Oberseminar “Vektorrechner und danach? . . . Streamprocessing. . . ”

Eine neue Computerarchitektur

Bernd Ulmann

09-JUN-2002

Gliederung

❶ Motivation

❷ Einführung

❸ Ein hypothetischer Streamprozessor, die STREAM/1

❹ Hochsprachenunterstützung

❺ Zusammenfassung und Ausblick

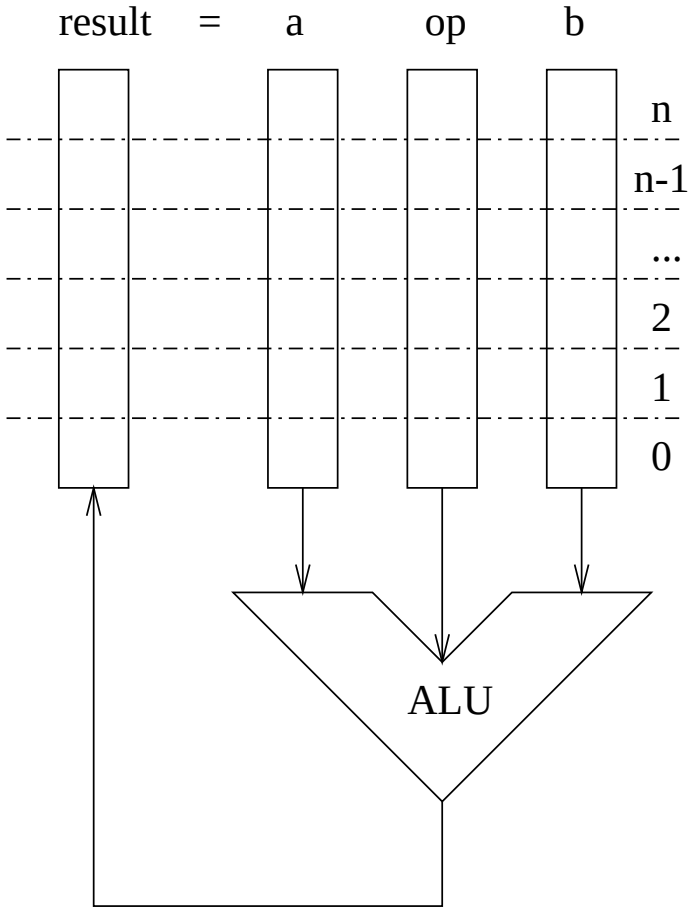
Motivation

- Spaß an komplexen Hard- und Softwaresystemen.
- Neues Architekturmodell, das die Nachteile herkömmlicher Vektorrechner zumindest teilweise vermeidet.
- Maschine mit extrem hoher Bandbreite für Zugriff auf Registerbänke.
- Nichtklassische Maschinenarchitektur.
- Möglichkeit, rekonfigurierbare Hardware quasi „organisch“ einsetzen zu können.

Einführung

- Vektorrechner wenden in einer Pipeline eine Instruktion sukzessive auf alle Elemente der beteiligten Datenvektoren an.
- Ein Streamprozessor hingegen wendet die Elemente eines Operationsvektors auf alle Elemente der beteiligten Datenvektoren an.

Prinzip des Streamprocessing



Beispiel 1

```
      sum = 0
      do 10 i = 1, 1000
        if (a(i) .gt. 0) sum = sum + a(i)
10    continue
```

- Ein Vektorrechner führt diese Summation mit Hilfe eines Maskenregisters aus.
- Ein Streamprozessor erzeugt einen passenden Operationstream, in dem nur an den gewünschten Stellen eine Addition codiert ist.

Beispiel 1

❶ Schritt (in Pseudo-FORTRAN-77):

```
          do 10 i = 1, 1000
10          op(i) = a(i) .gt. 0 ? + : id
```

❷ Schritt:

```
          sum = 0
          do 20 i = 1, 1000
20          sum = sum .op(i) a(i)
```

Bemerkungen

- Vektoren aus Operatoren (*operation streams*) können Ergebnis vorhergehender Berechnungen sein.
- Selbstgenerierender Code.
- Es müssen Mechanismen für die Abbildung von Ergebniswerten von Operationen auf Operationscodes vorgesehen werden (*instruction lookup tables*).
- Es sind hochuniverselle Adressgeneratoren notwendig, um möglichst viele Schleifenformen direkt abbilden zu können.

