Mikroarchitektursynthese

- Synthese aus funktionalen Spezifikationen -

Peter Marwedel
TU Dortmund,
Informatik 12

Übersicht

- Einführung
- SystemC
  - Vorlesungen und Programmierung
- FPGAs
  - Vorlesungen
  - VHDL-basierte Konfiguration von FPGAs mit dem XUP V II Pro Entwicklungssystem
- Synthese-Algorithmen
  - Mikroarchitektur-Synthese
    - Synthese aus funktionalen Spezifikationen
    - Synthese aus imperativen Spezifikationen
  - Logik- und Controllersynthese
  - …

Von der Spezifikation zur Implementierung

Nachteile der Simulation:

- Nicht vollständig bzgl. Eingabedaten
- Manuel zu erstellen
- u. U. Fehlerhaftes/Unvollständiges Modell
- Lange Simulationszeiten

Synthese als Ausweg?

- Def: **Synthese** ist das Zusammensetzen von Komponenten oder Teilen einer niedrigen (Modell-) Ebene zu einem Ganzen mit dem Ziel, ein auf einer höheren Ebene beschriebenes Verhalten zu erzielen.
  (☞ [Advanced-Learners]).

Problem:

- Synthese selten voll automatische
- Nur in wenigen Fällen ist die Korrektheit bewiesen worden.

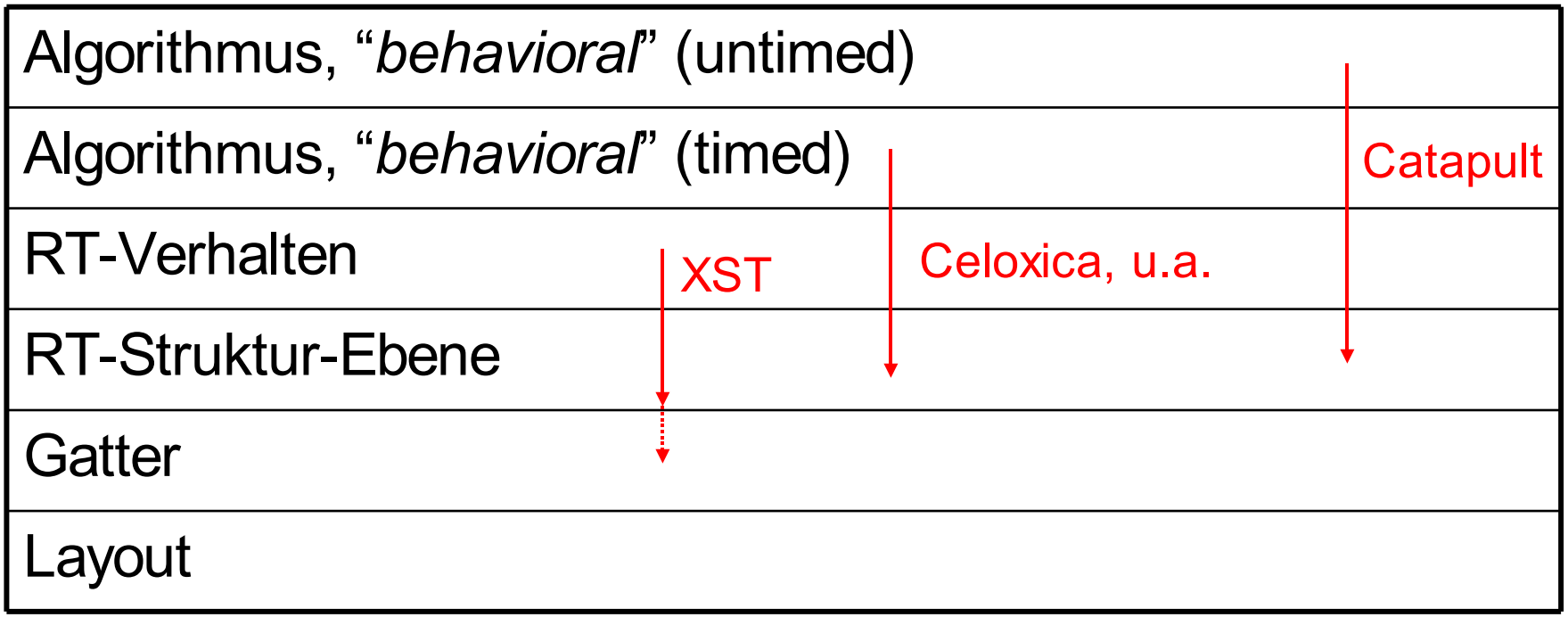
→ Simulation weiterhin notwendig!

Mögliche Entwurfsebenen

Ebene	Verhalten	Struktur	Geometrie
Vollständige Systeme	Verhalten des Gesamtsystems	Komponenten des Gesamtsystems	Geometrie des Gesamtsystems
…..			
Algorithmische Ebene	Entsprechend Berechnungsmodell	z.B. Knoten eines Taskgraphen	Abb. von Berechungen auf geometr. Inform.
PMS-Ebene	Gesamtverhalten eines Multiprozessormodells	*Processor, Memory, Switch*	Geometrische Inform. zu PMS-Komponenten
Instruction-Set Architecture	Befehlssemantik	Arithmetische & Transport-Operationen	Zuordnung zur Fest-/Fließkommaeinheit
Register-Transfer-Ebene	Register-Transfers	Register, RAMs, ALUs	Layout von RT-Bausteinen
Logik-Ebene	Boolesche Gleichungen	Gatter, Flip-Flops	Geometrieinformation zu Gattern & Flip-Flops
Schaltkreisebene	Netzwerkgleichungen	Transistoren	Schaltkreis-Layout
Bauelementebene	Gleichungen f. Gatter	Gates, Kanäle	Bauelemetlayout
Prozessebene	Diffusions-Verhalten	Kristallgitter	Masken

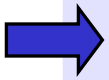
Formen der Synthese

Unterschiedliche Ziele und Quellen;
mögliche Darstellung:



Formen der Spezifikation

- Imperative Beschreibung (SystemC, C, …)
- …
- Funktionales Verhalten (im Sinne der Mathematik)



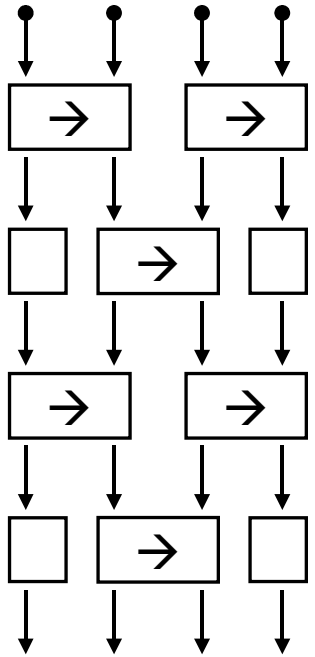
☞ Synthese von *systolischen* Feldern
aus funktionalen Spezifikationen

