Massgeschneiderte Echtzeit-Bildverarbeitung : Konzept und Realisierung eines heterogenen Mehrprozessorsystems

Autor(en): **Gunzinger, Anton**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses**

Band (Jahr): **82 (1991)**

Heft 21

PDF erstellt am: **22.07.2024**

Persistenter Link: https://doi.org/10.5169/seals-903030

Massgeschneiderte Echtzeit-Bildverarbeitung

Konzept und Realisierung eines heterogenen Mehrprozessorsystems

Anton Gunzinger

Eine praxisnahe Bildverarbeitungsanwendung benötigt im wesentlichen 10–20 Funktionen, von denen die meisten in Software (langsame Lösung, geringe Kosten) oder Hardware (schnelle Lösung, hohe Kosten) realisiert werden können. Der für die Echtzeit-Bildverarbeitung entwickelte Parallelrechner Sydama-2 erlaubt, ausgehend von der zur Verfügung stehenden Auswertezeit, durch seine modular erweiterbare Hardware und eine optimale Aufteilung der Funktionen auf Hard- und Software, eine kostengünstige Implementierung von Bildverarbeitungsalgorithmen.

Une application très proche de la pratique du traitement de l'image requiert pour l'essentiel 10–20 fonctions, la plupart pouvant être réalisées en logiciel (petit vitesse, petits frais) et matériel (haute vitesse, grands frais). L'ordinateur parallèle Sydama-2 développé pour le traitement en temps réel de l'image permet, du fait du temps d'analyse disponible, grâce à son matériel modulaire extensible et à une répartition optimale des fonctions sur le logiciel et le matériel, des implémentations d'un prix avantageux des algorithmes du traitement de l'image.

Adresse des Autors

Dr. *Anton Gunzinger*, dipl. El.-Ing. ETH, Institut für Elektronik, ETH-Zentrum, 8092 Zürich.

Im Bereich der industriellen Automation, Robotertechnik und Qualitätssicherung besteht ein wachsendes Bedürfnis nach Systemen, die den Sehsinn des Menschen nachbilden. Ein solches System besteht in der Regel aus einem Bildsensor, der die Szene aufnimmt, einer elektronischen Auswerteeinheit (Bildverarbeitungssystem) und einer Steuerung für einen Automaten oder Roboter. Der anspruchvollste Teil in dieser Umgebung bildet dabei das Bildverarbeitungssystem; aus der enormen Datenmenge, die der Bildsensor produziert (bis 50 MByte pro Sekunde), muss die relevante Information (z.B. Anwesenheit und Position eines Objekts) in Echtzeit extrahiert werden. Für einige Aufgaben in der industriellen Bildverarbeitung sind robuste Bildverarbeitungsalgorithmen bekannt; das grösste Problem liegt dabei oft in der benötigten Rechenleistung. Ein Beispiel: Ein Videosensor produziere 10 Millionen Abtastwerte pro Sekunde, für die Bearbeitung eines jeden Abtastwertes seien zum Beispiel 20 Operationen notwendig; damit ergibt sich eine benötigte Rechenleistung von 200 MOPS (Million Operations Per Second). Bei komplizierteren Algorithmen kann der Rechenleistungsbedarf um eine oder zwei Grössenordnungen ansteigen. Solche Rechenleistungen können mit den heute verfügbaren schnellsten frei programmierbaren Prozessoren nicht erreicht werden.

Es hat sich nun gezeigt, dass sich viele Bildverarbeitungsaufgaben mit einer Funktionenkombination aus einem Set von etwa 40 Funktionen lösen lassen. Beispiele solcher Funktionen sind: Filterung, Matrixrechnung, Momentenrechnung, Codierung, Fouriertransformation, «Nächster Nachbar»-Klassierung usw. Auf dem Markt sind einige an diese Funktionen angepasste Spezialprozessoren erhältlich. Um eine konkrete industrielle Bildverarbeitungsaufgabe zu lösen, müssen die entsprechenden Spezialprozessoren zusammengeschaltet und programmiert werden. Ausserdem wird oft zusätzliche Software zur Interpretation der Daten und zur Kommunikation mit anderen Systemen benötigt. Der Aufbau eines nach diesem Prinzip realisierten Spezialrechners ist sehr aufwendig und damit sehr teuer. Es muss deshalb nach Systemen gesucht werden, die eine schnelle und damit kostengünstige Adaption an eine Bilderkennungsaufgabe erlauben.

