Draft: Übersicht über Hard- und Softwarearbeiten für die MoM3

1. Hardware für die MoM3

1.1. Anforderungen

Folgende Anforderungen an die Xputer-Hardware werden gestellt:

1. Mehrere Data Scan Caches stellen Daten von verschiedenen Stellen aus dem Arbeitsspeicher bereit (nicht-lokale Kommunikation).
2. Jeder Cache kann ein unabhängiges Scan Pattern abarbeiten, d.h. für jeden Cache wird ein eigener Adressgenerator benötigt.
3. Die Caches sind über parallele Leitungen mit den rALU-Subnetzen verbunden.
4. Datenaustausch zwischen den Caches findet über die rALU-Subnetze statt.
5. In die rALU-Subnetze können komplexe Verbundoperatoren konfiguriert werden, die ganze arithmetische Ausdrücke berechnen.
6. Bildverarbeitungsoperationen, die auf Pattern Matching basieren, benötigen in der rALU eine Art Assoziativspeicher zum Mustererkennen und ein RAM zur Selektion der Ergebnismuster, allerdings kommen diese Algorithmen i.a. mit einem Cache aus.
7. Die Parameter zur Steuerung der Data Scan Caches werden aus dem Datenspeicher gelesen (durch einen reservierten Cache, den sog. Escape Cache), die Auswahl der nächsten Parameter erfolgt durch eine Restkontrolle in der rALU.

1.2. Vorschlag zum Aufbau der IC's für die MoM3

Dies resultiert in folgendem Bedarf an ICs, die zu entwickeln und zu fertigen sind:

- Adressgenerator für einen zweidimensional organisierten Speicher (Bild 1). Wegen der Bitbreite der steuernden Parameter (ca. 200 Bit), werden die entsprechenden Speicherzellen des Escape Cache auf diesem Chip implementiert und wort-sequentiell aus dem Datenspeicher geladen, sobald die Restkontrolle aktiv wird. Pro Cache wird ein solcher IC benötigt.

Draft: Übersicht über Hard- und Softwarearbeiten für die MoM3

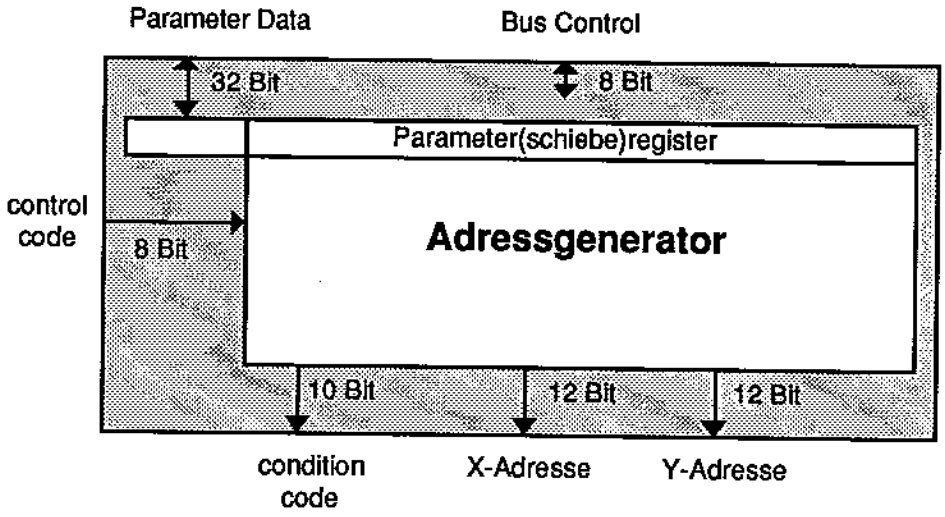

Bild 1: Der Adressgenerator IC

Gatterkomplexität:	ca. 10000 Gatter
Spezifikation:	1 Monat
Standardzellenentwurf*:	6 Monate
Simulation und Test*:	4 Monate
Fertigung*:	2 Monate
Testplatine*:	2 Monate