Systolisches Feld

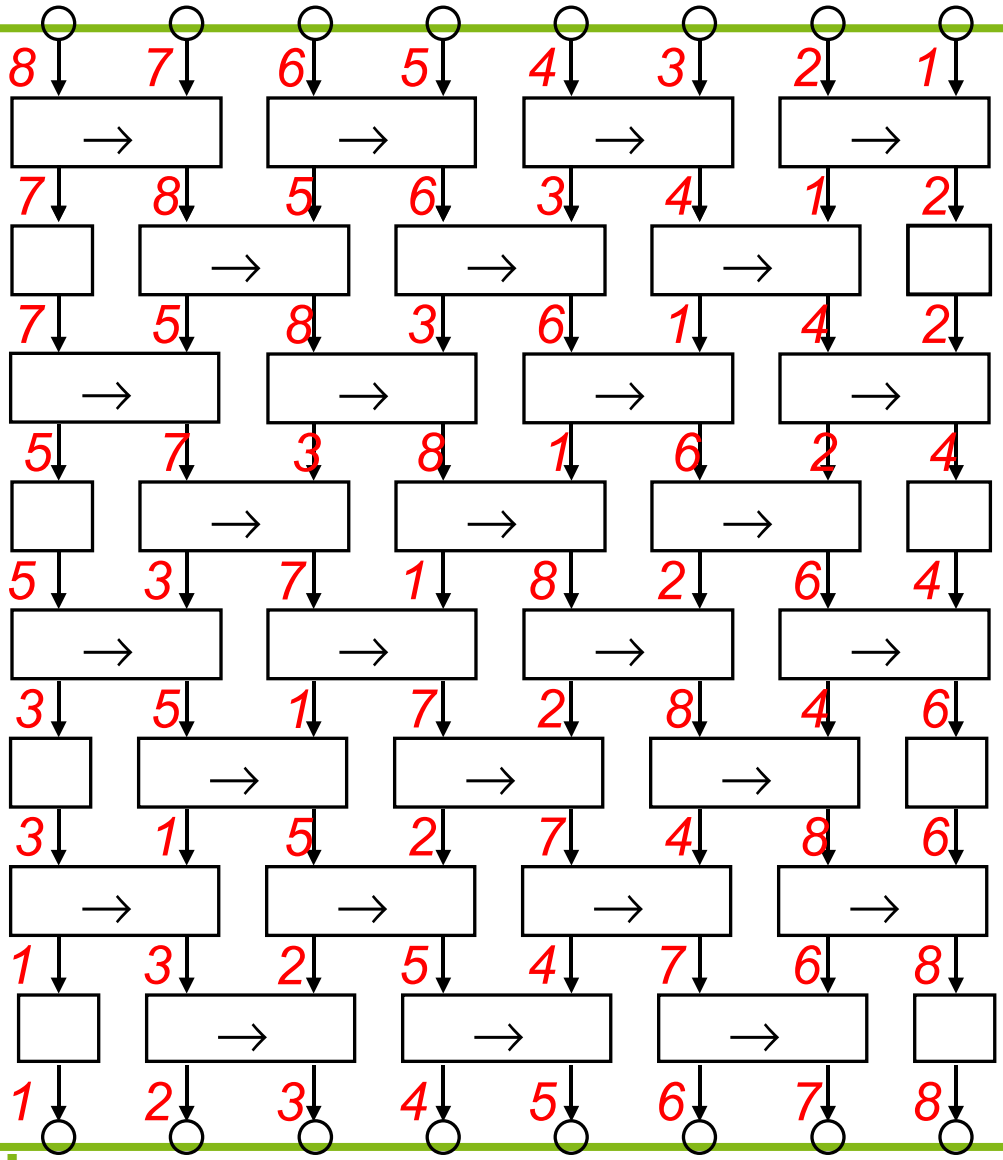
A systolic array is a special-purpose parallel device, made out of a few simple cell types which are regularly and locally connected. [☞ H.T. Kung]

Beispiel:

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



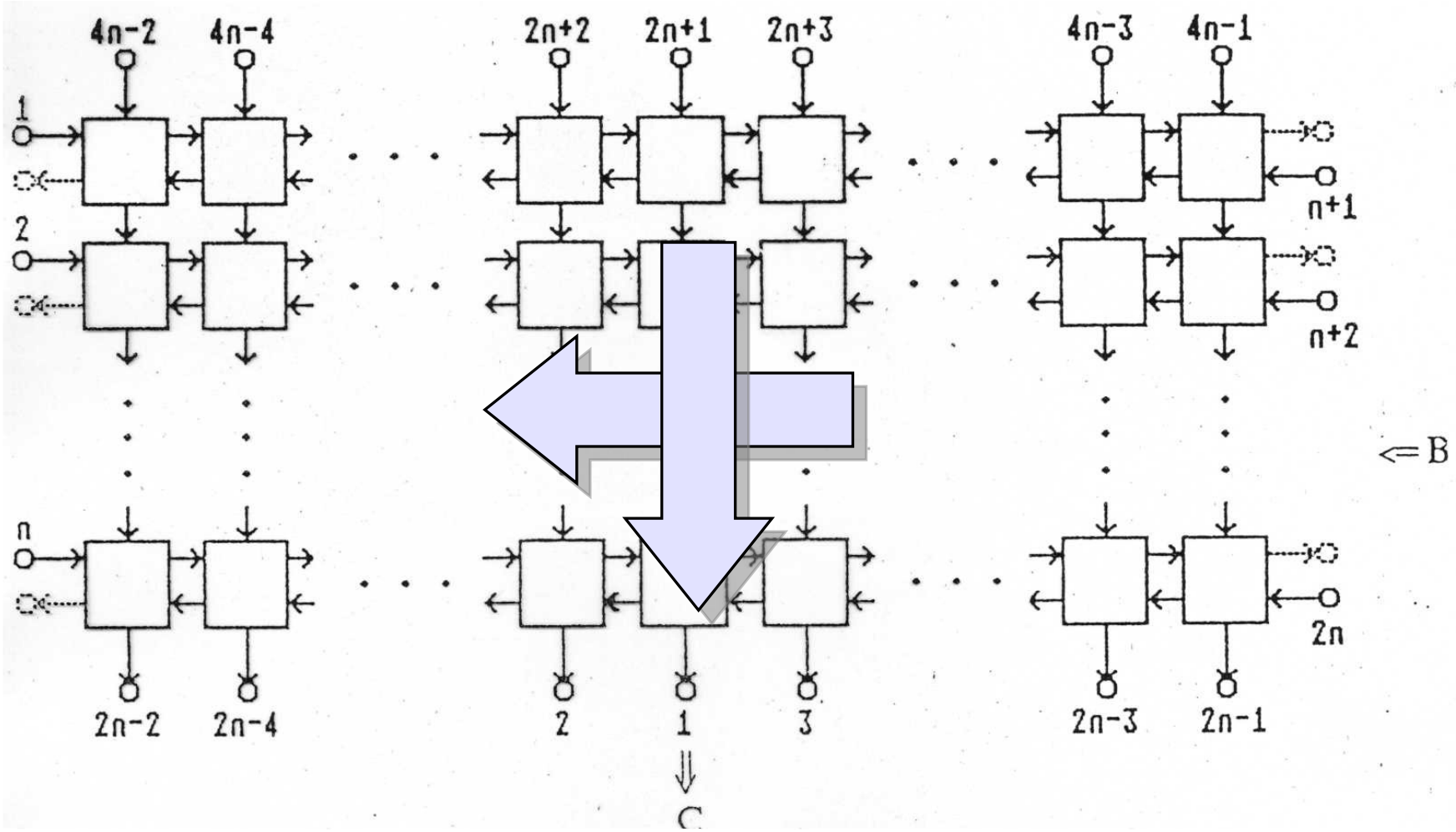
Beispiel für ein systolisches Feld: *odd-even transposition sort (OETS), n=8*



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

Prozessoren führen feste Befehle aus.