Ziel des im folgenden vorgestellten Projekts war die Konzeption und Realisierung eines heterogenen Mehrprozessorsystems zur Echtzeitbildverarbeitung. Das Mehrprozessorsystem sollte modular aufgebaut und erweiterbar sein; es sollte in einfacher Weise die Integration der zur Lösung einer Aufgabe benötigten Bildverarbeitungsfunktionen in Hard- bzw. Software erlauben. Die Wahl der Implementierung (Hard- oder Software) ist dabei von der durch die Anwendung vorgegebenen Zeitlimite abhängig.

Es ist bekannt, das der Bau und die Programmierung eines homogenen Mehrprozessorsystems (bei dem alle Teilprozessoren vom gleichen Typ sind) für den allgemeinen Fall noch nicht gelöst ist und auch heute noch Gegenstand vieler Forschungsprojekte ist; mit der Konzeption und Realisierung eines heterogenen Mehrprozessorsystems begibt man sich an die Grenze des heute Machbaren. Am Institut für Elektronik der ETH Zürich ist bereits vor einiger Zeit der Bau eines heterogenen Mehrprozessors gelungen [1; 2]. Der Austausch einer implementierten Funktion (Hardware

oder Software) war in diesem System nahezu unmöglich. Will man einen solchen Austausch berücksichtigen, so wird das System sehr viel komplexer. Wie die untenstehenden Ausführungen zeigen, ist es aber trotzdem möglich, ein heterogenes Mehrprozessorsystem, zugeschnitten auf eine Anwendungsklasse (in diesem Fall industrielle Bildverarbeitung), zu entwerfen und zu bauen.

Dieses heterogene Mehrprozessorsystem beruht auf den folgenden Grundgedanken:

- Beschränkung auf eine Anwendungsklasse. Als Anwendung wurde die industrielle Bildverarbeitung gewählt. Eine typische Anwendung besteht aus 5–15 Funktionen aus einem Set von etwa 40 Funktionen. Mit dieser Annahme lassen sich etwa 80% der industriellen Bildverarbeitungsaufgaben lösen.
- Beschreibung des Verhaltens der Funktionen durch synchrone Datenflussgraphen. Dabei wird angenommen, dass sich von jeder Teilfunktion die Abarbeitungszeit und die Kommunikationsrate zur Programmentwicklungszeit bestimmen lässt. Diese Annahme ist für die meisten verwendeten Funktionen gültig.
- Identische Hard- und Software-Schnittstelle aller Spezialmodule.
- Rekonfigurierbares, modular erweiterbares Hochgeschwindigkeitskommunikations-Netzwerk.

In den folgenden Abschnitten sollen Konzept und Realisierung eines solchen Rechners zur industriellen Echtzeit-Bildverarbeitung näher vorgestellt werden.

Konzept

Es lässt sich zeigen, dass sich viele Bildverarbeitungsaufgaben aus dem Gebiet der Qualitätskontrolle und der industriellen Automation mit einer Bibliothek von etwa 40 parametrisierbaren Funktionen lösen lassen. Die meisten dieser Funktionen sind sowohl als Software-Modul (geringe Kosten, hohe Ausführungszeit) wie auch als Hardware-Modul (hohe Kosten, geringe Ausführungszeit) verfügbar. Die von der Anwendung vorgegebene Auswertezeit kann von einigen Millisekunden bis zu einigen Sekunden variieren. Es lohnt sich deshalb, aus Kostengründen nur für jene Module eine Hardware-Implementierung vorzunehmen, die zur Erfüllung der zeitlichen Anforderungen unbedingt nötig sind. Der Anwender sollte von der Neuprogrammierung des heterogenen Rechners möglichst entlastet werden. Um dies zu erreichen, muss das Verhalten des gesamten Systems in einer von der Implementierung (Spezial-Hardware oder -Software) unabhängigen Art beschrieben werden können. Diese Beschreibung erfolgt hier durch *synchrone Datenflussgraphen*. Dabei wird davon ausgegangen, dass sich die Ausführungszeit, die Kommunikationsrate und die Implementierungskosten einer einzelnen Funktion zur Programmentwicklungszeit vorausberechnen lassen. Diese Randbedingung trifft für viele Funktionen aus der Signal- und Bildverarbeitung zu [3].