* Bei Vergabe als Diplomarbeit an Studenten

- Data Scan Cache und rALU müssen wegen der hochparallelen Datenpfade auf einem IC liegen, da sonst das Pinout-Problem zu starke Restriktionen aufwirft. Herkömmliche feldprogrammierbare Bausteine bieten keine geeignete Struktur an, um eine große Menge Master-Slave Flipflops (Cache), hochparallele Leitungsbündel und vielstufige Logik (rALU für Arithmetik) zu implementieren. Daher werden zwei rALU-Varianten angeboten, die den extrem unterschiedlichen Anforderungen logischer und arithmetischer Anwendungen gerecht werden:

  Ein Cache mit Assoziativspeicher (DPLA), Auswertelogik und Speicher für die Ergebnismuster für Pattern Matching (Bild 2).

Draft: Übersicht über Hard- und Softwarearbeiten für die MoM3

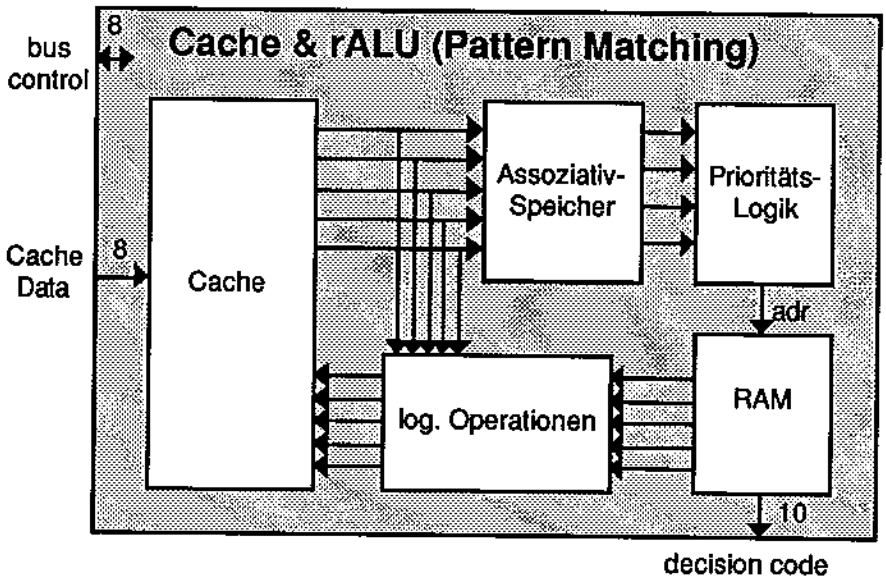

Bild 2: Prinzipieller Aufbau eines Caches mit rALU für Pattern Matching

Wegen des enormen Platzbedarfs des Assoziativspeichers wird hierfür auf eine vorhandene Platine zurückgegriffen, die mit selbstentwickelten DPLA-Chips bestückt ist. Nur der Cache, die Auswertelogik und das RAM zur Erzeugung der Ergebnismuster müssen auf einem IC zusammengefaßt werden (Bild 3).

