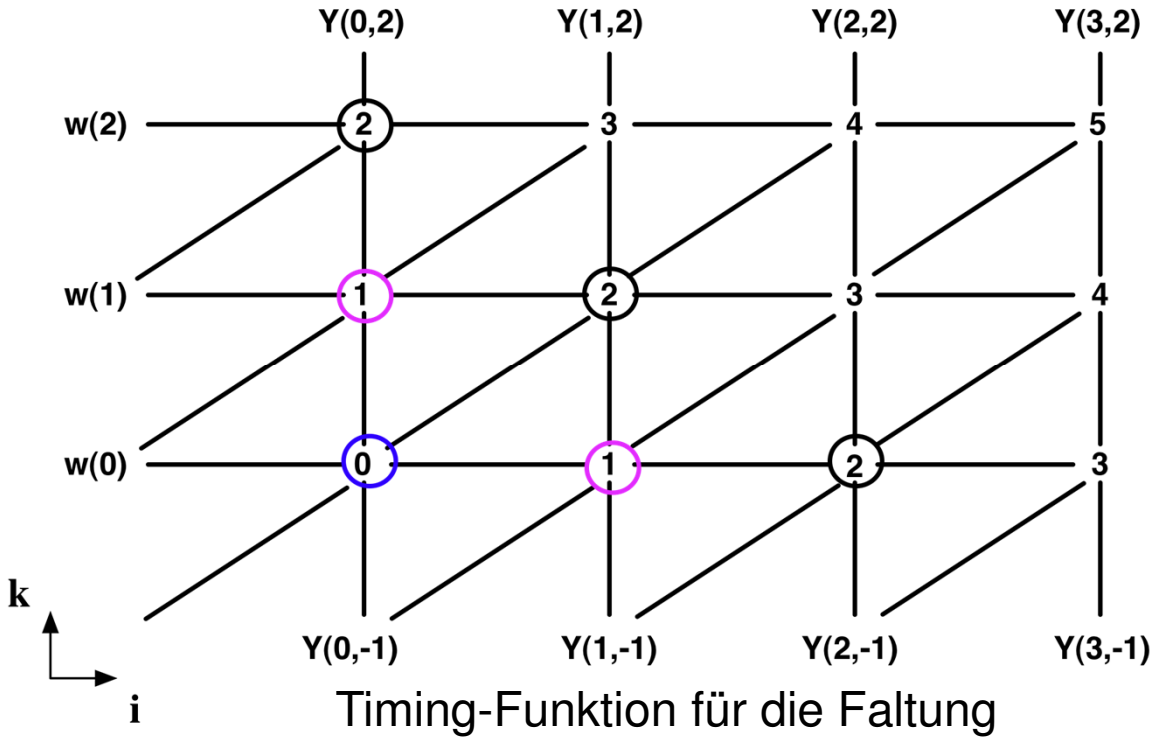




Suche nach einer Timing-Funktion (Scheduling)

- Zuordnung τ: Operation $\rightarrow$ Ausführungszeiten.
- Bedingungen an τ: $\tau: \mathbb{Z}^n \rightarrow \mathbb{Z}$:
  - $\forall u \in D : \tau(u) \geq 0$ und
  - $\forall u,v \in D : u$ abhängig von $v \Rightarrow \tau(u) > \tau(v)$

Im Beispiel: Die Berechnung von $Y(2,2)$ wird eine Zeiteinheit (Takt) nach den Berechnungen von $Y(1,2)$ und $Y(2,1)$ sowie zwei Zeiteinheiten nach der Berechnung von $Y(1,1)$ ausgeführt.