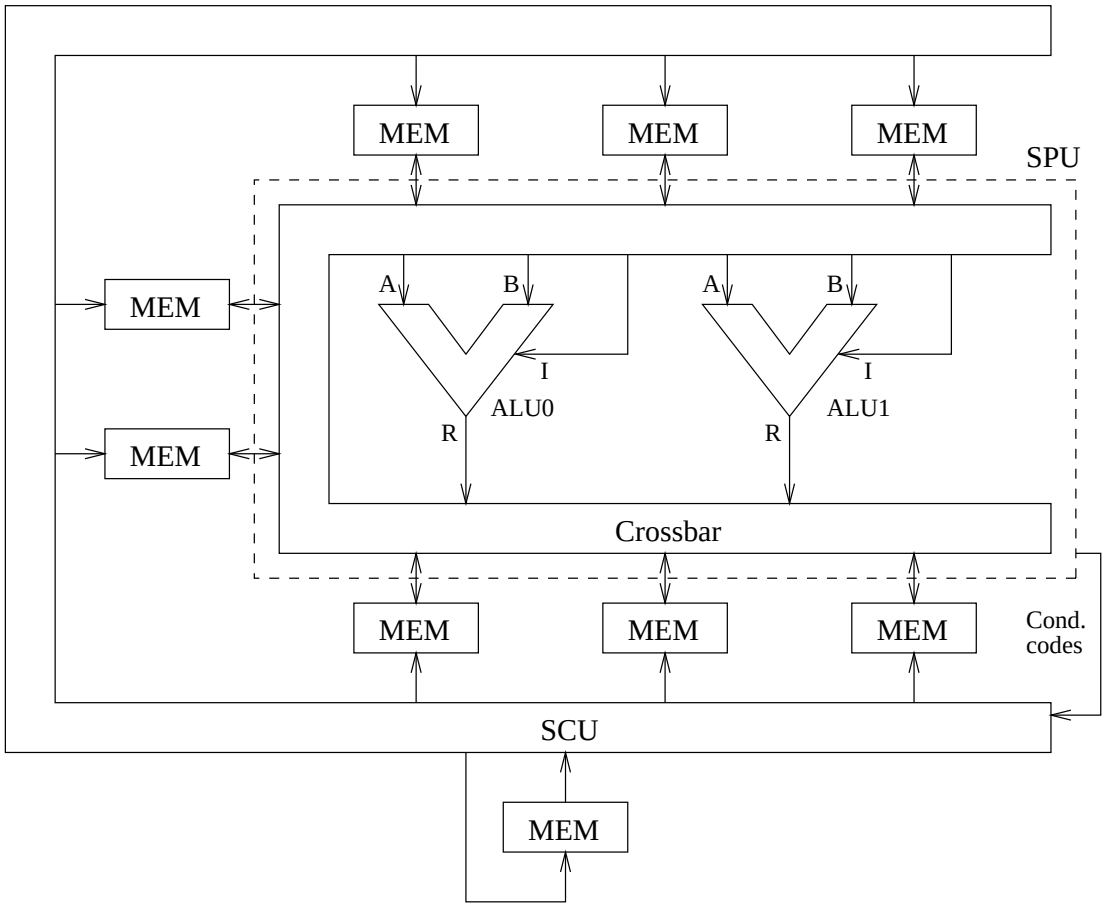
Die STREAM/1

- Hypothetischer Streamprozessor – vorerst nur als Simulation existent.
- Implementation als Coprozessor auf FPGA-Basis geplant.
- Stark eingeschränkter Funktionsumfang, dient nur als Machbarkeitsstudie.