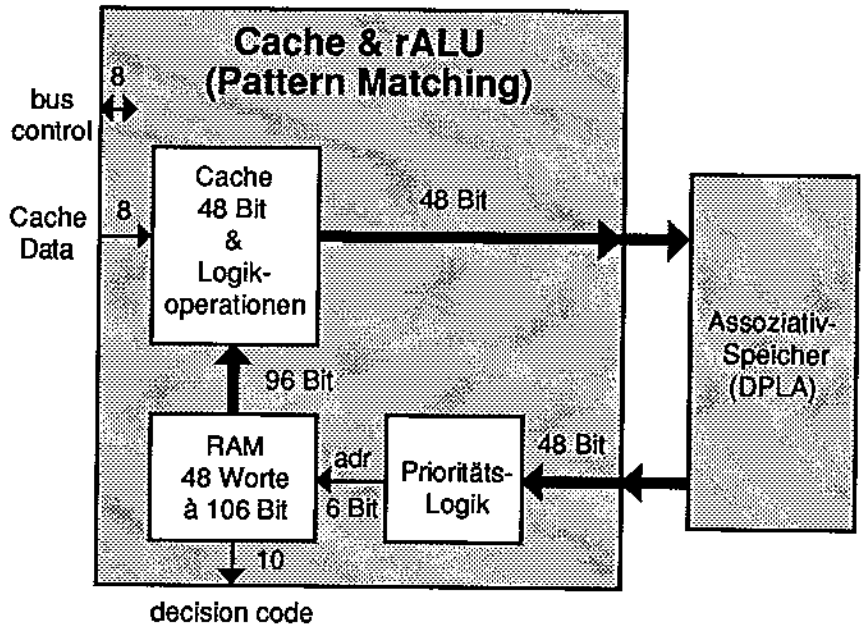

Bild 3: Cache und Auswertelogik für Pattern Matching als IC

Gatterkomplexität:	ca. 30000 Gatter
Spezifikation:	2 Monate
Standardzellenentwurf*:	4 Monate
Simulation und Test*:	4 Monate
Fertigung*:	2 Monate
Testplatine*:	2 Monate

Eine Cachezelle mit integrierter Logik für einfache Bitmanipulationen ist in Bild 4 dargestellt, zur besseren Verständlichkeit ist die Schaltung nicht minimiert. Aus solchen Zellen wäre der Cache im rALU-Chip nach Bild 3 aufgebaut.

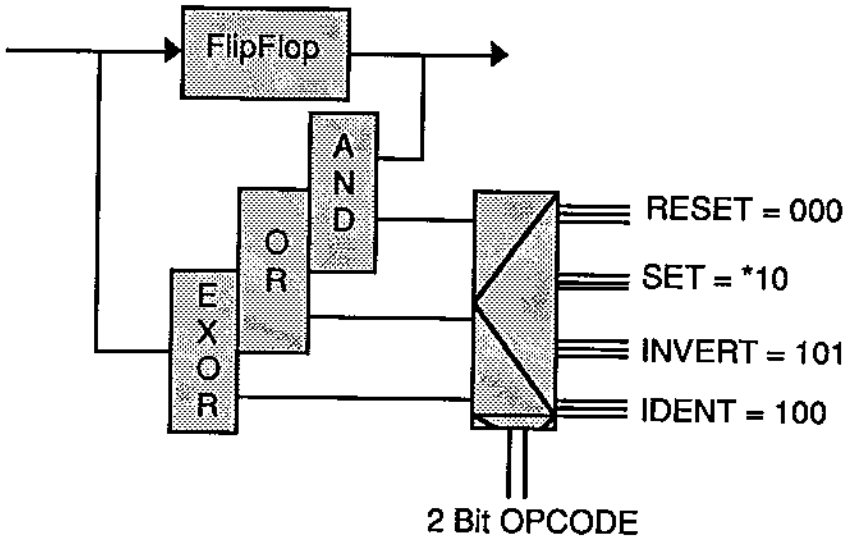

Bild 4: Prinzipieller Aufbau einer Cachezelle mit integrierter Logik

Ein weiterer IC ist für arithmetische Anwendungen gedacht, und vereinigt Cache, konventionelle ALUs für die Operatoren und ein konfigurierbares Verbindungsnetzwerk zur Erhaltung der Flexibilität bei der Ausdrucksabarbeitung (Bild 5).