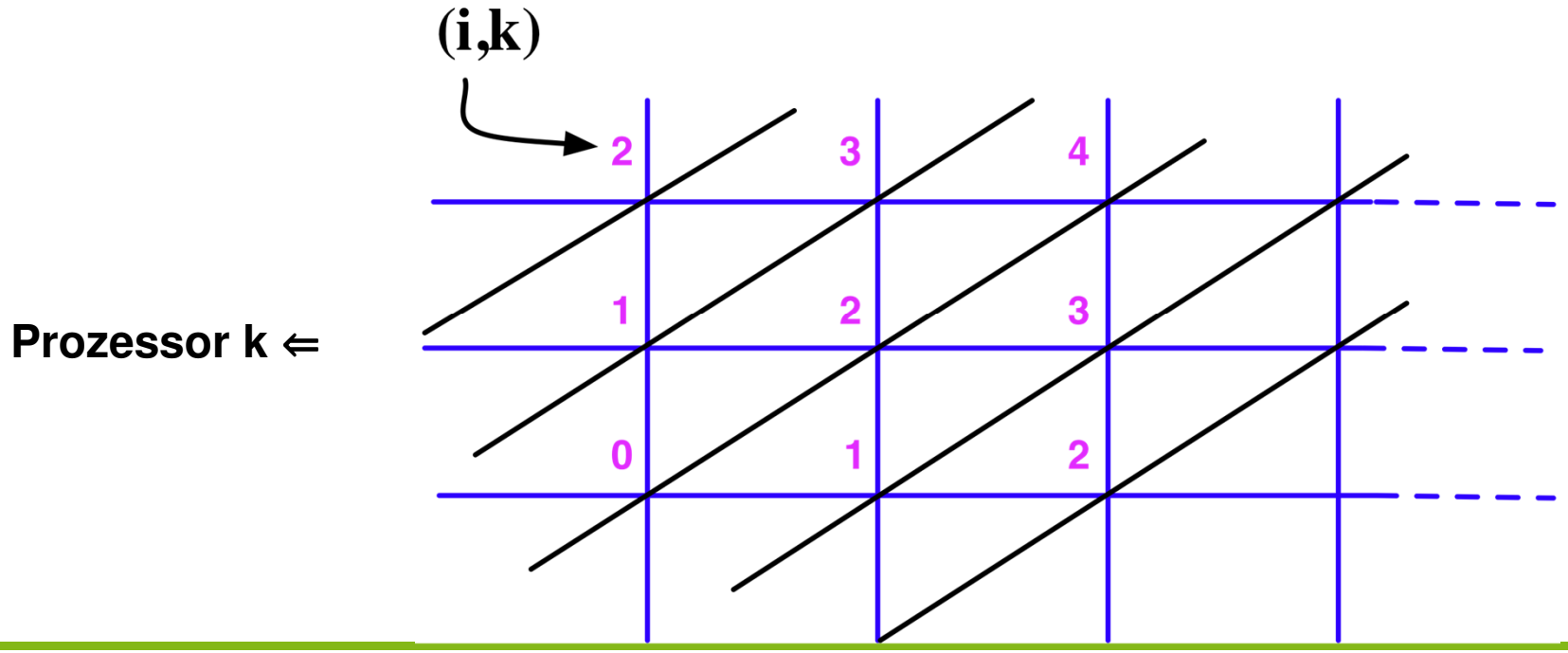


Timing-Funktion für die Faltung

Suche nach einer Prozessorzuordnung

$$a: \mathbb{Z}^n \rightarrow \mathbb{Z}^m \text{ mit } \forall x,y: a(x) = a(y) \underset{x \neq y}{\Rightarrow} t(x) \neq t(y)$$