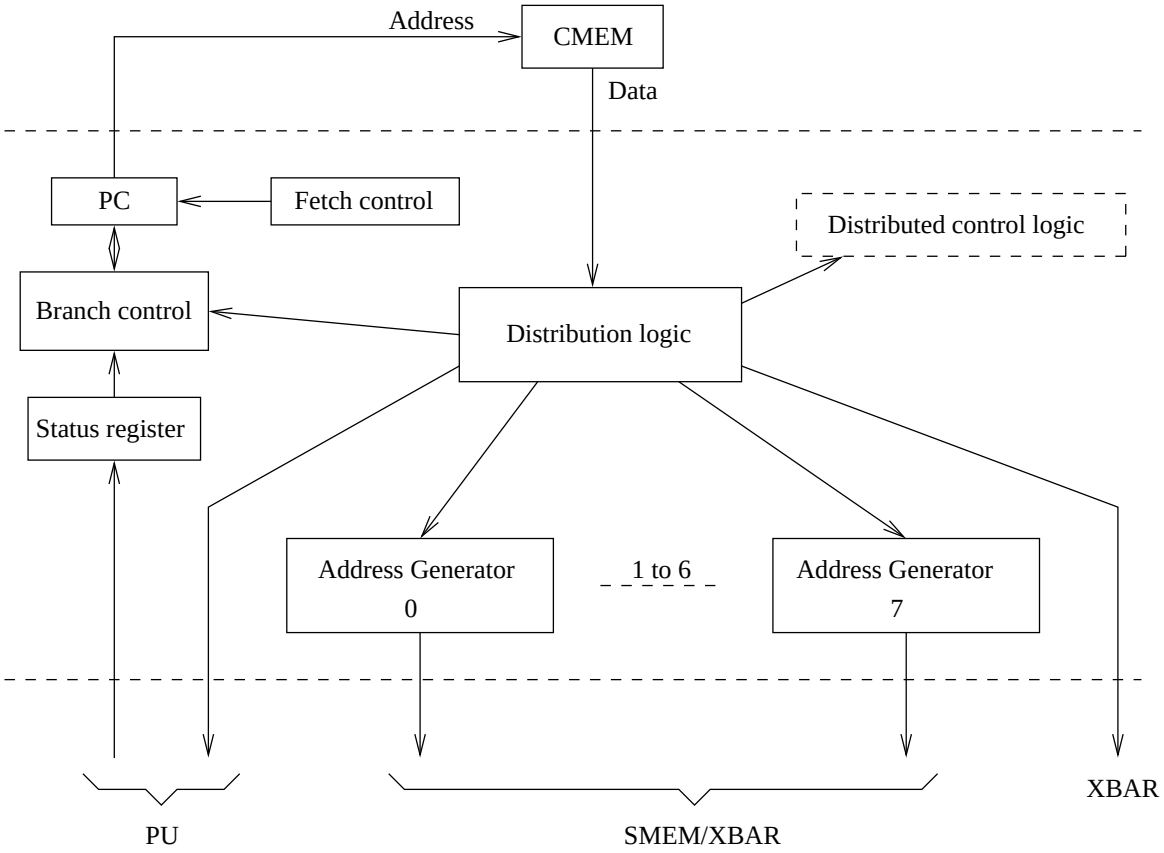
Grundelemente der STREAM/1

- Mainmemory: Enthält *main-instructions*.
- Stream-Control-Unit (*SCU*): Enthält Adressgeneratoren, Statusregister, etc.
- Streammemory: Vergleichbar herkömmlichen Vektorregistern, enthält Daten beziehungsweise Operatoren. 8 Bänke verfügbar.
- Kreuzschienenverteiler (*crossbar*): Verknüpft einzelne Streammemorybänke mit den Ein-/Ausgängen der Berechner.
- Stream-Processing-Unit (*SPU*): Enthält (rekonfigurierbare) ALU(s).