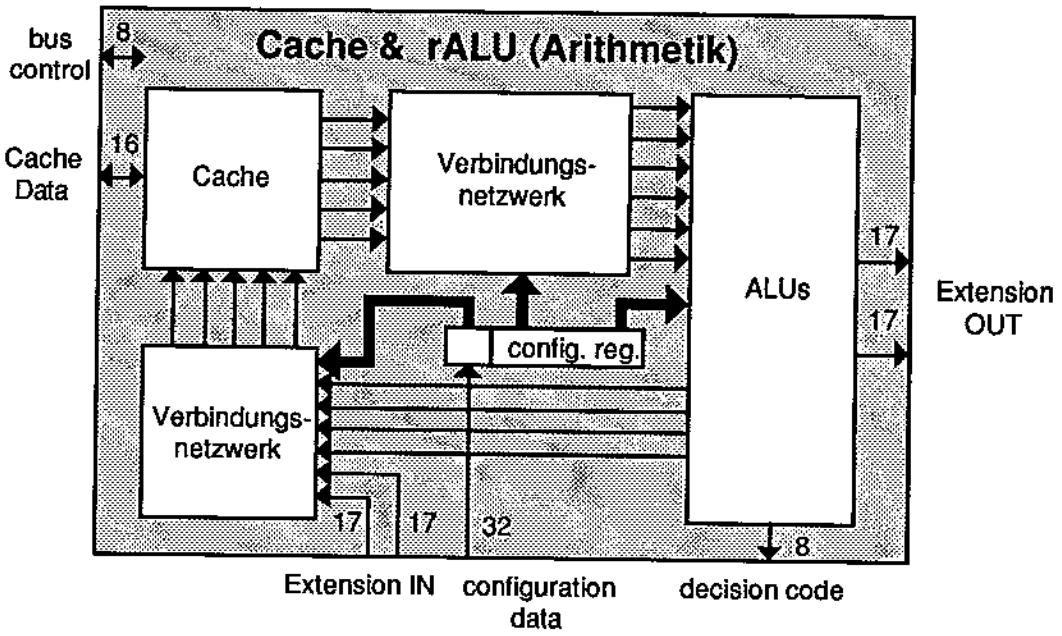

Bild 5: Cache und rALU IC für arithmetische Anwendungen

Draft: Übersicht über Hard- und Softwarearbeiten für die MoM3

Bild 6 zeigt die detailliertere Struktur für einen Cache/Ralu Baustein für Arithmetik bestehend aus 6 konventionellen ALU-Bausteinen mit jeweils 16 Bit Breite. Die ALU's bieten neben den üblichen Operationen (Add, Minus, Shifts, logische Operatoren) auch Multiplikation und Division an, wobei letztere Operationen aus Platzgründen serialisiert werden. Dieser Teil soll in einem Chip mit einer Gatterkomplexität von etwa 15 000 Gattern realisiert werden.

Jeweils zwei dieser Chips werden auf einer Cache/rALU-Platine zusammengefaßt. Nach ersten Abschätzungen können wegen der Limitierung bei den Platinenkonnektoren und des erheblichen Platinen-Interconnects nicht mehr als zwei Cache/rALU Chips auf einer Platine zusammengefaßt werden.