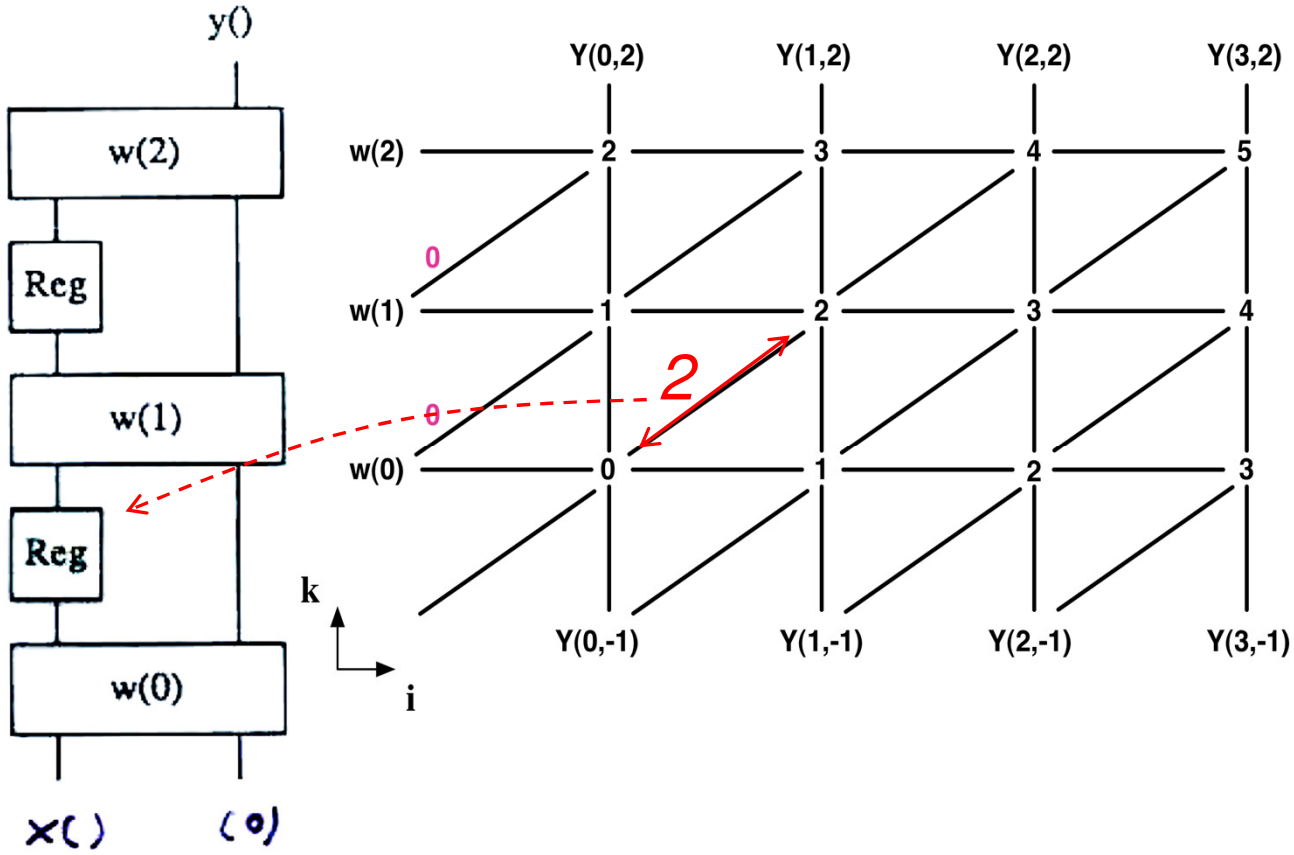
z.B. $\forall i, k: a(i, k) = (k, 0)$



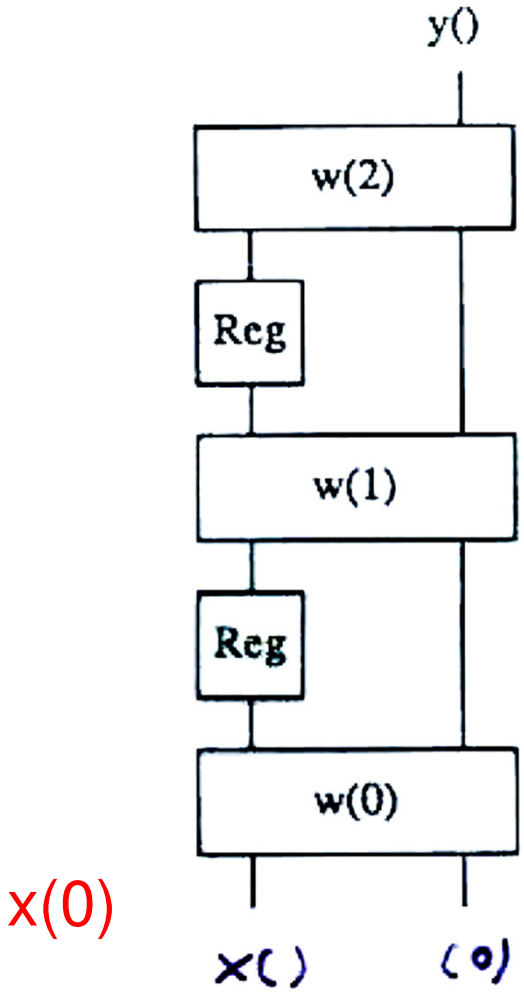
Bestimmung von Verbindungen und Registern

Für die Verzögerung des Stroms der Werte werden Register benötigt.

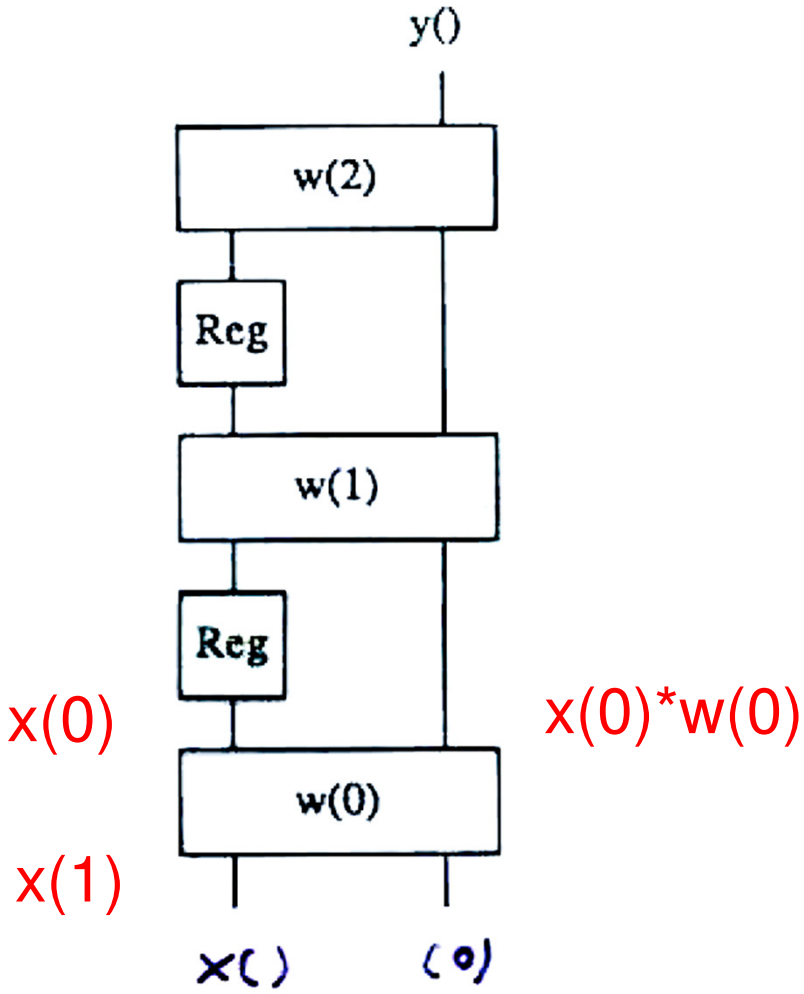
- *x*-Werte: verzögerte Weiterleitung in der Diagonalen
- *w*-Werte: fest in Prozessoren gespeichert
- *y*-Werte: werden unverzögert vertikal weitergeleitet.



