Der MANNA 16x16 Crossbar

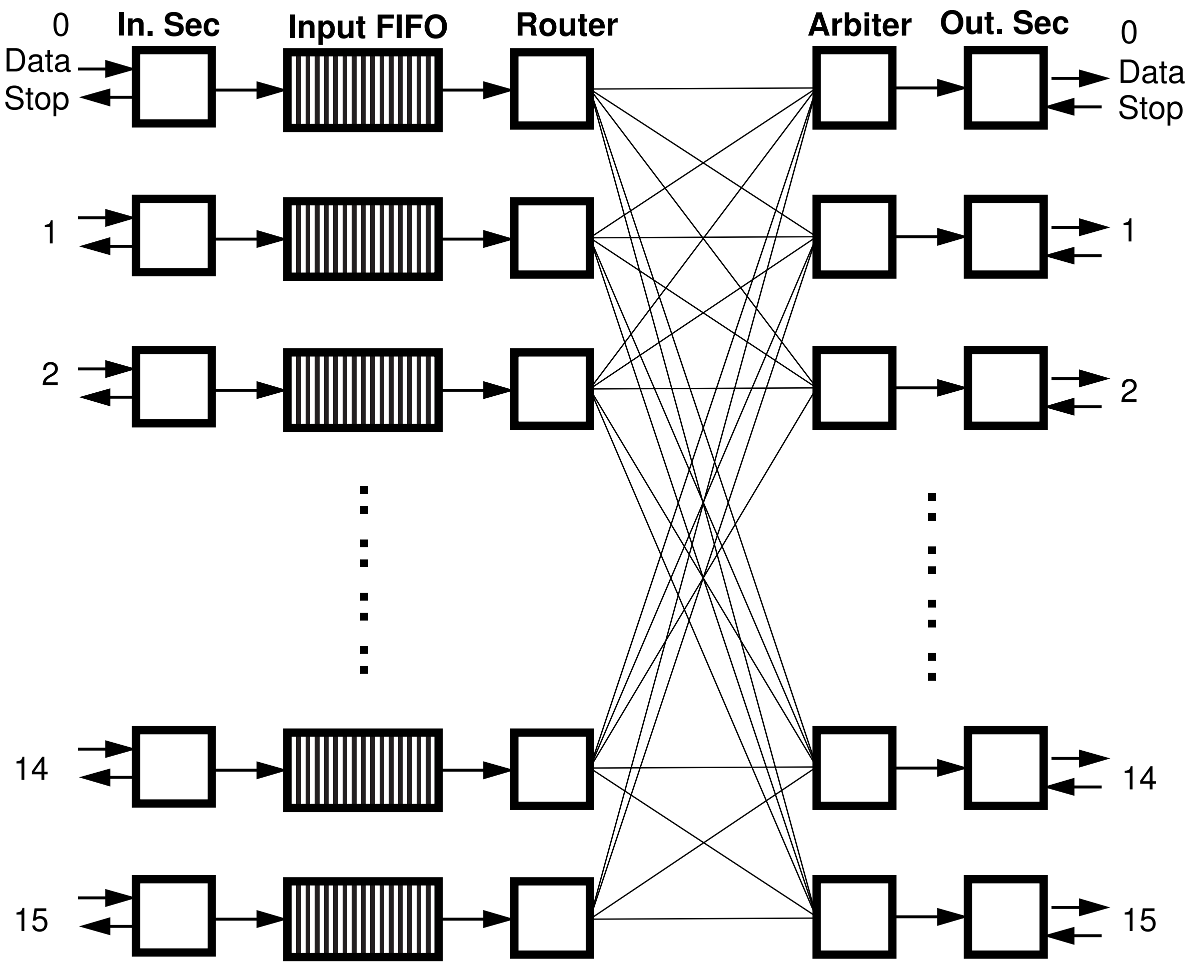
Gesellschaft für Mathematik und Datenverarbeitung mbH