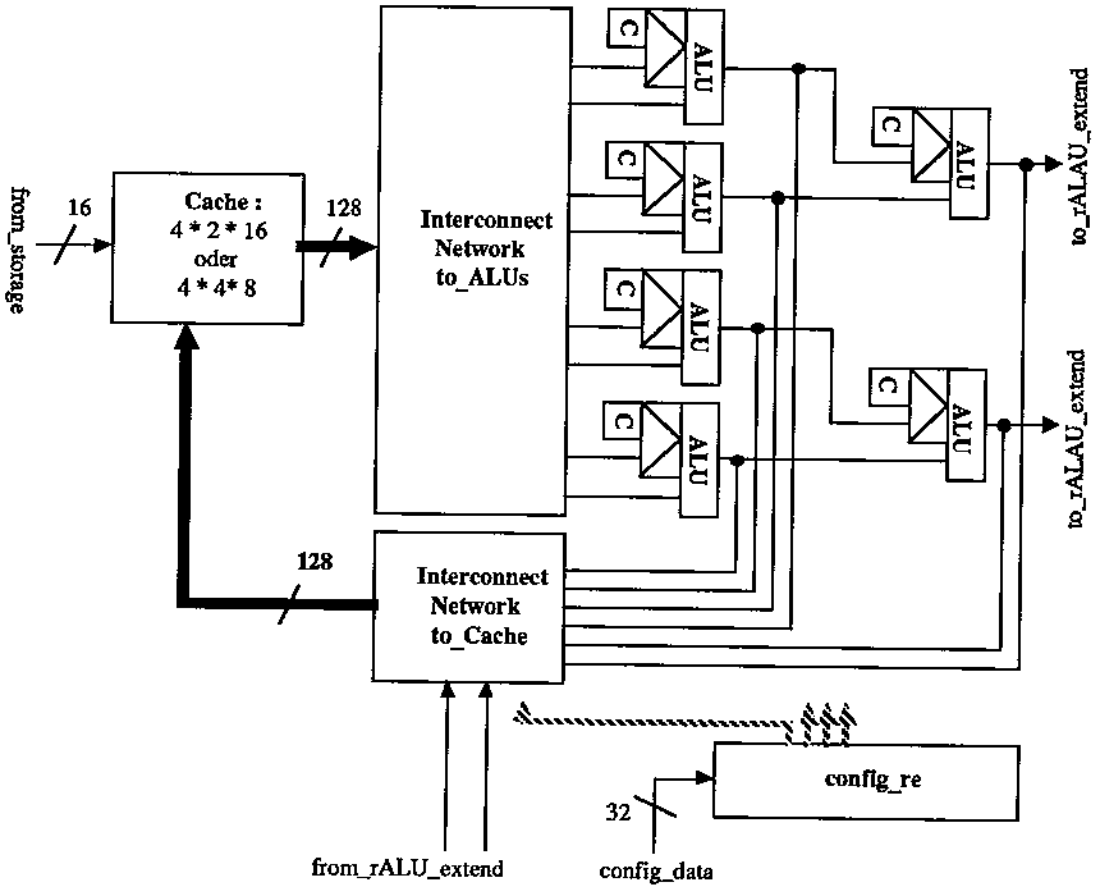

Bild 6: Detaillierte Architektur des Cache/rALU-Chips

Gatterkomplexität:	ca. 15000 Gatter
Spezifikation:	6 Monate inklusive Simulation des Chips mit VERILOG
Standardzellenentwurf*:	5 Monate (CAD-System, Timing/Poszlayout-Simulation, Testentwicklung)
Fertigung*:	2 Monate
Testplatine*:	2 Monate

Der Cache kann standardmäßig für eine Wortbreite von 8 oder 16 Bit konfiguriert werden. Daraus ergeben sich zwei mögliche maximale Cache-Konfigurationen: 4*2*16 (für Integer-Arithmetik) oder 4*4*16 (für Bildverarbeitung).

- Zur Erhöhung der Schachtelungstiefe der Verbundoperatoren und zum Datentransfer zwischen mehreren Caches wird ein rALU-Extension IC benötigt, der bis auf den fehlenden Cache eine ähnliche Struktur hat wie der vorher beschriebene IC (Bild 7).

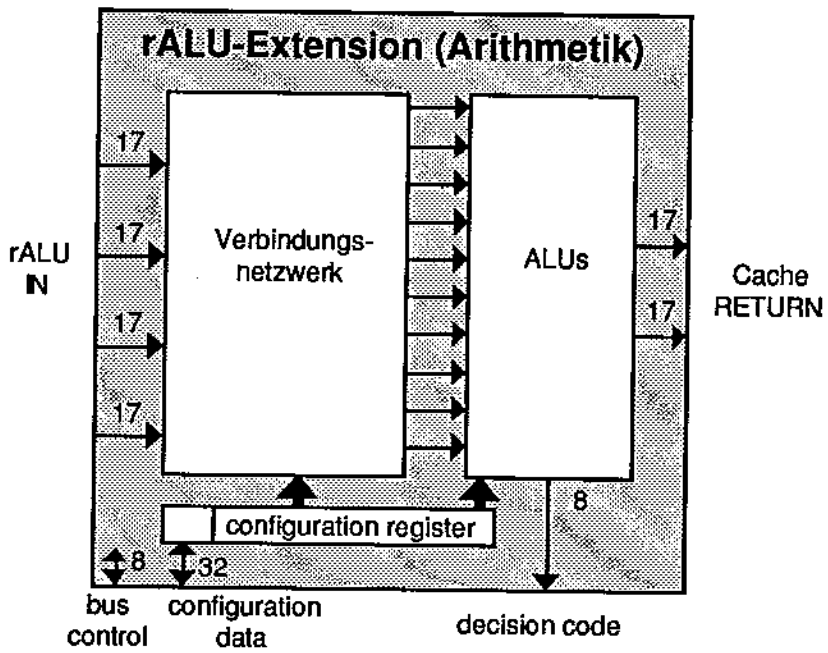

Bild 7: rALU-Extension IC

Gatterkomplexität:	ca. 15000 Gatter
Spezifikation:	6 Monate
Standardzellenentwurf*:	4 Monate
Simulation und Test*:	4 Monate
Fertigung*:	2 Monate
Testplatine*:	2 Monate