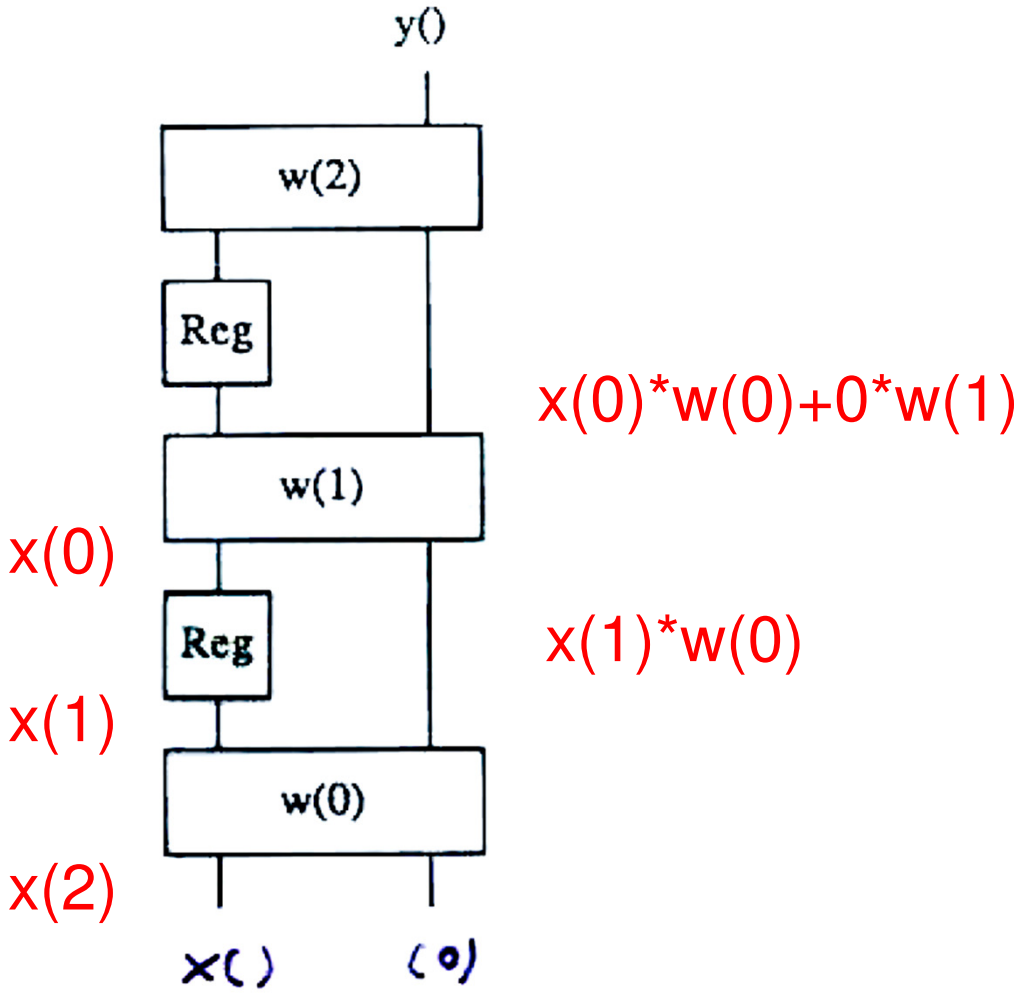
Ablauf (1)



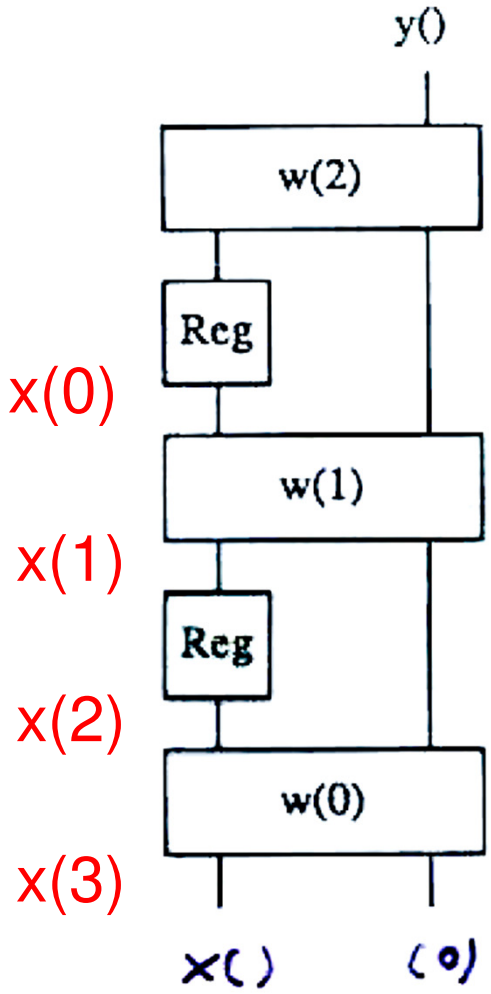
Ablauf (2)



Ablauf (3)



Ablauf (4)



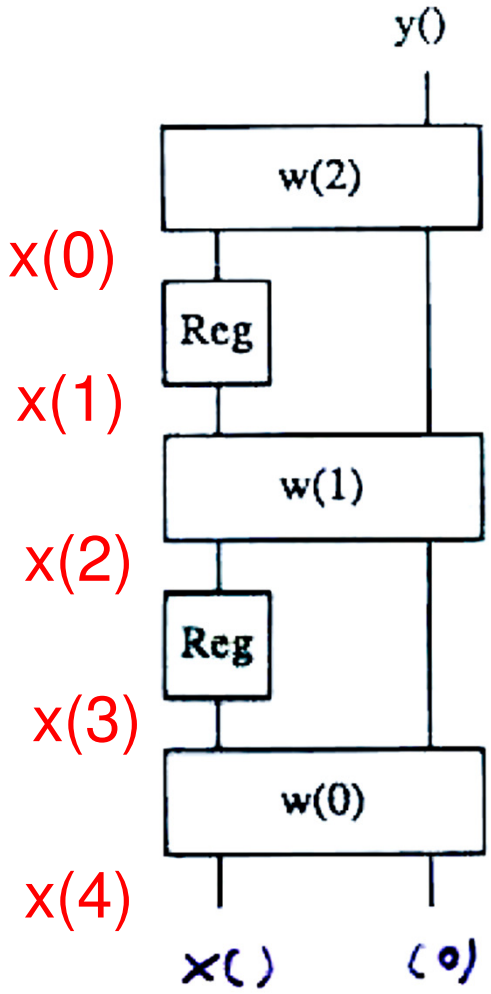
x(0)*w(0)+0*w(1)+0*w(2)

x(1)*w(0)+x(0)*w(1)

x(2)*w(0)