Prinzip der Kostenoptimierung

Um die Realisierungskosten zu bestimmen, werden die einzelnen Kosten der Funktionen summiert. Diese Kosten sind von der jeweiligen Implementierung abhängig: Beispielsweise werden für eine Grundkarte 20000 $ und für ein Spezialmodul 4000 $ eingesetzt. Wie später beschrieben wird, können im Rechner programmierbare Gate Arrays (Logic Cell Array, LCA) eingesetzt werden. Je nach geladener Schaltung kann ein solches Gate Array verschiedene Funktionen in Videoechtzeit abarbeiten. Eine solche Schaltung wurde mit 4000 $ veranschlagt. Für die in Software realisierten Funktionen wurden keine Kosten eingesetzt, da diese Funktionen in der Regel mit dem System mitgeliefert werden. Es ist nun Aufgabe eines Optimierungsprogramms für die jeweils vorhandene Aufgabe eine kostenoptimale Hardware zu finden. Diese Optimierung findet zur Programmentwicklungszeit statt. Da es sich bei dieser Optimierungsaufgabe um ein np-komplettes Problem handelt (Anzahl zu untersuchenden Möglichkeiten wächst mit der Fakultät von n), können nicht alle möglichen Implementierungen in absehbarer Zeit getestet werden. Um die Suchzeit zu reduzieren wird ein modifiziertes Depth-First-Suchverfahren verwendet. Nach der Kostenoptimierung ist die Implementierung der einzelnen Funktionen festgelegt.

Kommunikationsoptimierung

Die in Hardware implementierten Funktionen erzeugen und konsumieren in der Regel eine sehr grosse Datenmenge; deshalb ist der zentrale Punkt die Kommunikation. In der Bildverarbeitung wird oft mit relativ hohen Abtastraten (5–30 MHz) gearbeitet. Spezielle Videoprozessoren (z.B. Video AD- und DA-Wandler, lineare und nichtlineare Filter, Korrelatoren, Look-up-Tabellen, Bildspeicher) sind in der Lage, solche Datenströme zu verarbeiten. Um die Daten zwischen diesen Videoprozessoren beliebig austauschen zu können, ist ein flexibles, programmierbares Hochgeschwindigkeitskommunikations- Netzwerk nötig. Ein solches Netzwerk wurde am Institut für Elektronik entwickelt. Es handelt sich dabei um ein Mehrbusse-System mit unterbrechbaren Bussen; ein einzelner Bus kann deshalb mehrmals zum Datentransport verwendet werden [1; 2]. Es kann gezeigt werden, dass sich ein solches Netzwerk sehr gut für die Implementierung von in der Bildverarbeitung oft verwendeten Kommunikationsstrukturen eignet.

Die Auslastung des Netzwerks ist von der Zuordnung der Teilaufgaben an die verschiedenen Prozessorelemente abhängig (Systemkonfiguration). Für die Lösung der Konfigurationsaufgabe wird ebenfalls ein modifiziertes Depth First Search-Verfahren verwendet. Die Systemkonfiguration ist statisch und erfolgt zur Programmentwicklungszeit. Nach der Bestimmung der Konfiguration kann ein ladbares Programm erzeugt werden.

Funktion	Software	I/O-PE	Histogramm-PE	Moment-PE
Bildaufnahme		20		
Histogramm	200		0	
Schwellen-berechnung	10			
Segmentierung	200		0	0
Momente	400			0
Schwerpunkt	5			

Tabelle I Auswertezeit von Teilfunktionen für verschiedene Implementationen

Beispiel

Die Kostenoptimierung soll nun anhand eines Beispiels für ein Trackingsystem näher erläutert werden (Bild 1). Es wird davon ausgegangen, dass sich das Objekt in seinem Grauwert