- Zur Realisierung der Restkontrolle beim Scan Pattern Wechsel oder bei datengetriebenen Bewegungen wird ein weiterer IC benötigt, der die Entscheidungsdaten aus den rALU-ICs verarbeitet und den Update des Escape Cache steuert (Bild 8). Bis auf die Daten des Escape Cache, die aus Pinoutgründen in des Adressgeneratoren lokal gespeichert werden, muß dieser IC auch die sonstigen Steuerparameter des Xputer speichern und den anderen Einheiten zur Verfügung stellen.

Draft: Übersicht über Hard- und Softwarearbeiten für die MoM3

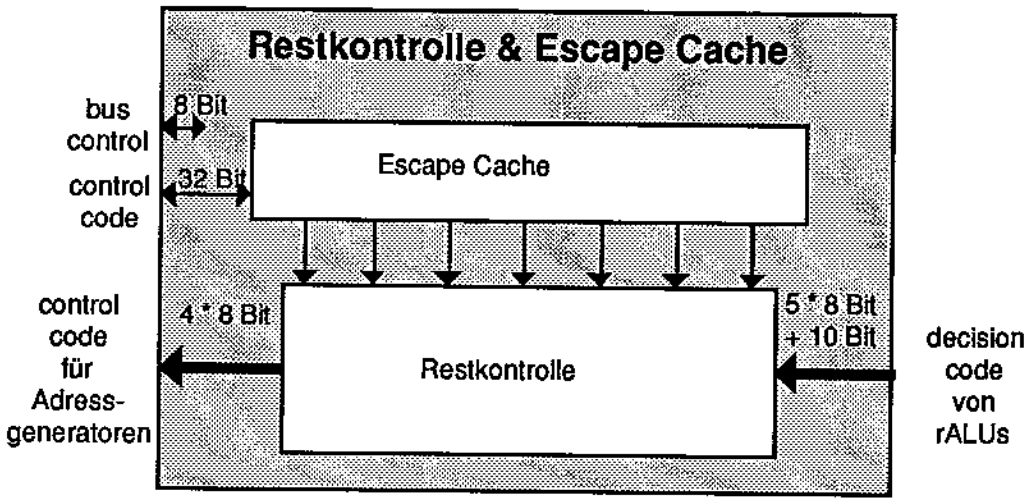

Bild 8: Restkontrolle und Escape Cache

Gatterkomplexität:	ca. 10000 Gatter
Spezifikation:	3 Monate
Standardzellenentwurf*:	4 Monate
Simulation und Test*:	3 Monate
Fertigung*:	2 Monate
Testplatine*:	2 Monate

Das Gesamtsystem wird aus 4 Adressgeneratoren bestehen, die die Caches dreier arithmetischer rALUs steuern und einen Cache für Pattern Matching Anwendungen. Die arithmetischen rALUs werden mit 2 rALU Extension Chips miteinander gekoppelt, und ein IC übt die Restkontrolle über das Gesamtsystem aus. Diese ICs werden über 4 Platinen verteilt und mit einem lokalen Datenpfad verbunden. Eine schematische Darstellung des Ganzen findet sich in Bild 9.