Ablauf (5)

x(0)*w(0)+0*w(1)+0*w(2)



x(1)*w(0)+x(0)*w(1)

x(2)*w(0)+x(1)*w(1)

x(3)*w(0)

Ablauf (6)

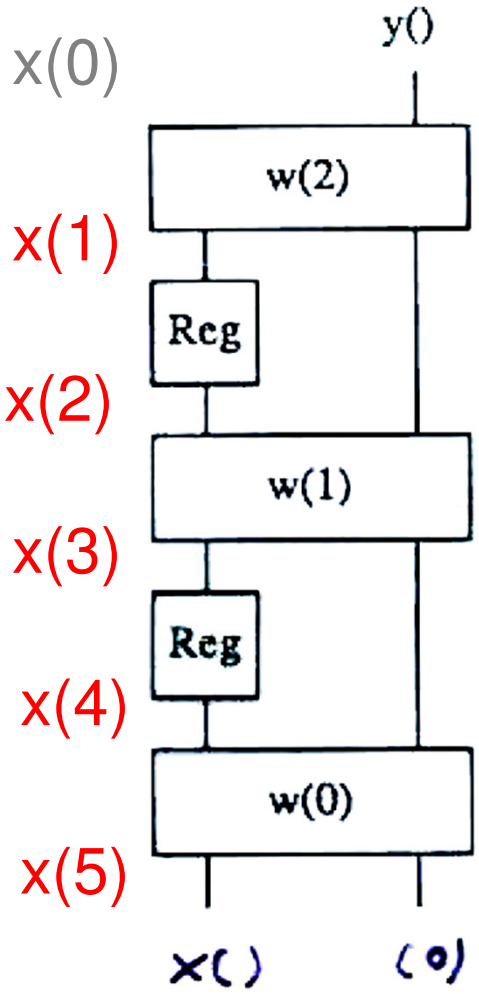
x(0)*w(0)+0*w(1)+0*w(2)

x(1)*w(0)+x(0)*w(1)

x(2)*w(0)+x(1)*w(1)+x(0)*w(2)

x(3)*w(0)+x(2)*w(1)

x(4)*w(0)



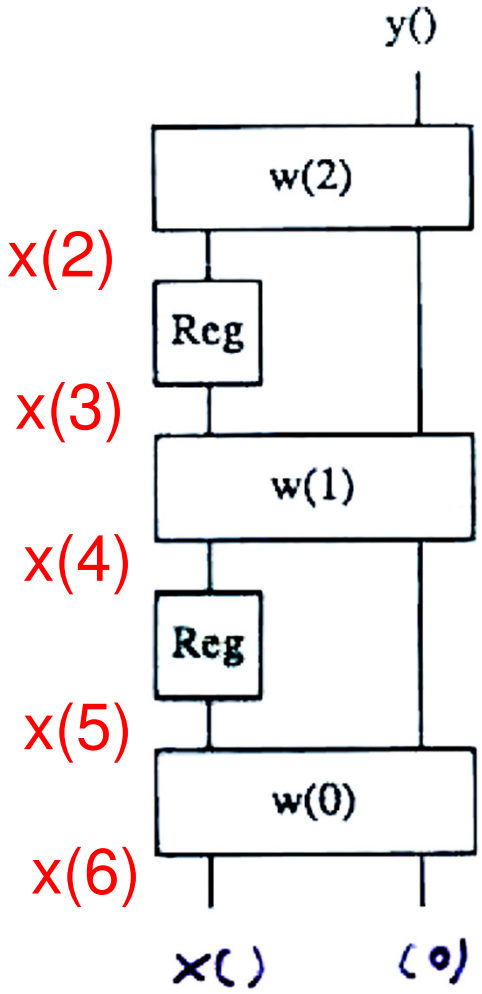
Ablauf (7)

x(0)*w(0)+0*w(1)+0*w(2)

x(1)*w(0)+x(0)*w(1)

x(2)*w(0)+x(1)*w(1)+x(0)*w(2)

x(3)*w(0)+x(2)*w(1)+x(1)*w(2)



x(4)*w(0)+x(3)*w(1)

x(5)*w(0)