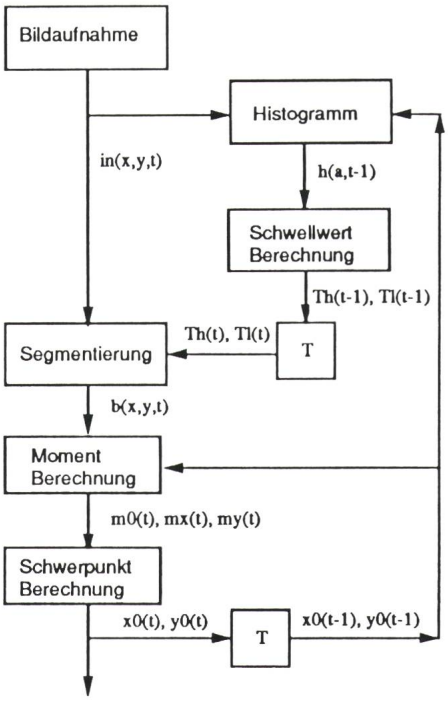

Bild 1 Synchroner Datenflussgraph für ein Trackingsystem

vom Hintergrund unterscheidet; die Objekt-Hintergrund-Separation Bildsegmentierung) erfolge mit Hilfe zweier adaptiver Schwellen, die aus der Grauwertverteilung (Histogramm) des vorangehenden Bildes bestimmt werden. Vom Objekt wird der Schwerpunkt berechnet. In detaillierter Schreibweise: Das einkommende Videobild (x, y, t) wird mit der Hilfe zweier Schwellen $T_l(t)$, $T_h(t)$ in ein binäres Videobild $b(x, y, t)$ umgewandelt, wobei die Objektpunkte markiert sind. Vom Objekt werden innerhalb eines Fensters mit dem Zentrum $x_0(t-1)$, $y_0(t-1)$ die Momente bis erster Ordnung $m_0(t)$, $m_x(t)$, $m_y(t)$ berechnet. Mit Hilfe der Momente kann der Objektschwerpunkt $x_0(t)$, $y_0(t)$ bestimmt werden. Diese Information geht wieder zurück in den Histogramm- bzw. Momentenrechner. Die Segmentierungsschwellen $T_l(t)$, $T_h(t)$ werden aus dem Histogramm $h(a, t-1)$ bestimmt.

Wie die Tabelle I zeigt, lassen sich die verschiedenen Funktionen in Software oder in Hardware in der Form von speziellen Prozessorelementen (PE) implementieren. Jede Implementierung ist mit einer bestimmten Auswertezeit und mit bestimmten Hardware-Kosten verbunden. Für einige Implementierungen beträgt die Rechenzeit 0, da die Berechnungen direkt während des Bildeinzugs erfolgen.

Der vorgestellte Bildverarbeitungsalgorithmus könnte beispielsweise zur Steuerung eines Fahrzeuges, eines Roboterarms oder als Testsystem zur Kontrolle der Positionierung einer Lichtquelle in einem optischen Gerät eingesetzt werden. Je nach Anwendung kann die zur Verfügung stehende Auswertezeit zwischen 20 ms und 4 s variieren. Tabelle II zeigt für einige Implementierungen des Tracking-Algorithmus die Auswertezeit und die dabei entstehenden Hardware-Kosten.

Aus der Tabelle II ist ersichtlich, dass bereits mit einer Minimalkonfiguration des Systems eine Zykluszeit von weniger als 1 s erreicht werden kann (Anwendung in der Robotertechnik). Soll derselbe Algorithmus zur Steuerung eines Strassenfahrzeuges eingesetzt werden, das mit 30 m/s (rund 100 km/h) der Leitlinie nachfährt, so muss die schnellste Implementierung mit dem grössten Systemausbau gewählt werden.

Realisierung

Software

Ein Bildverarbeitungsalgorithmus wird mit Hilfe einer Bibliothek in einer C-ähnlichen Sprache formuliert (Bild 2). Damit aus dieser Sprache der statische Datenflussgraph extrahiert werden kann, gilt auf Funktionsebene die *Single Assignement Rule,* das heisst einer Variablen oder einem Datenstrom kann nur an einer Stelle im Programm ein Wert zugewiesen werden. Im Optimierer wird zuerst der Datenflussgraph extrahiert. Für jede einzelne Funktion werden jetzt die möglichen Implementierungen (Software auf Transputer, Software auf Signalprozessor, Spezial-PE, LCA-PE), die Rechenzeit und die Implementierungskosten angegeben. Nun legt der Optimierer nach einem vorgegebenen Optimierungskriterium (z.B. minimale Kosten bei vorgegebener Ausführungszeit oder minimale Ausführungszeit bei vorgegebener Hardware) eine Zuordnung der Bildverarbeitungsfunktionen zu den PE-Typen fest.

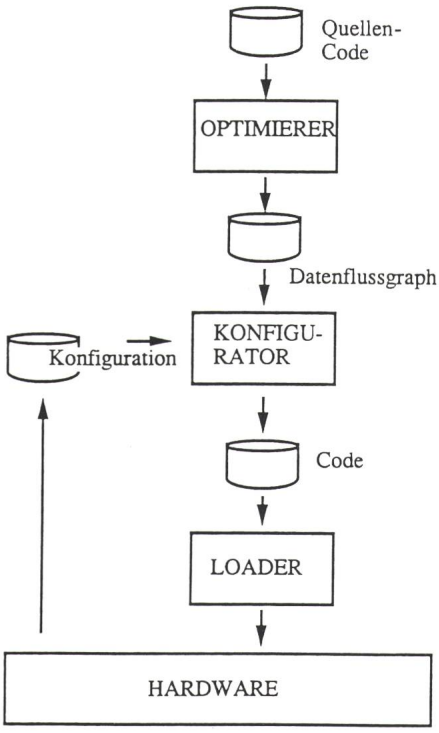

Bild 2 Übersicht über die System-Software

Im Normalfall sind mehrere gleichwertige PE-Typen in einem System enthalten. Die Zuordnung der einzelnen Funktionen zu den einzelnen Prozessorelementen erfolgt nun mit dem Ziel, die Kommunikation zu minimieren.