Draft: Übersicht über Hard- und Softwarearbeiten für die MoM3

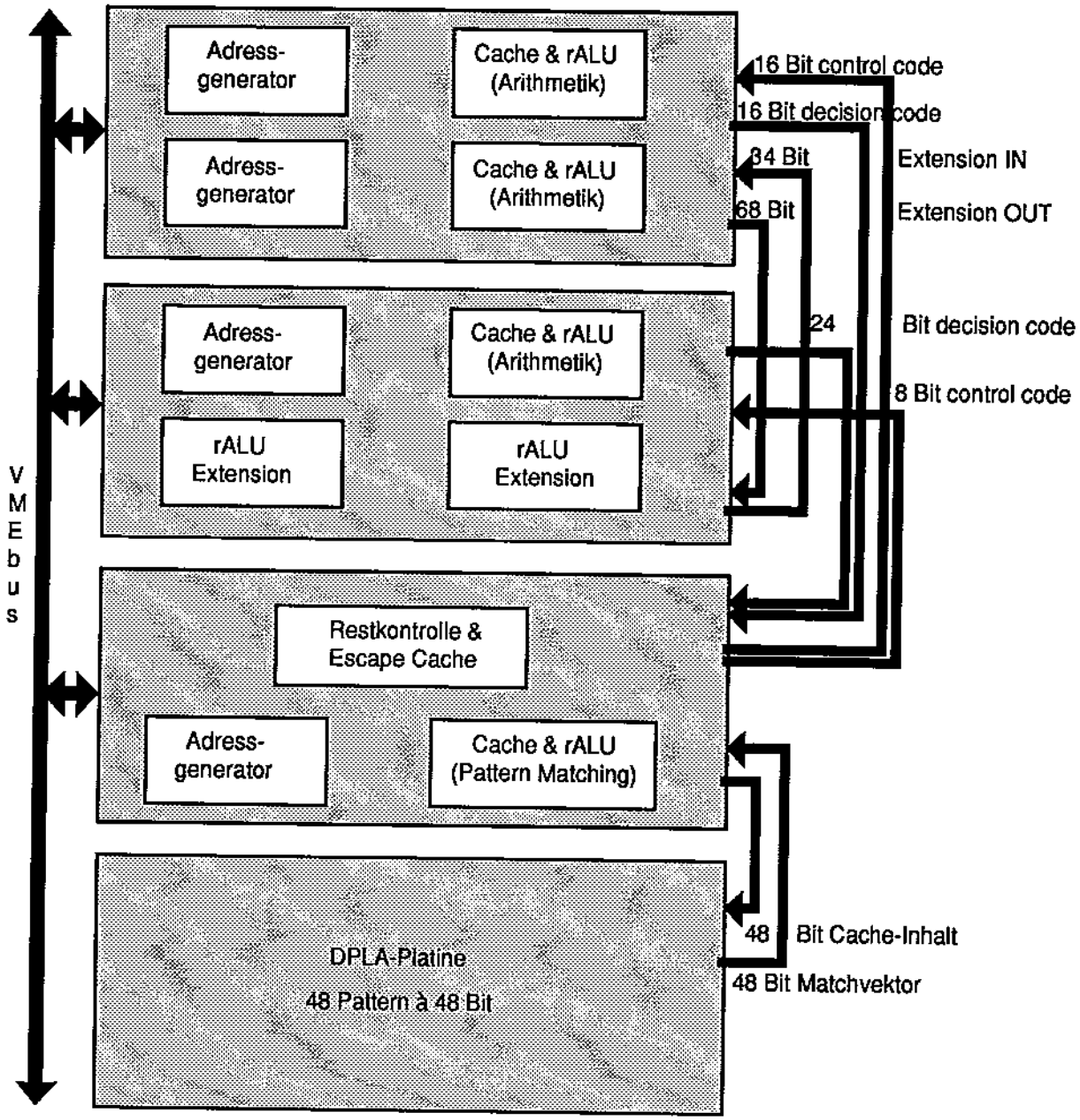

Bild 9: Aufteilung der ICs im Gesamtsystem

2. Software-Anforderungen für die MoM3

2.1. Ziel

Durchgängiger Weg von einer höheren Programmiersprache für systolische und nicht-systolische generische Algorithmen bis auf die Hardware der MoM3.