Behauptung: systolisches Array für Matrixmultiplikation



Wie kann man systolische Arrays systematisch konstruieren?

Ansatz zur Konstruktion: Vorgabe von Rekursionsformeln

Beispiele:

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

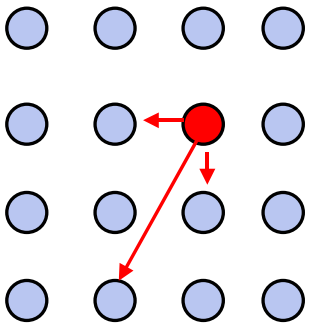
2.
$$He_{n+1}(x) = x * He_n(x) - n * He_{n-1}(x)$$

☞ 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

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



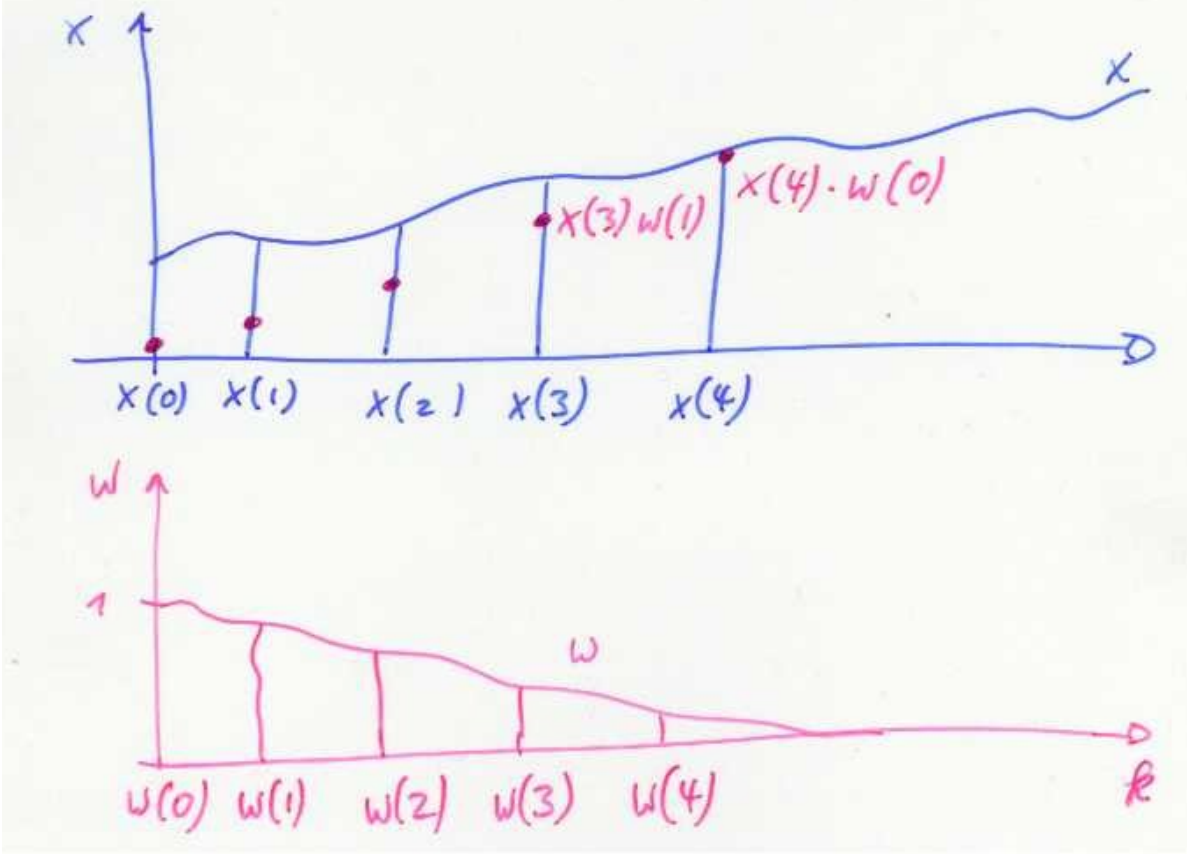
Raum D von Berechnungen

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

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

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

$\forall\, i \ge 0\ \forall\, k : W(i,k) = W(i-1,k)$

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

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

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

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

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

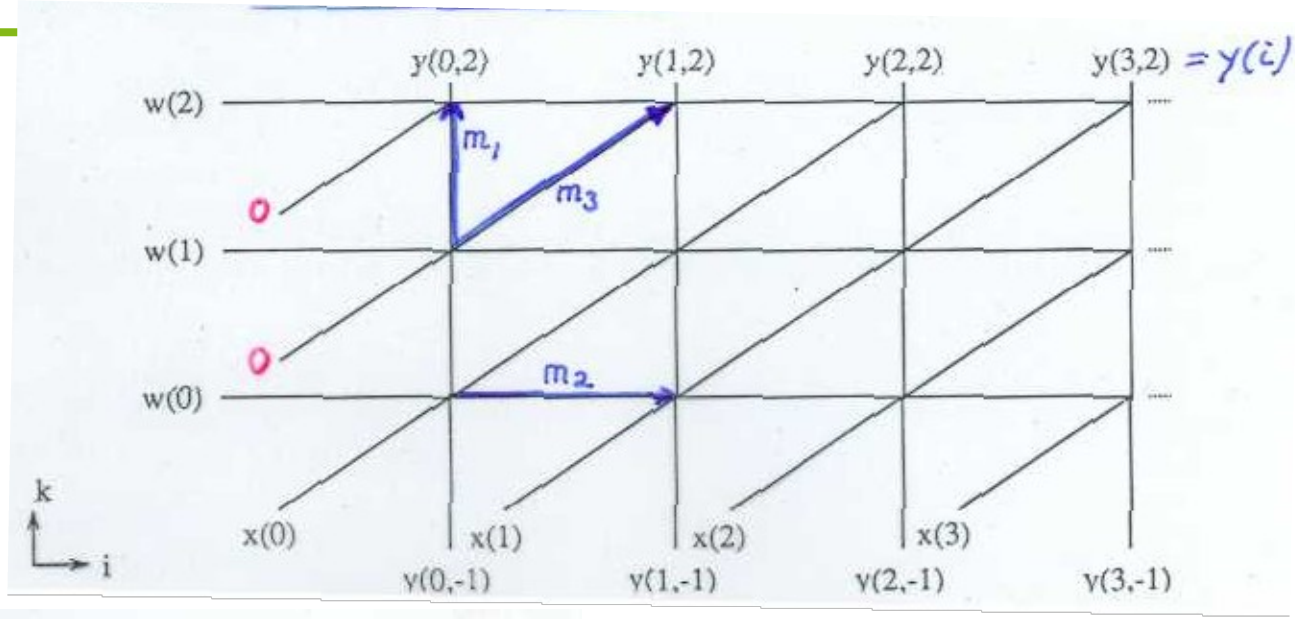
$\forall i \geq 0$: $X(i\text{-}1,\text{-}1) = x(i)$