Die Struktur der STREAM/1



Die SCU


Address
CMEM
Data
PC
Fetch control
Distributed control logic
Branch control
Distribution logic
Status register
Address Generator
0
1 to 6
Address Generator
7
XBAR
PU
SMEM/XBAR

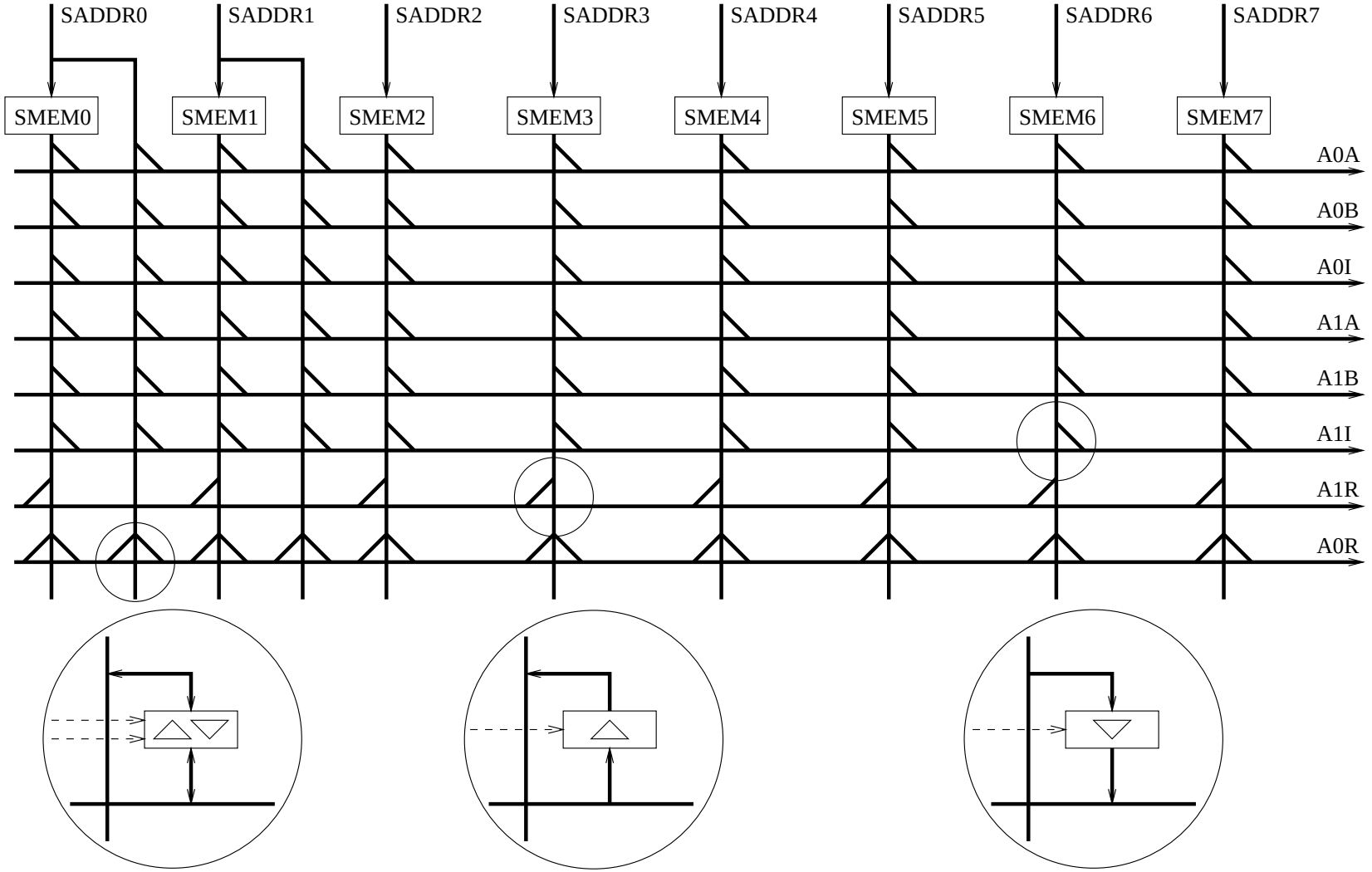
Die Adressgeneratoren

- Komplexester Teil der SCU.
- Deutlich flexibler als herkömmliche Adressgeneratoren von Vektorrechnern.
- Erzeugt Adresssequenzen folgender Form:

$$a_i := \lfloor \frac{c + o_i}{d_i} \rfloor stride_i + s_i \text{ mod } m_i$$

i: Nummer des Generators	a: Ausgabeadresse	c: Zyklusnummer
o: Startoffset	d: Vorteiler (Takt)	$stride$: Schrittweite
s: Startwert	m: Modulus	