Weitere Arbeiten zu systolischen Feldern

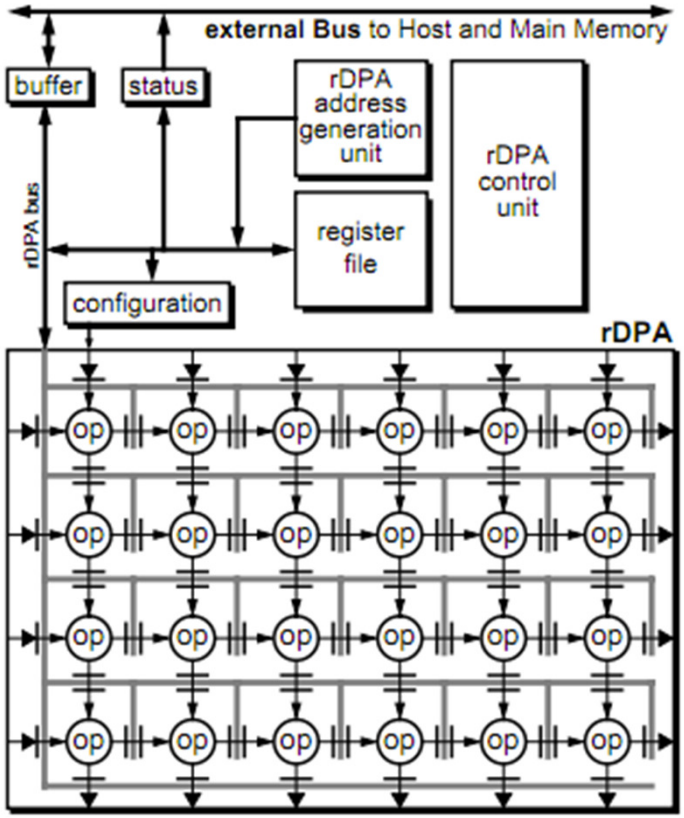
- Kung (ex-CMU), „*you just pump the data through*“; Gilt als „Vater“ der systolischen Arrays.
  Hat früher vorhandenen Ansätzen den Namen gegeben.

- Monica Lam

- Moldovan
- Hartenstein/Kress

Rekonfigurierbare systolische Arrays

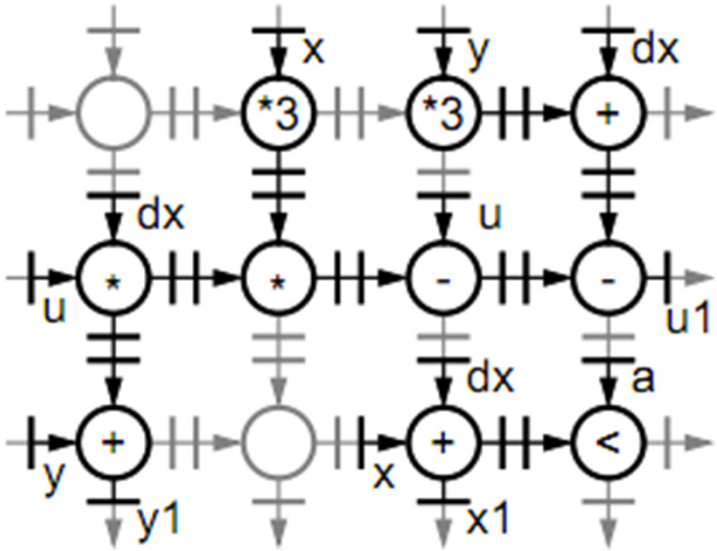
- rDPA: reconfigurable data path arrays
  - Verallgemeinerung systolischer Arrays
- Ersetzung der algebraischen Synthesemethoden durch *simulated annealing*
  - Heuristisches Optimierungs-verfahren für komplexe Probleme
  - Auch beim Chip-Layout verwendet
- Realisierung nicht gleichförmiger Strukturen möglich
  - Zickzack, Spiralen, fork/join



Rekonfigurierbare systolische Arrays

- Beispiel:

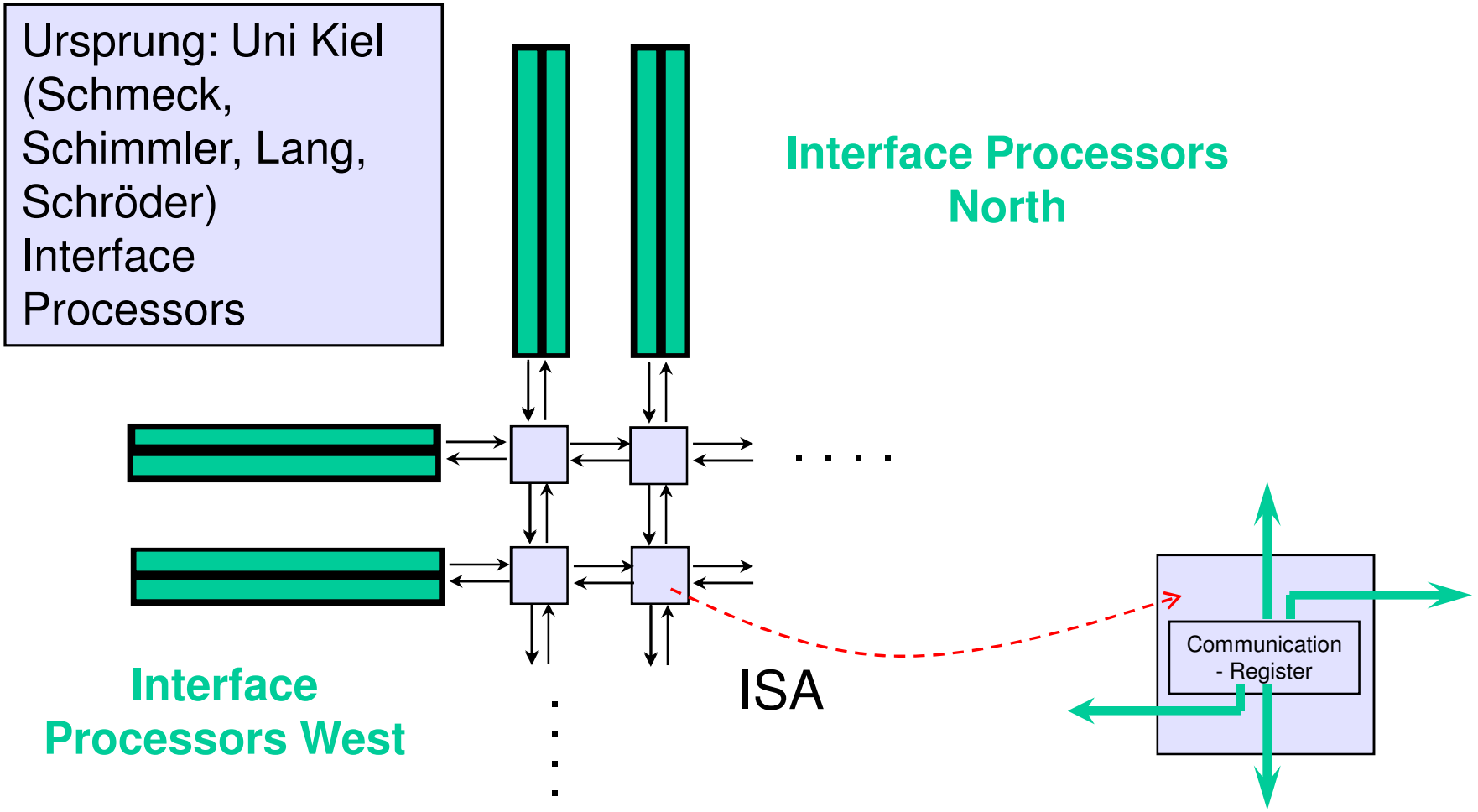
```
x1 = x + dx;
u1 = z – 3 * x * u * dx – 3 * y * dx;
y1 = y + u * dx;
c  = x1 < a;
```