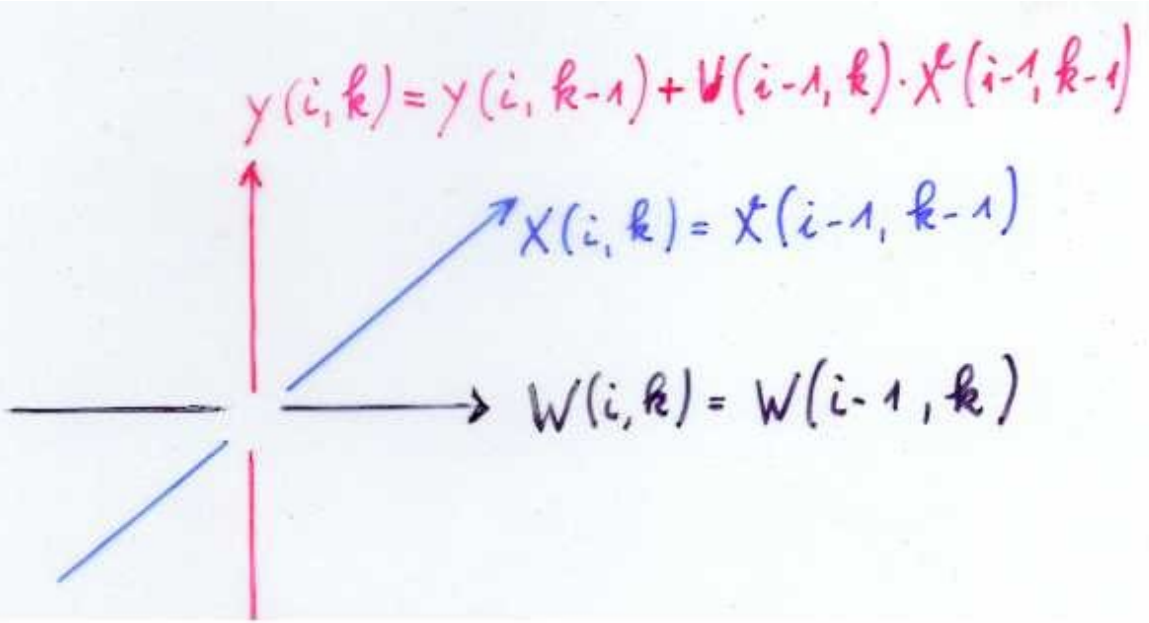
$\forall k,\ 0 \leq k \leq K$: $W(\text{-}1,k) = w(k)$

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

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.

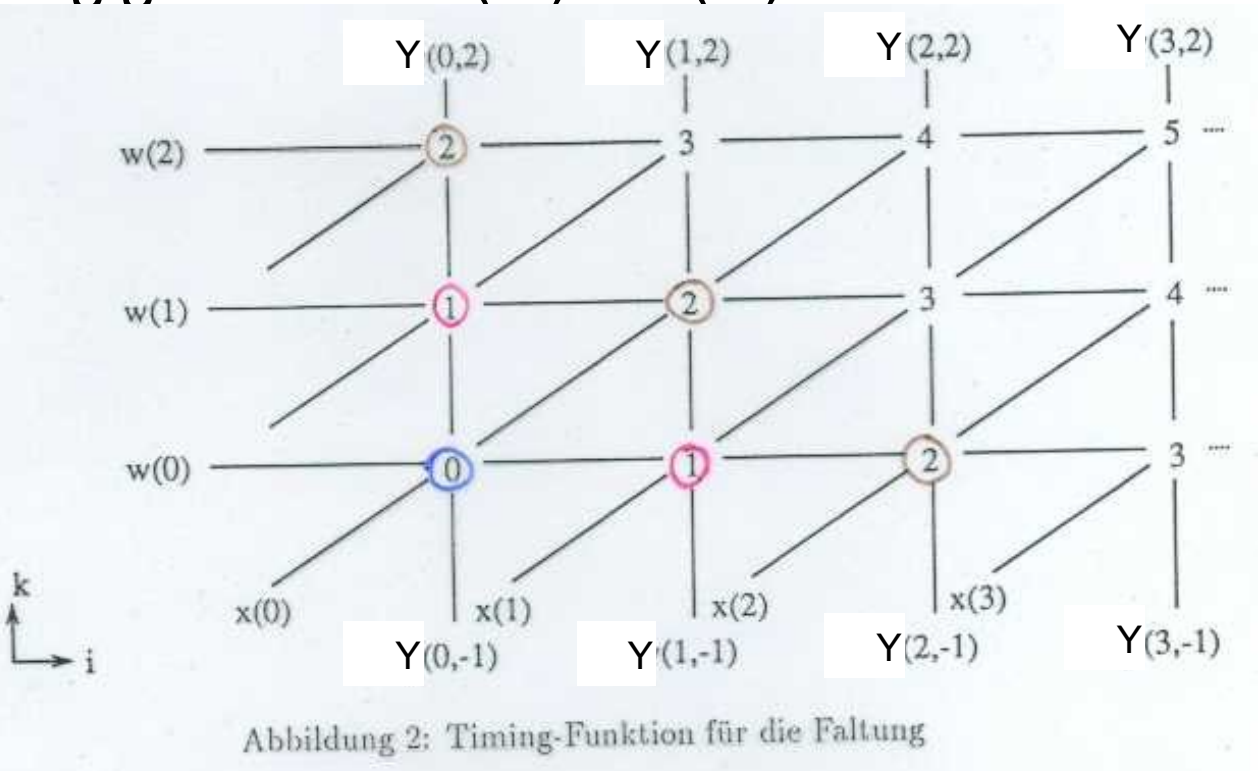
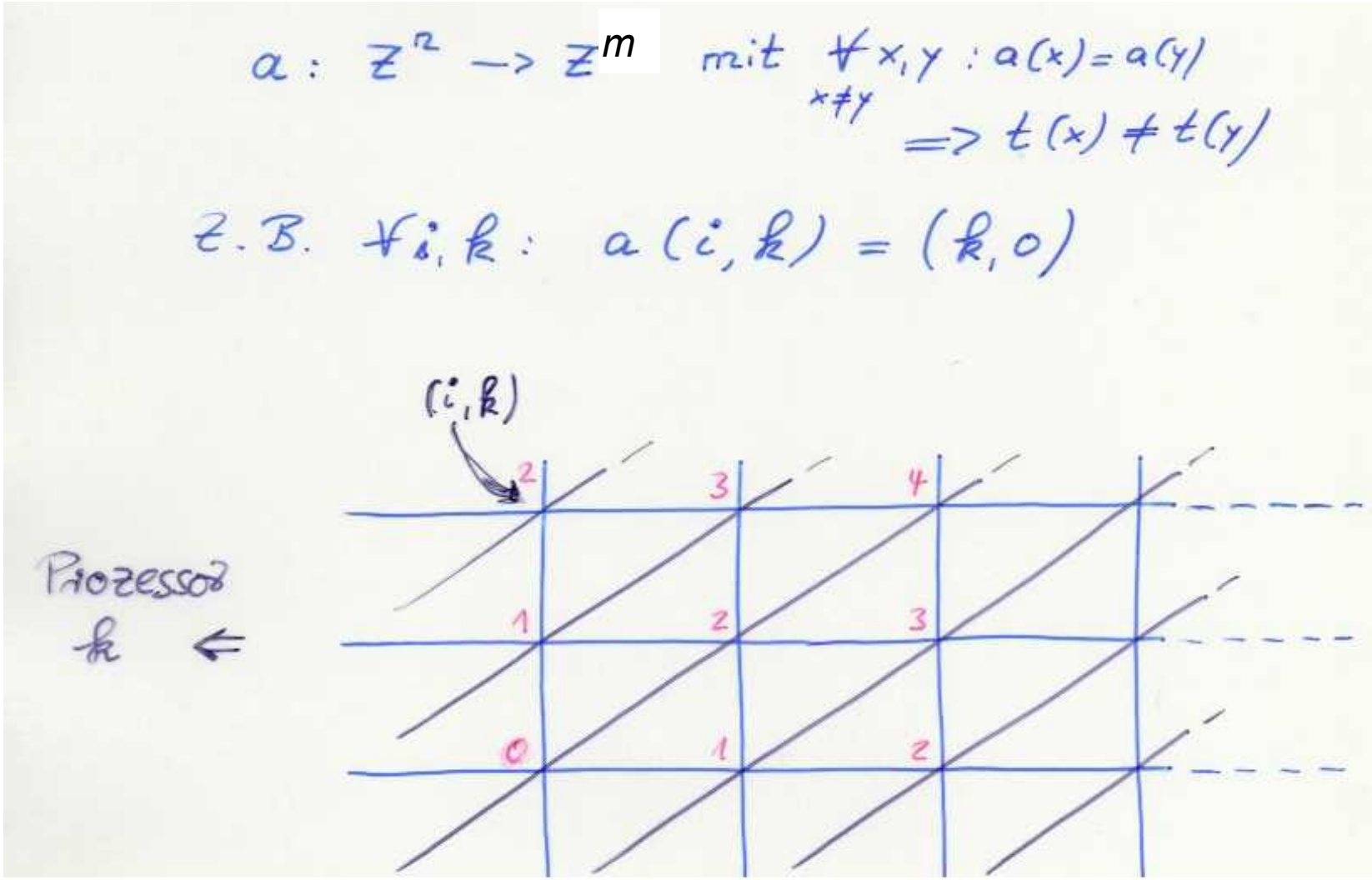