Da das Netzwerk im Pipelineverfahren arbeitet, muss die Verzögerung der Daten durch Kommunikation und Verarbeitung mitberücksichtigt werden. Der Ausgleich erfolgt durch einstellbare digitale Verzögerungsleitun-

Software	I/O-PE	Histogramm-PE	Moment-LCA	Auswertezeit [ms]	Kosten [k $]
×	×			835	24
×	×	×		435	28
×	×		×	235	28
×	×	×	×	35	32

Tabelle II Auswertezeit und Implementationskosten für den Trackingalgorithmus

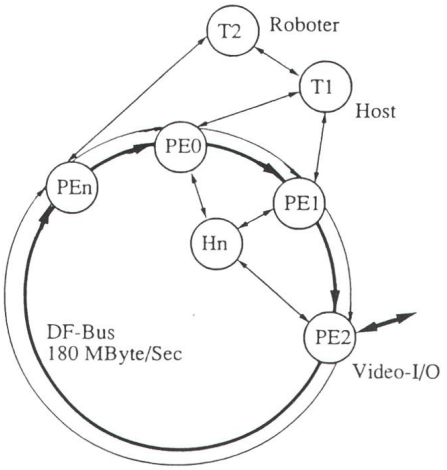

Bild 3 Sydama-2-System mit mehreren Rechnerknoten und dem Anschluss an mehrere Steuerknoten (Host, Roboter usw.)

gen. Nach der Bestimmung der einzelnen Verzögerungszeiten kann ein ladbarer Code erzeugt werden.

Hardware

Ein Sydama-2-System (der Name stammt von SYnchrone DAtenfluss-MAschine) besteht aus mehreren in einem Ring angeordneten Rechnerknoten (Bild 3). Die einzelnen Rechnerknoten sind über einen Hochgeschwindigkeitsbus und über einen Steuerbus miteinander verbunden. Der Hochgeschwindigkeitsbus ist 96 Bit breit und arbeitet mit einer Frequenz von bis zu 15 MHz. Als Steuerbus werden Transputerlinks verwendet. Zusätzliche Transputer, zum Beispiel zur Robotersteuerung oder als Mensch-Maschine-Interface, können in das System integriert werden.

Bild 4 zeigt das Blockschaltbild eines typischen Rechnerknotens. Der Rechnerknoten besteht aus einem Steuerrechner (Transputer) und fünf Prozessorelementen (PE). Der Steuerrechner initialisiert das System, lädt die gewünschten Programme in die PE und kontrolliert den Verarbeitungsablauf. Die fünf PE besitzen je einen Anschluss an den Hochgeschwindigkeitsbus. Sie können dabei sowohl Daten konsumieren als auch produzieren. Drei PE sind als sogenannte LCA-PE ausgeführt. Bei den LCAs (Logic Cell Array) handelt es sich um wiederprogrammierbare Gate Arrays. Je nach geladener Schaltung kann ein LCA-PE als Bildspeicher, Look-up-Tabelle, Transitionscoder, Momentenrechner, binärer Nachbarschaftsprozessor usw. eingesetzt werden. Zwei PE sind als Steckplätze für Steckmodule ausgeführt. Als Steckmodul sind zum Beispiel die folgenden Spezialprozessoren verfügbar: RGB-I/O, FIR-Filter, Rangordnungsfilter, Histogrammrechner, Farbsegmentierer, Signalprozessor.

In Bild 5 ist die Grundkarte abgebildet. Deutlich sind die fünf Adapter an den Hochgeschwindigkeitsbus sichtbar (bei den Steckern). Im weiteren fallen die drei LCA-PE und der Transputer auf. Ein grosser Teil der Karte wurde in SMD-Technologie (Surface Mounted Devices) realisiert. Es wurde eine 3fach-Europakarte mit acht Leiterebenen verwendet. (Nach der manuellen Plazierung der Bauteile erfolgte die Verdrahtung dieser komplexen Karte durch ein CAD-System 100% automatisch.)