Der Kreuzschienenverteiler



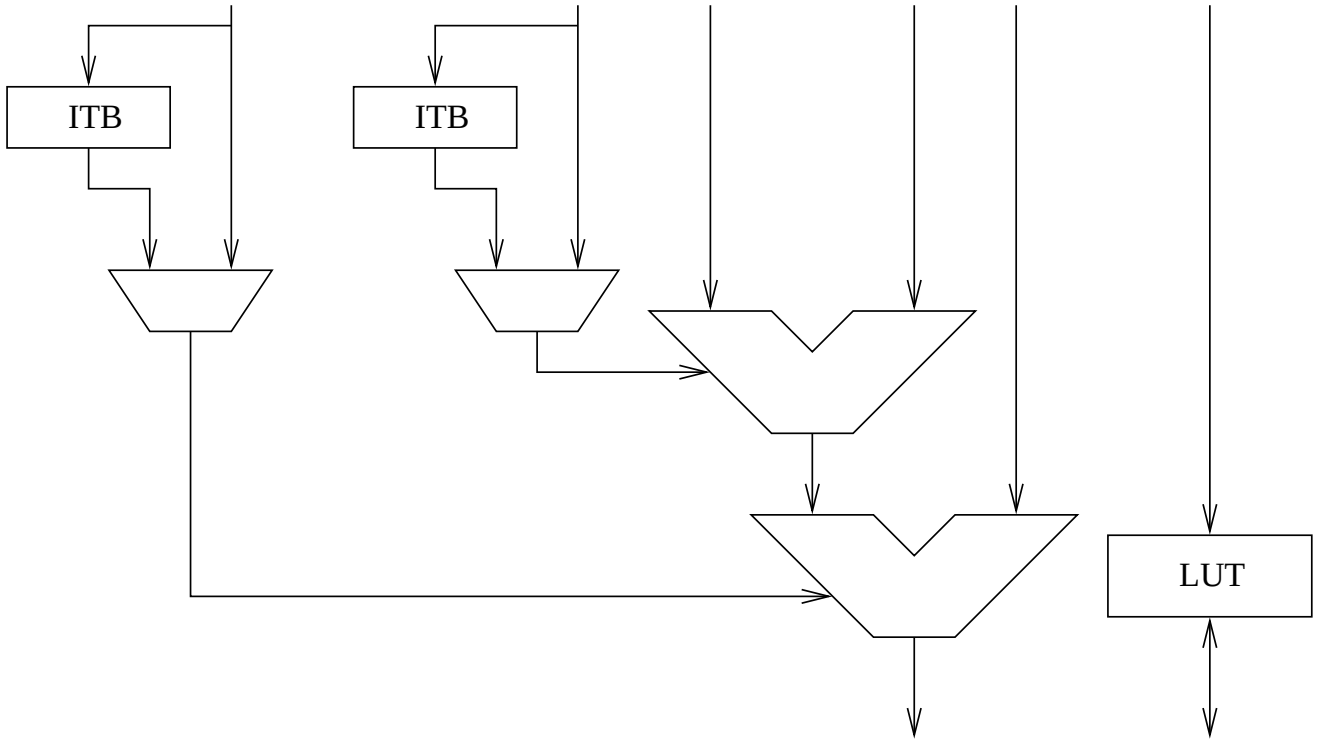
Bemerkungen zum Kreuzschienenverteiler

- 8-aus-10 Kreuzschienenverteiler.
- Input aus den 8 Streammemorybänken sowie direkt zwei der Adressgeneratoren (beispielswiese zum Erzeugen arithmetischer Progressionen oder zur Initialisierung von Streams).
- Durch Verzicht auf Butterfly-Netzwerk können auch mehrere Streammemorybänke beispielsweise mit denselben Werten (aber durchaus mit anderer Adressierung) befüllt werden, etc.