Abbildung 2: Timing-Funktion für die Faltung

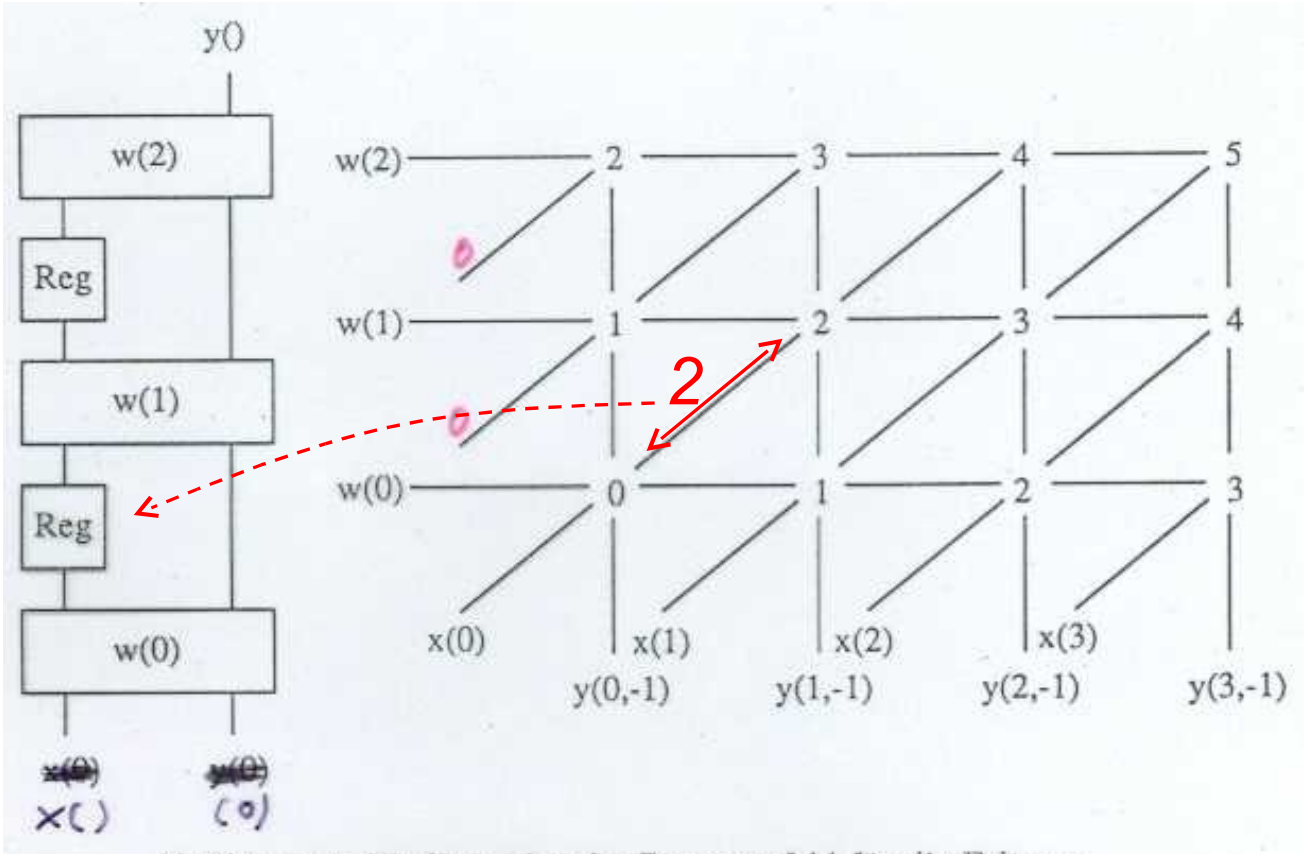
Suche nach einer Prozessorzuordnung



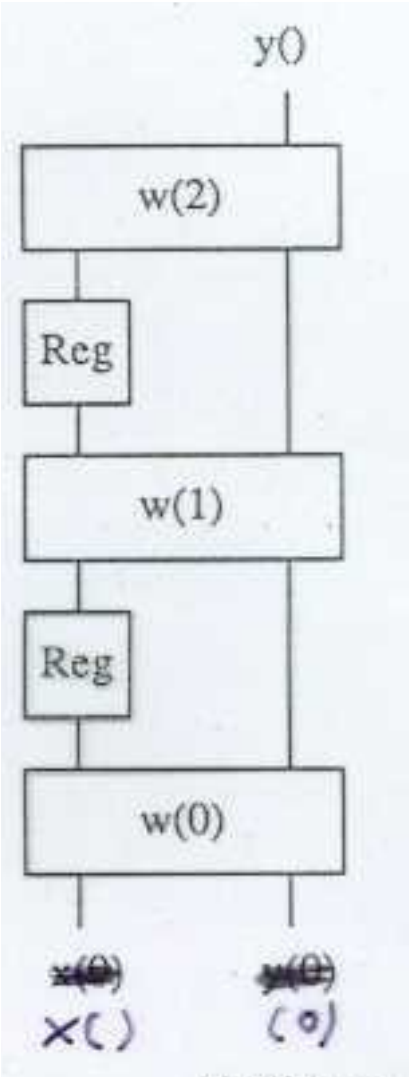
Bestimmung von Verbindungen und Registern

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.