Technische Daten

Ein Sydama-2-System besteht aus einer oder mehreren Grundkarten. In einem 19-Zoll-Gehäuse finden bis zu elf Karten (11 Transputer, 55 PE) Platz. Die Sydama-2-Grundkarte weist die folgenden technischen Daten auf:

- **Steuerrechner**: Transputer T 800 mit 4 MByte RAM,
- **Prozessorelemente**: fünf, davon drei LCA-PE und zwei Steckplätze für weitere Steckmodule. Jedes LCA-PE besteht aus einem LCA XC 3090, einem 8 k Fifo und einem Speicher von 256 K zu 9 Bit. In das LCA können die folgenden Funktionen geladen werden:
  - *LUT-PE:* Look-up-Tabelle für zwei Operanden mit 9 Bit Auflösung und einem Resultat von 9 Bit, wobei ein Operand um bis zu 8000 Bildpunkte (10 Bildzeilen) verzögert werden kann, oder Bildspeicher 256 K zu 9 Bit beliebigen Formats.
  - *Transitions-PE:* Segmentiert ein Grauwertbild mit Hilfe von zwei einstellbaren Schwellen in ein Binärbild. Berechnet innerhalb eines einstellbaren Fensters den Transitionscode des Binärbildes.
  - *Moment-PE:* Segmentiert ein Grauwertbild mit Hilfe von zwei einstellbaren Schwellen in ein Binärbild. Berechnet innerhalb eines einstellbaren Fensters die Momente bis zweiter Ordnung.
- **Steckmodule**: Auf jeder Grundkarte sind Steckplätze für zwei Steckmodule vorhanden. Die folgenden Steckmodule sind verfügbar:

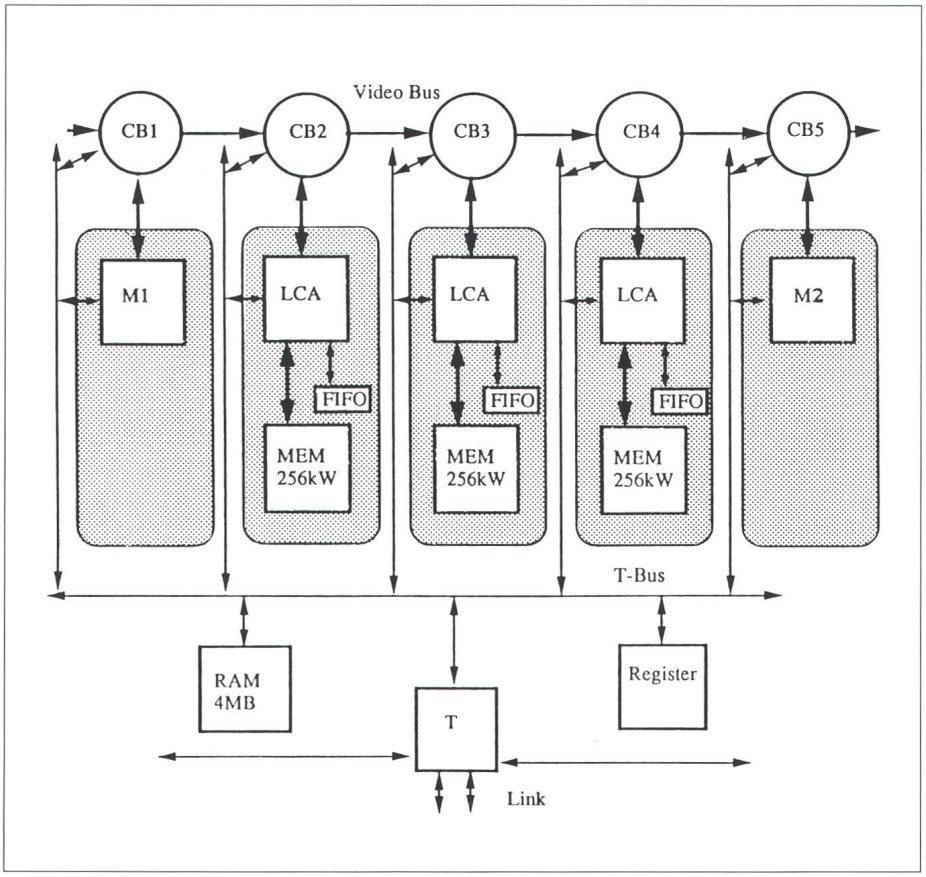

Bild 4 Blockschaltbild der Sydama-2-Grundkarte

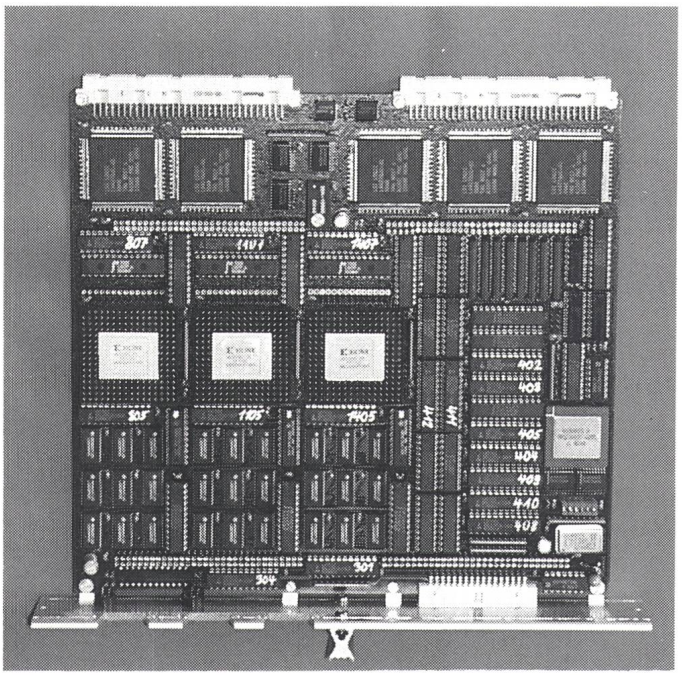

Bild 5
Ansicht einer Grundkarte des Sydama-2-Rechners

- *RGB-I/O-Modul:* wandelt ein analoges Farbsignal in digitale Daten um und erlaubt, verarbeitete Bilder wieder auf dem Bildschirm darzustellen. Für jeden Farbkanal ist im Darstellungspfad eine Look-up-Tabelle vorhanden. Synchronisation: extern, intern, pixelsynchron.
- *FIR-Modul:* ermöglicht die Filterung eines Bildes in Videoechtzeit mit einem 64-Tap-FIR-Filter. Der Filterkern kann zwischen 1×64 bis 8×8 variieren.
- *Rangordnungsfilter-Modul:* ermöglicht die Filterung eines Bildes in Videoechtzeit mit einem 64-Tap-Rangordnungsfilter. Der Filterkern kann zwischen 1×64 bis 8×8 variieren.
- *Histogramm-Modul:* berechnet das Histogramm in zwei vorwählbaren Fenstern in Videoechtzeit.
- *Farbsegmentierungs-Modul:* Transformiert ein Farbsignal mit Hilfe einer 3×3-Matrixmultiplikation in einen anderen Farbraum. In diesem normierten Farbraum können mit Hilfe von sechs freiprogrammierbaren Look-up-Tabellen mit je zwei Eingängen die meisten heute verwendeten Farbklassifikationsverfahren in Videoechtzeit implementiert werden.
- *Signalprozessor-Modul:* Auf diesem Modul befindet sich ein Signalprozessor TMS 320C30 mit 30 MFloppis Spitzenleistung. Der Signalprozessor verfügt über 4 MByte Dram und über zwei Bildspeicher mit je 256 k zu 9 Bit Sram. Die Videodaten gelangen über ein LCA direkt vom Videobus in die Bildspeicher.
- Weitere Steckmodule sind in Planung.
- Der Rechner kann von jedem beliebigen Host mit Transputerinterface (z.B. PC, SUN) programmiert und betrieben werden.

Bild 6 zeigt als Beispiel das Blockschaltbild des Signalprozessor-Moduls. Die Videodaten werden mit Hilfe des LCAs in einen der Bildspeicher eingelesen, während die Daten des anderen Bildspeichers vom Signalprozessor (DSP) ausgewertet werden. Dem DSP stehen 4 MByte Dram für Programm und Daten zur Verfügung. Der Knotenmanager (Transputer) steuert das ganze Modul.

Anwendungen

Viele aus der Bildverarbeitung bekannte Funktionen wie zum Beispiel ortsabhängige Korrekturen, lineare und nichtlineare Lokaloperationen, temporale Operationen, geometrische Bildtransformationen, multispektrale Verarbeitung, Histogrammrechnung, Bildsegmentierung, Erosion, Dilatation, Binärbildverarbeitung, Transientencodeberechnung, Connect-Component Labeling, Merkmalsberechnung und Objektklassierung können auf dem Bildverarbeitungsrechner Sydama-2 in Videoechtzeit berechnet werden.