R. Kress et al.: A Datapath Synthesis System for the Reconfigurable Datapath Architecture; Asia and South Pacific Design Automation Conference, ASP-DAC'95, Makuhari, Chiba, Japan, 1995

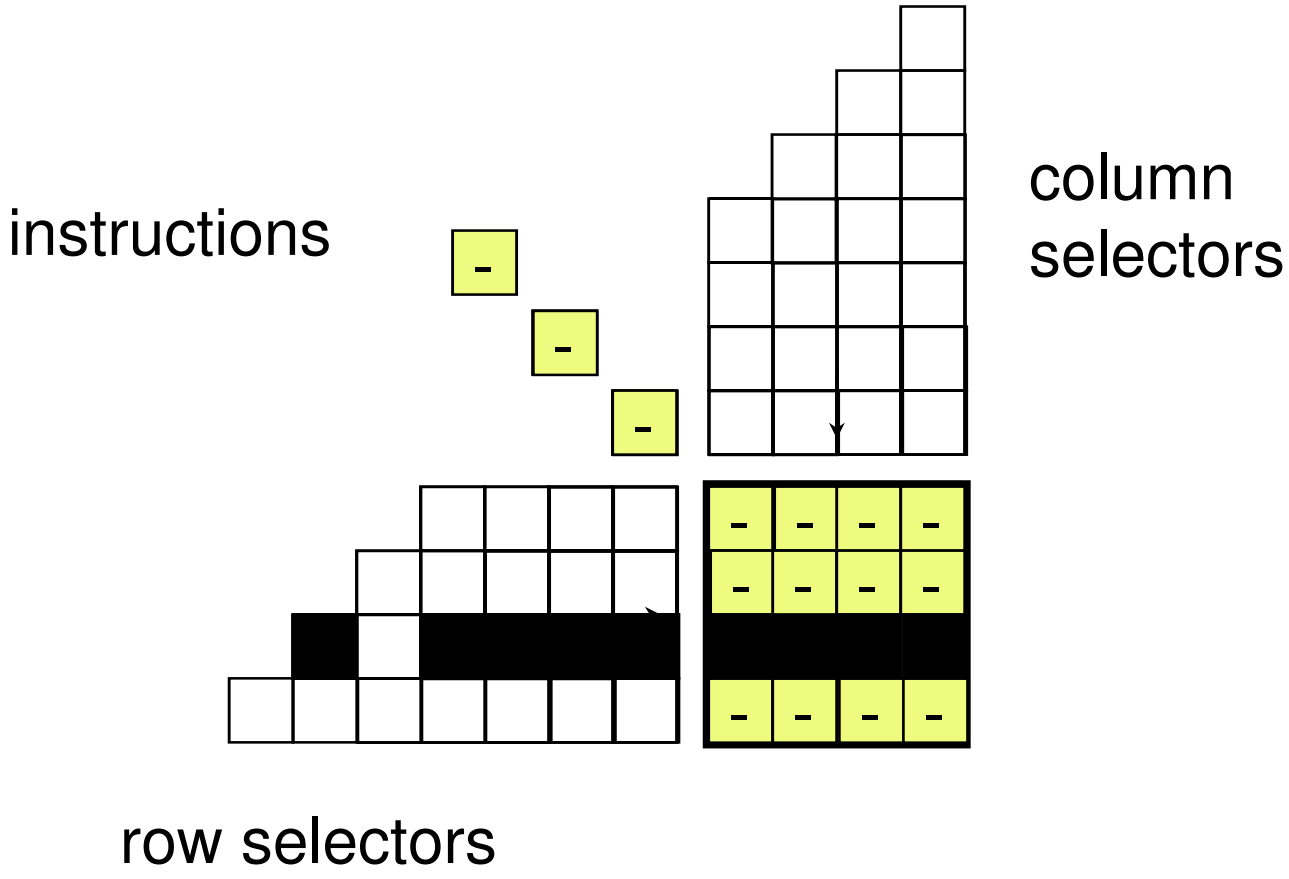
Instruction systolic array (ISA)