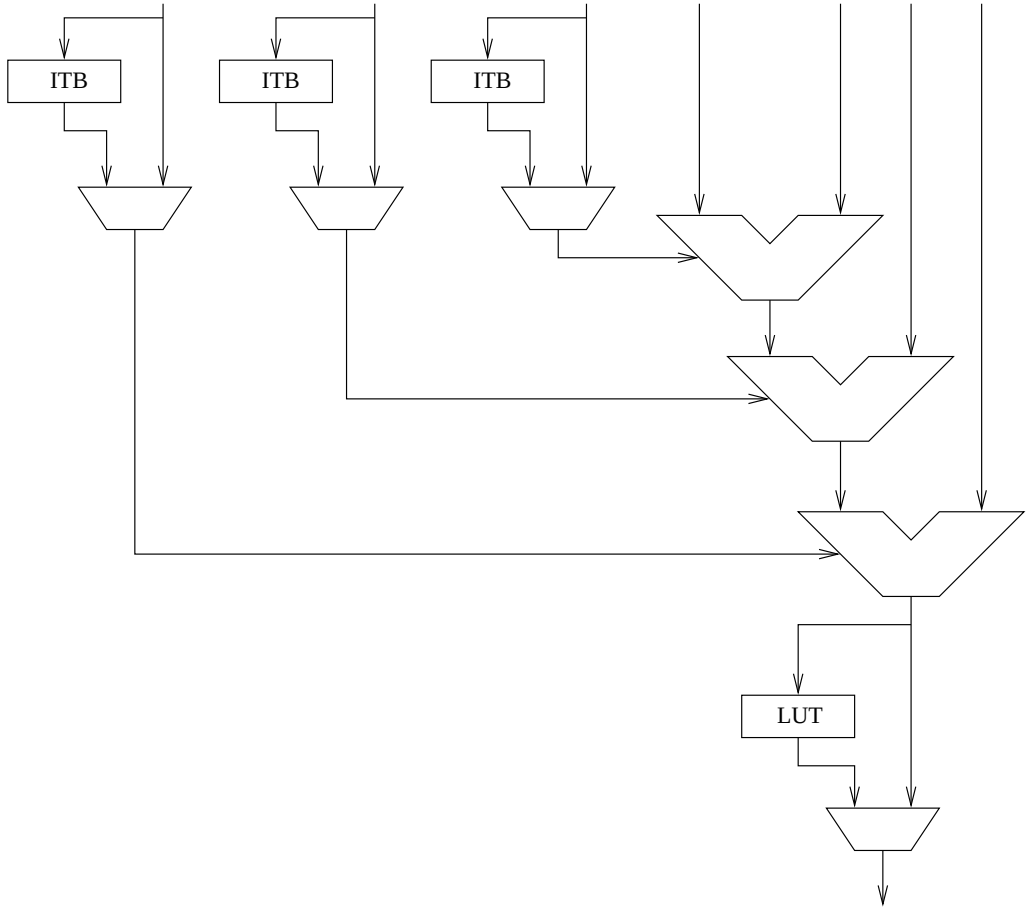
Die SPU

- Herzstück des Streamprozessors.
- Rekonfigurierbar – es stehen sechs fertige Strukturen zur Verfügung, die direkt verwendet werden können, zwei weitere Strukturen sind benutzerdefinierbar.
- Es steht eine allgemein verwendbare Lookup-Tabelle zur Verfügung, um beispielsweise Zellregeln für Zellularautomaten zu implementieren, etc.
- Jeder ALU kann eine Instruktionsumsetztabelle (*ITB*) vorangeschaltet werden, um Datenströme effizient als Operatorströme uminterpretieren zu können.