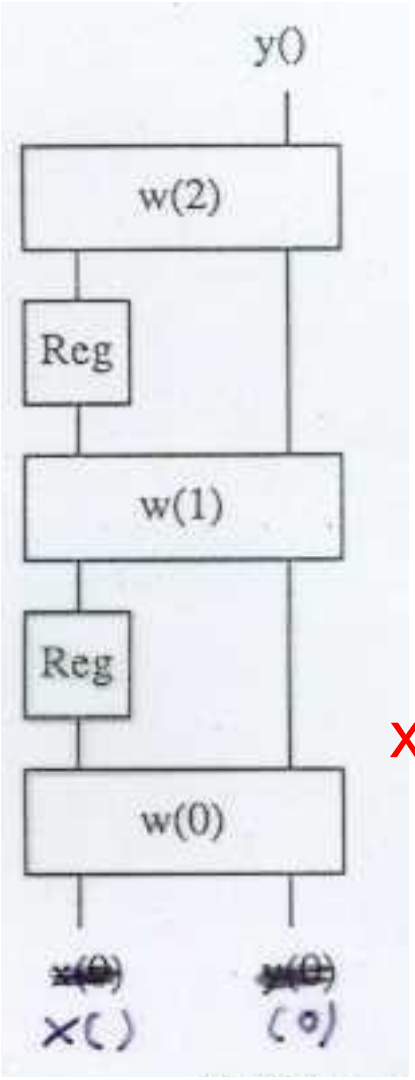


Ablauf (1)



x(0)

Ablauf (2)

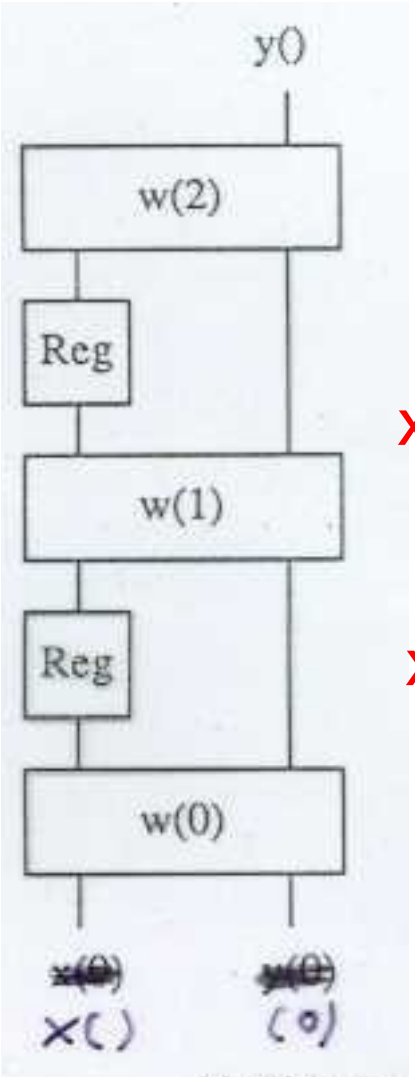


x(0)

x(1)

x(0)*w(0)

Ablauf (3)



x(0)

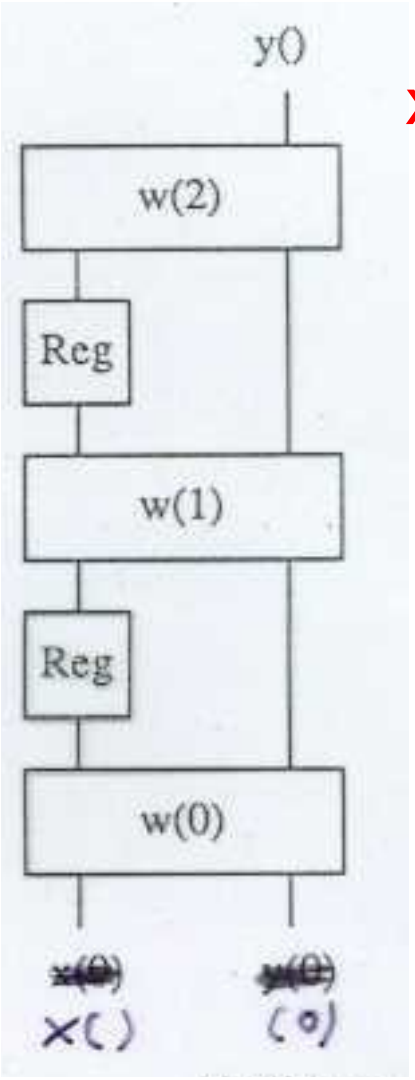
x(1)

x(2)

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

x(1)*w(0)

Ablauf (4)



x(0)

x(1)

x(2)

x(3)

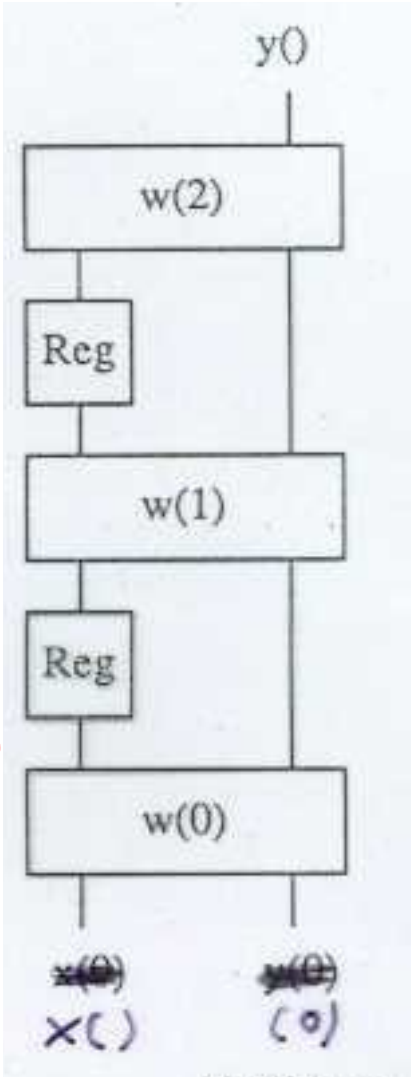
x(0)*w(0)+0*w(1)+0*w(2)

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)