Ursprung: Uni Kiel (Schmeck, Schimmler, Lang, Schröder) Interface Processors



Quelle: www.cs.umd.edu/class/spring2003/ cmsc838t/slides/reddyg.**ppt**

Instruction Systolic Array

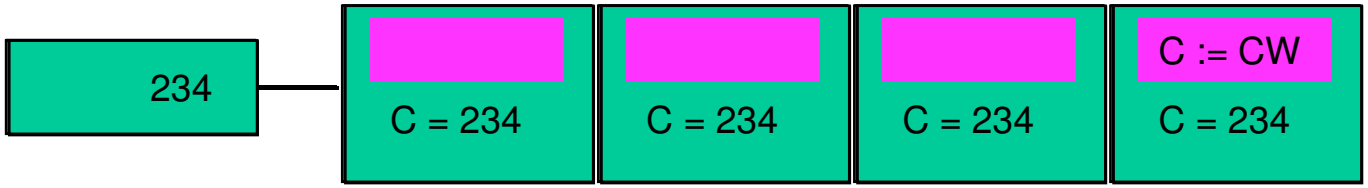


wavefront instruction execution

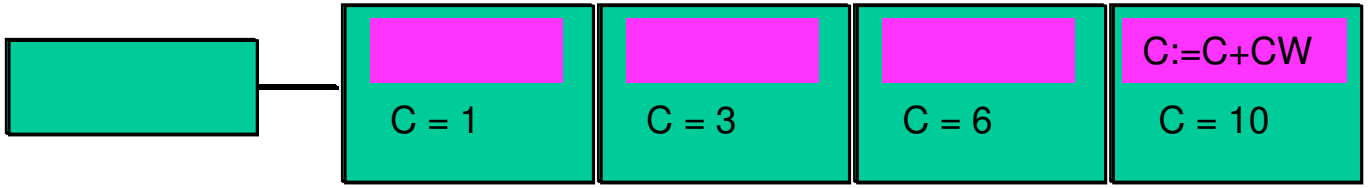
⇒ fast accumulation operations (e.g. row sum, broadcast, ringshift)

Vorteil von ISA's: Ausführung aggregierter Funktionen

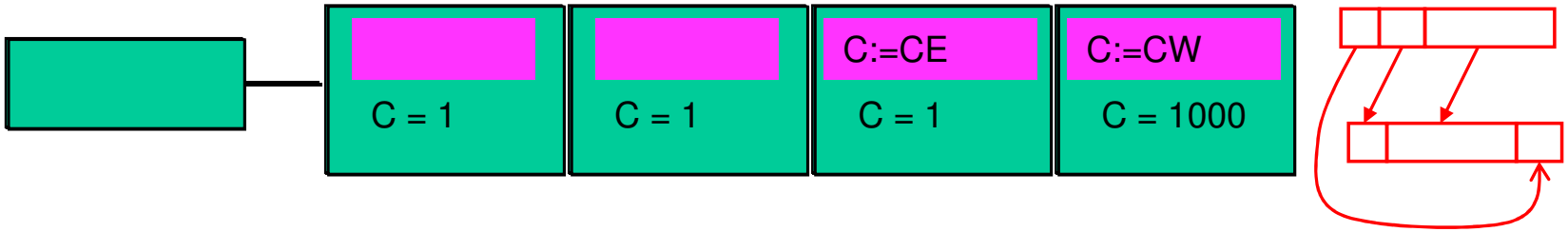
- Row Broadcast **C := C[WEST]**



- Row Sum **C := C + C[WEST]**



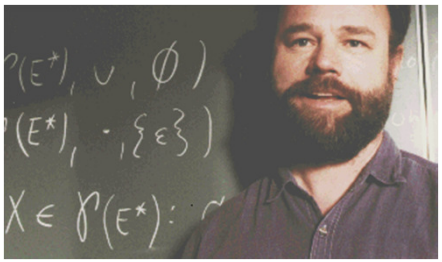
- Row Ringshift **C := C[WEST]; C:=C[EAST]**



Weiteres zu ISAs

- Weitere Infos
- Interaktive Demos:

  http://www.iti.fh-flensburg.de/lang/papers/isa/index.htm



Realisierung von systolischen Arrays mit FPGAs

Beispiele:

- Systematische Implementierung von Walsh-Hadamard Transformationen (verall. Fourier-Transformationen):
  http://ww.ll.mit.edu/HPEC/agendas/proc02/presentations/HPEC%20Day%201/Session%202/2.4fang-new.**ppt**
- Viterbi Decoder
  - Decodierung eines mit *forward error correction* codierten Bitstroms
  - mit systolischen Algorithmus in FPGA realisiert:
    http://academic.research.microsoft.com/Publication/1433687/fpga-design-and-implementation-of-a-low-power-systolic-array-based-adaptive-viterbi-decoder
- Viele weitere Implementierungen

Zusammenfassung

- Prinzip von systolischen Arrays
- Projektionsmethode von Quinton zur systematischen Erzeugung von systolischen Arrays
- Rekonfigurierbare systolische Arrays
- Schieben von Befehlen: *instruction systolic array* (ISA)
- Realisierung von systolischen Arrays mit FPGAs