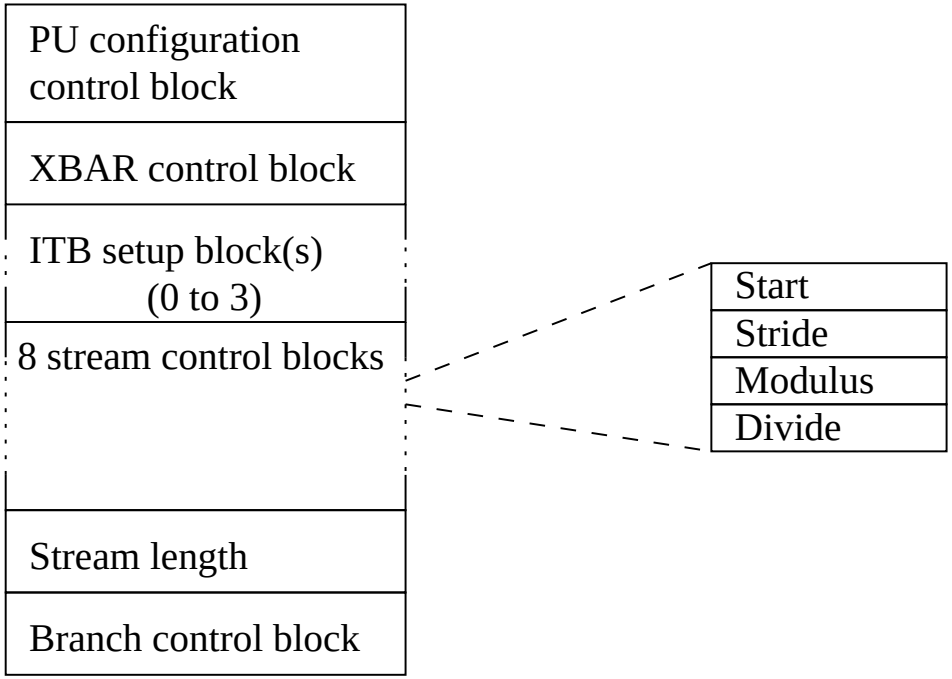
ALU-Konfiguration 1



ALU-Konfiguration 2



Main instruction format



Hochsprachenunterstützung

- Für SISD-Maschinen konzipierte Sprachen, wie beispielsweise `FORTRAN-77` müssen um zusätzliche Konstrukte erweitert werden, um Parallel- und Vektorrechner effizient programmieren zu können.
- Eine speziell für solche Spezialprozessoren ausgelegte Sprache kann wesentlich bessere Leistungen erbringen als eine allgemeingültige Sprache.
- Historisch: `APL` und die Control-Data `STAR-100` – Sprach- und Maschinenkonzeption gingen Hand in Hand.

APL

- Vorteile:
  - Hohes Maß an Abstraktion.
  - Kurzer, prägnanter Code.
  - Direkte Abbildung komplexer Datenstrukturen möglich.
  - Weitgehender Verzicht auf Schleifenkonstruktionen.
  - Dadurch ideal für Spezialprozessoren wie Vektor- und Parallelrechner geeignet.
- Nachteile:
  - Gewöhnungsbedürftiger Zeichensatz.
  - Teilweise „zu mathematische“ Herangehensweise an Probleme.

TSL

- TSL – Tiny-Stream-Language.
- Verwandt mit APL (beziehungsweise J) und BLISS.
- Einziger Datentyp ist das Maschinenwort.
- Bei Feldelementen wird nicht zwischen Operatoren und numerischen Werten unterschieden.
- Direkte Unterstützung für feingradigen Parallelismus in der SPU.
- Keine eigenen I/O-Routinen, verläßt sich auf einen Front-End-Prozessor.

TSL-Beispiel 1

```
dcl m[100,100], n[100,100], c;
...
c = 0;
c +/ m[5, ] * n[5, ]
```

- `+/` entspricht dem APL-Reduktionsoperator, d.h. es wird in `c` aufsummiert.
- `m[5, ]` stellt ein Slice aus dem zweidimensionalen Feld `m` dar (resp. n).

TSL-Beispiel 2

```
dcl a[4], b[4], c[4], op[4];
...
a = (1, 2, 3, 4)
b = (4, 3, 2, 1)
op = (+, -, *, /)
...
c = a .op b;
```

Zusammenfassung und Ausblick

- Streamprozessing erweitert die Vektorrechnerarchitektur um den Begriff des Operationsvektors.
- Streamprozessoren zeichnen sich durch komplexe und universell einsetzbare Adressgeneratoren aus.
- Neue Sprachstrukturen zur effizienten Programmierung notwendig – eventuell positiver Effekt für die Programmierung von Vektorrechnern möglich.
- Streamprozessoren machen einfach Spaß – zumindest dem Vortragenden.