2.2. Übersicht über die Arbeiten

Momentan wird/ist realisiert:

- **Automatische Generierung eines MoPL-Programms** aus Spezifikation beschrieben in einer höherenProgrammiersprache (SYS2) für **systolische Algorithmen.**
  (Anpassung an MoPL2 : 2 Tage)
- **Spezifikation** der Xputer-Sprache **MoPL2.**
  (Weitgehend durchgeführt)
- **Parser des MoPL-Programms** und Aufbau des abstrakten Programmbaums (APB).
  (In Arbeit (Projektarbeit); Fertigstellung: frühestens Ende April 91 lauffähige Version)

Realisiert werden muß noch:

- **Interpreter** der auf dem generierten APB ansetzt, um mit MoPL2-Programmen arbeiten zu können.
  (Interessent vorhanden. Falls Zusage, früheste Fertigstellung Ende Juli 91)
- **Codegenerator für logische/ arithmetische rALU.**
  (Spezifikation steht noch aus; hängt sehr stark von der definierten rALU-Hardware ab;
  Arbeitsumfang: 1. Spezifikation: 1 Monat; 2. Implementierung: ca 6 Monate
  Ausschreibung der Arbeit frühestens nach Hardware-Spezifikation)

- **Codegenerator der Registersätze für die Data-Sequencer Hardware**
  (Spezifikation steht noch aus; hängt sehr stark von den definierten Data Sequencer Strukturen ab;
  Arbeitsumfang: 1. Spezifikation: 2 Monate; 2. Implementierung: ca 9 Monate
  Ausschreibung der Arbeit frühestens nach Hardware-Spezifikation)

- **Linker** zur Kombination von Data Sequencing und Daten in der Data Map, zur absoluten Adreßberechnung für beide Teile

2.3. Detailliertere Angaben zu den Anforderungen an den Code Generator

Eingabe:

abstrakter Programmbaum (APB), der das analysierte MoPL-Programm widerspiegelt

Ausgabe:

1.) Konfiguration der variablen Teile der rALU
2.) Konfigurationsdaten der verwendeten Caches
3.) Scan Pattern zur Steuerung der einzelnen Data Sequencer

Zur Konfiguration der variablen Teile der rALU gibt es zwei verschiedene Möglichkeiten:
- logische
- und arithmetische rALUs

Für die logischen rALUs sollte eine spezielle Hardware vorgesehen werden, die Teile der derzeit verfügbaren MoM-Hardware verwendet. Eine Abbildung auf die für arithmetische Operationen vorgesehene Hardware wäre entweder sehr ineffizient oder gar komplett unmöglich wie z.B. Implementierung des Design-Rule-Check für CMOS mit einer Vielzahl von Mustern.

Für die arithmetischen rALUs erscheint eine Struktur günstig, die zum Teil aus festen rechnenden Elementen besteht, wobei der Interkonnekt zwischen diesen Elementen mit möglichst vielen Freiheitsgraden konfigurierbar bleibt. Der Code-Generator hat dann die Aufgabe einen vom Benutzer in MoPL spezifizierten Ausdruck so umzusetzen, daß er durch die zur Verfügung stehende Hardware berechnet werden kann und die nötigen Konfigurationsdaten für die Operatoren bzw. zu schaltenden Verbindungen ermitteln.
Falls eine Berechnung mit den vorhandenen Ressourcen nicht möglich sein sollte, erhält der Benutzer eine entsprechende Meldung und muß die fragliche Berechnung in mehrere Teile aufspalten, die sich realisieren lassen und evtl. nacheinander abgearbeitet werden.

Der Code-Generator kann auch die Struktur des aktuellen Ausbaus der Hardware vor dem Benutzer verbergen. Dieser programmiert dann seine Anwendung ohne daß er sich z.B. über die Zahl oder Größe der aktuell im System eingesetzten Caches gedanken machen muß. Zum Beispiel verwendet er in seinem Programm einen Cache der Größe 4*4, im System sind aber aus technischen Gründen nur Caches der Größe 2*4 vorhanden. Der Code-Generator kann dann zwei dieser Caches 'zusammenschalten' und synchron zueinander bewegen, so daß sie wie ein 4*4 Cache benutzt werden können. In einem Konfigurations-File können dann die Informationen über den Stand des Hardware-Ausbaus abgelegt sein, die bei der Verarbeitung verwendet werden.