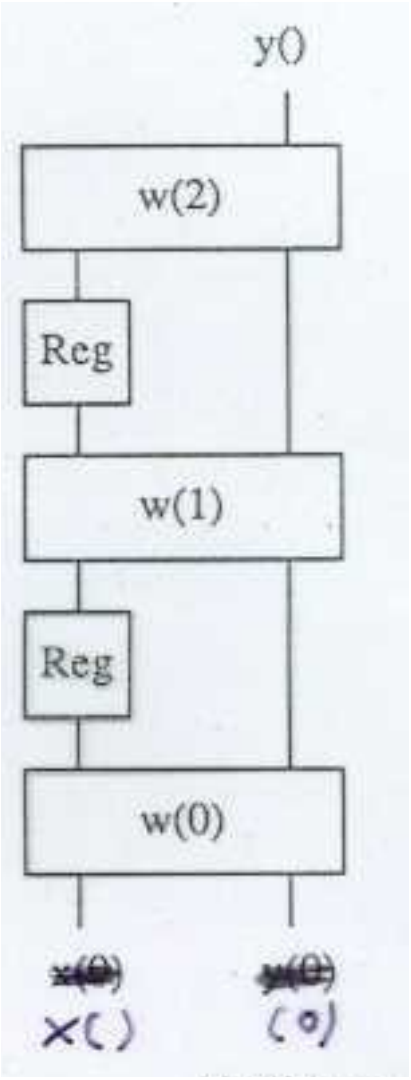
x(0)*w(0)+0*w(1)+0*w(2)

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)



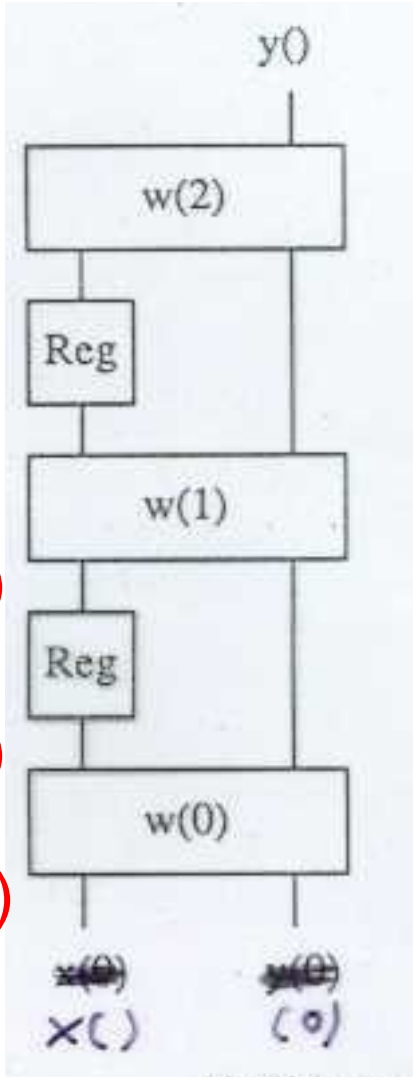
Ablauf (7)

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

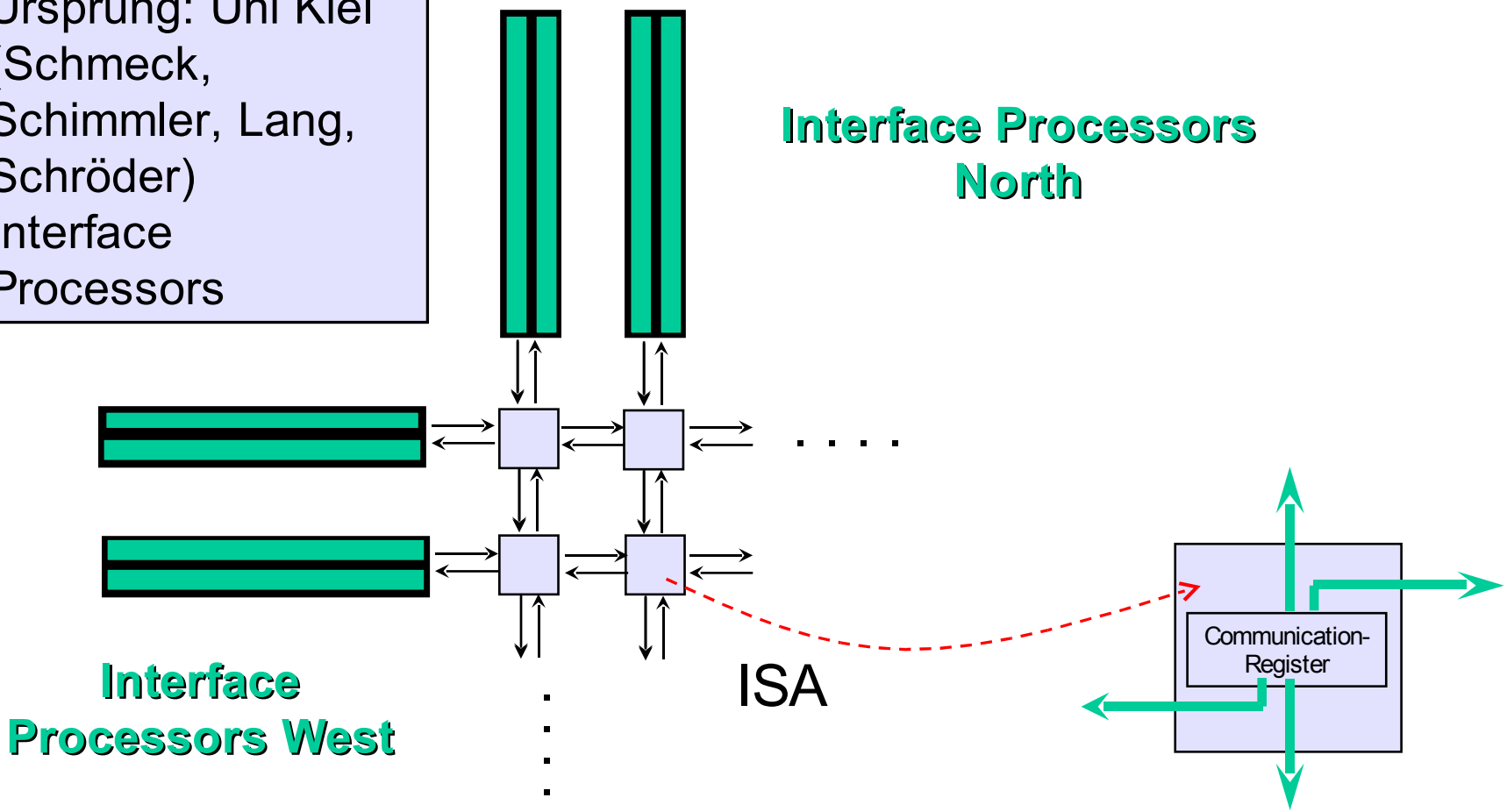
- Kung (ex-CMU), „*you just pump the data through*“; Gilt als „Vater“ der systolischen Arrays.
  Hat früher vorhandenen Ansätzen den Namen gegeben.