Der Rechner lässt sich in der industriellen Automation, Robotertech-

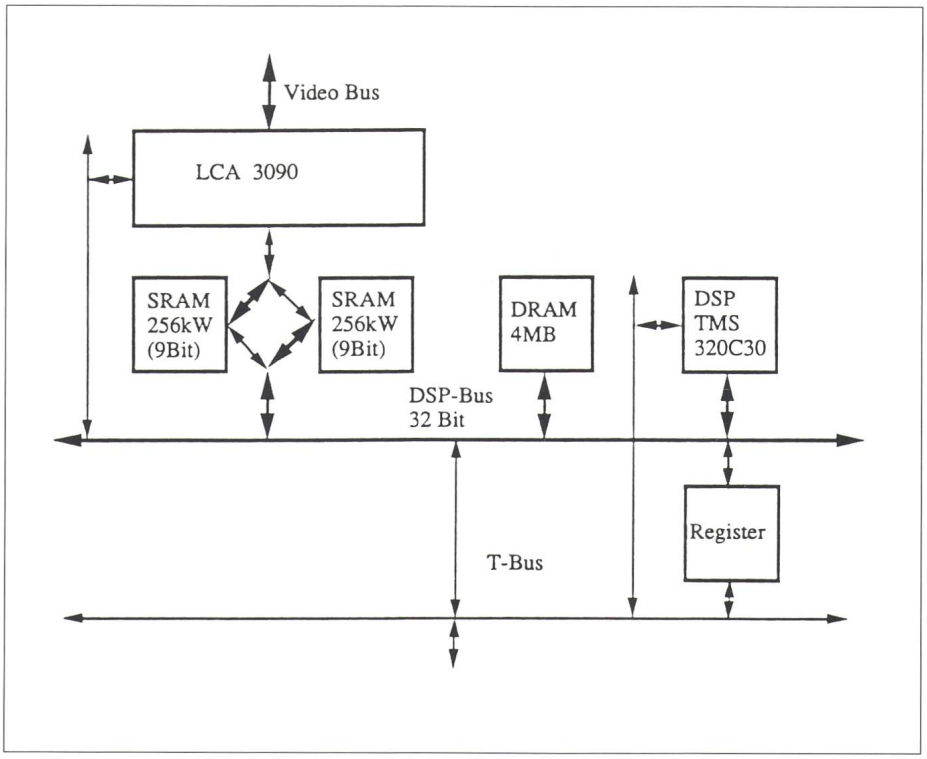

Bild 6 Blockschaltbild des Signalprozessor-Moduls mit dem Signalprozessor TMS 320C30 mit Gleitkomma-Arithmetik

nik, Überwachungstechnik, Qualitätskontrolle, zur autonomen Fahrzeugsteuerung sowie in der Forschung und Entwicklung einsetzen.

Aussicht

Der vorgestellte modulare, heterogene Parallelrechner Sydama-2 gehört zu den leistungsfähigsten und ausbaufähigsten heute verfügbaren Echtzeit-Bildverarbeitungssystemen. Ein Minimalsystem besteht aus einer Grundkarte mit einem I/O-Modul. Das System ist fast beliebig ausbaubar. Ein System mit sieben Karten (35 Prozessorelemente, 7 Transputer) ist funktionsfähig, und eine Kleinserie des Systems wird im Moment in Betrieb genommen. Das System soll in einer weiteren Phase der Industrie zugänglich gemacht werden.

Der systematische, modulare Aufbau ermöglicht – im Gegensatz zu anderen Echtzeit-Bildverarbeitungssystemen – die Programmierung des Parallelrechners in einer anwenderfreundlichen Form. Die Einsetzbarkeit wird anhand praktischer, anspruchsvoller Aufgaben (Abräumen von Postpaketen mit Tiefensensor, autonome Fahrzeugsteuerung mit Hindernisdetektion) demonstriert. Zu einem späteren Zeitpunkt soll auch der Einsatz in industrieller Umgebung demonstriert werden.

Dank

Die Realisierung dieses anspruchsvollen Projektes war nur durch die Zusammenarbeit vieler Leute möglich. Ihnen allen möchte ich an dieser Stelle danken. Ein besonderer Dank gebührt dem Vorsteher des Instituts für Elektronik, Herrn Prof. Dr. W. Guggenbühl für seine Unterstützung an diesem Projekt. Danken für ihre Mitarbeit möchte ich allen meinen Arbeitskollegen sowie den über dreissig Studenten, die in ihren Studien- oder Diplomarbeiten an diesem Projekt mitgearbeitet haben.

Literatur

[1] *A. Gunzinger, S. Mathis, W. Guggenbühl:* Synchroner Datenflussrechner zur Echtzeit-Bildverarbeitung. Bulletin SEV/VSE 79(1988)7, S. 362–367.

[2] *A. Gunzinger*: Synchroner Datenflussrechner zur Echtzeit-Bildverarbeitung. Dissertation, VDF 1990. ISBN 3 7281 17919.

[3] *E.A. Lee, D.G. Messerschmitt*: Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing. IEEE Trans. on Comp. C-36(1987)1, pp. 24–35.