- Monica Lam

- Moldovan
- …

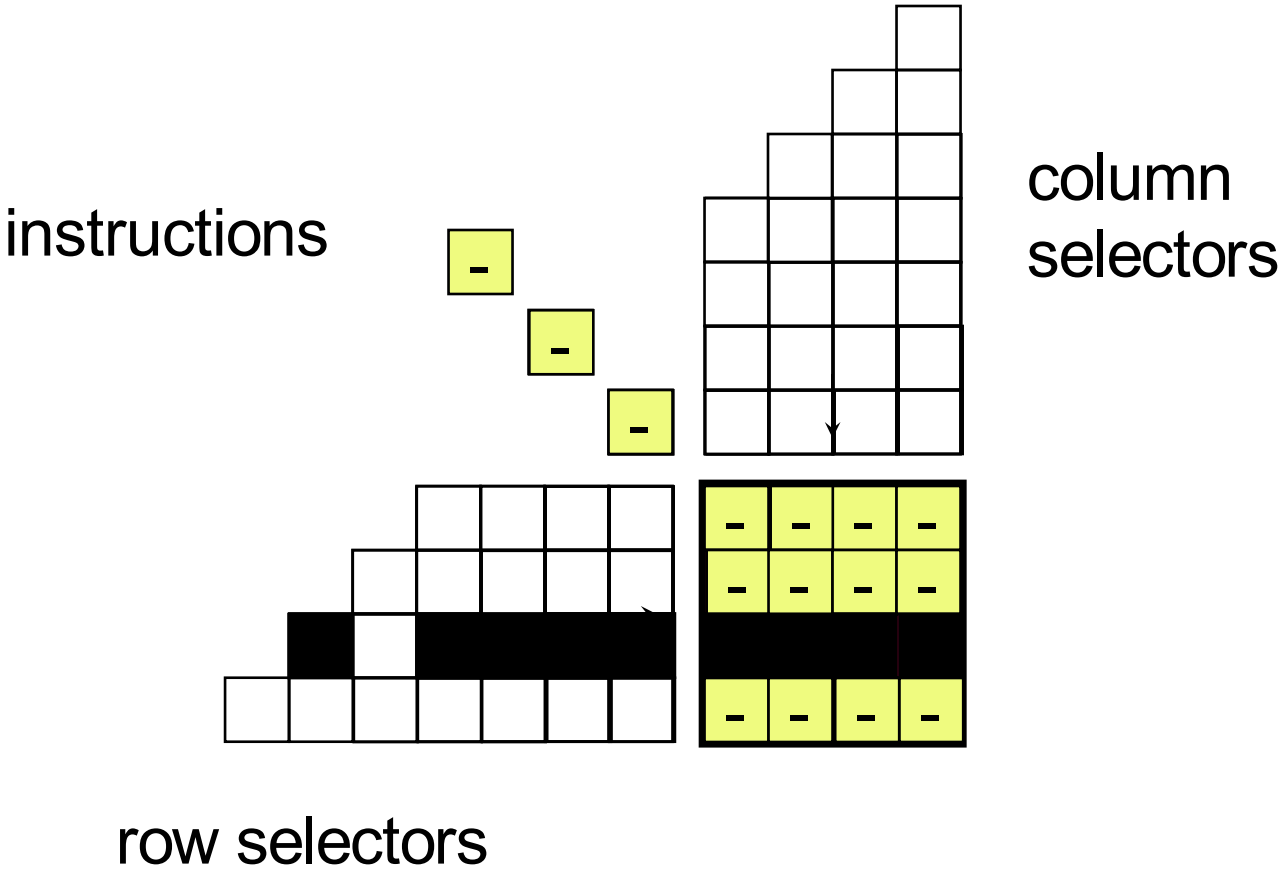
Instruction systolic array (ISA)

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



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

Instruction Systolic Array

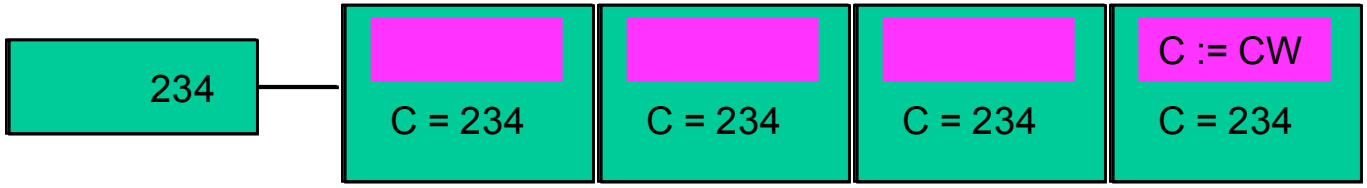


wavefront instruction execution

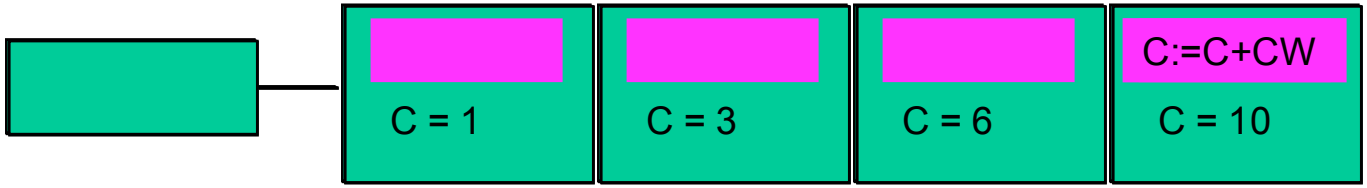
⇒ fast accumulation operations (e.g. row sum, broadcast, ringshift)

Advantage of ISA's: Performing Aggregate Functions

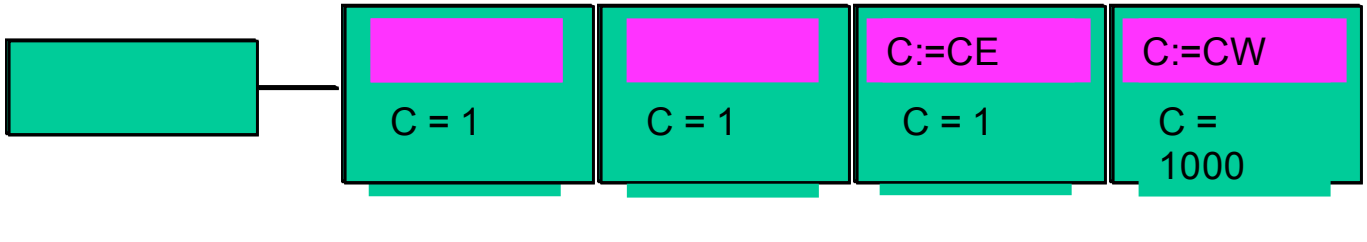
- Row Broadcast **C := C[WEST]**



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



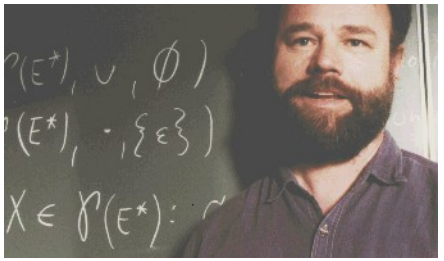
- Row Ringshift **C := C[WEST]; C:=C[EAST]**



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 u.a.:

- Systematische Implementierung von Walsh-Hadamard Transformationen
  //www.ll.mit.edu/HPEC/agendas/proc02/presentations/HPEC%20Day%201/Session%202/2.4fang-new.**ppt**
- Viterbi Decoder mit systolischen Algorithmus in FPGA realisiert:
  //www.ee.nthu.edu.tw/~jtchen/com5128/handout/ho4.**ppt** (?)
- (Viele weitere) …

Zusammenfassung

- Begriff der Synthese, Nutzung zur (sicheren) Erzeugung von Implementierungen
- Prinzip von systolischen Arrays
- Projektsmethode von Quinton zur systematischen Erzeugung von systolischen Arrays
- Schieben von Befehlen: *instruction systolic array* (ISA)
- Realisierung von systolischen Arrays mit FPGAs