GMD-FIRST

Hardenbergplatz 2

1000 Berlin 12

Der 16x16 Crossbar hat 16 Input Ports und 16 Output Ports. Die Ports sind alle unabhängig voneinander. Ein Input- und ein Output Port können zusammen verwendet werden, um einen Full Duplex-Anschluß für 2 Richtungen (bidirectional full duplex port) zu bekommen. Auch dann bleiben die Ports unabhängig voneinander. Durch einen Router, der in jedem Port integriert ist, kann der Crossbar intern jeden Input Port mit jedem Output Port verbinden.Wenn ein Input Port mit einem Output Port verbunden ist, sagt man, es bestehe eine logische Verbindung zwischen beiden Ports. Zum Weiterleiten aller Daten- und Befehlsinformationen wird, durch einen Routing-Befehl, eine logische Verbindung hergestellt. Sie bleibt offen bis ein Schließbefehl ankommt. Der Sender ist ein Output Port des Crossbars oder des Link Interface, der Empfänger ist ein Input Port.



1. Die Ports

Die Ports haben eine Breite von einem Byte und arbeiten alle synchron, mit einer maximalen Frequenz von 50 MHz (50 MByte/s).Die Ports aller Geräte (devices) müssen den gleichen Takt, Frequenz und Phase, mit einer maximalen Verschiebung (maximal skew) von 5 Prozent haben. Alle Signale verlaufen synchron zum Systemtakt. Der Zustand eines Signals kann sich nur nach einer steigenden Taktflanke verändern und muß dann bis zur nächsten Taktperiode stabil bleiben (siehe Bild 1 und 2).

Die Transfer Einheit (control und data) besteht aus einem Byte Data und einem Kontroll-Bit. Dieses Bit wird benötigt, um zwischen Befehlen und Daten zu unterscheiden. Die übrigen 8 Bits dienen als Daten oder Befehle (data or commands) eines Ports. Alle Wörter, die an einem Input Port ankommen, werden interpretiert und konsumiert oder an den entsprechenden (oder keinem [/dev/null]) Output Port weitergegeben.

Die Signale der Ports sind:

9x Data: Output des Senders -> Input des Empfängers

Diese Signale befördern Daten und Routing-Informationen. Das neunte Kontroll-Bit dient dazu, die Daten und Routing-Informationen mit den anderen 8 Signalen zusammenzukodieren. Wenn “control” eingestellt ist (1), dann transportieren die Datensignale Routing- oder Kontrollinformationen. Ist es nicht eingestellt, haben die Datensignale ein Byte Data, das an den entsprechenden Empfänger weitergeleitet werden muß.

Die Befehle und Daten werden in den 9 Daten-Signalen, wie folgt, kodiert:

0 1 2 3 4 5 6 7 8	
x x x x x x x x 0	Daten-Byte (xxxxxxxx)
a b c d 0 1 1 1 1	Route to port (abcd: 0.11)
x x x x 0 0 1 1 1	Close
x x x x 1 0 1 1 1	Abort
1 1 1 1 1 1 1 1 1	Filler (keine Daten)

- Alle anderen Codes werden nicht ausgewertet, sondern wie Daten behandelt.

Wenn der Sender nichts zu senden hat oder mit dem Senden warten muß, werden Füllkommandos (filler commands) erzeugt. Der Empfänger zerstört die Füllfelder (fillers). Sie werden nicht zwischengespeichert bzw. gepuffert und auch nicht in den Speicher geschrieben.

-STOP#: Input des Senders <- Output des Empfängers

-STOP# benutzt man, um eine weiche Datenflußsteuerung zu realisieren. Ist -Stop# inaktiv (1), kann der Sender bei jeden Taktzyklus “data” und “control” senden. Wird STOP# aktiv (0), muß der Sender so bald wie möglich aufhören, Daten weiterzugeben, aber nicht unbedingt sofort (soft handshake). Der Receiver-Puffer ist dann fast voll, hat jedoch noch Platz für ein paar weitere Bytes (siehe Signale STP_SEL). Der Crossbar regiert auf das Stop-Signal nach 4 Takten, d.h. er kann jeweils nach 4 Takten aufhören bzw. fortfahren zu senden.

Halt (Crossbar-Input):

Halt wird benutzt, um den gesamten Datentransfer im Crossbar zu stoppen. Wenn Halt aktiv wird (1), aktiviert der Crossbar alle seine -STOP#-Signale (0). So kann einer Überhitzung vorgebeugt werden.

STP_SEL0 & STP_SEL1 (Crossbar-Input):

Diese Signale können nur geändert werden, wenn alle FIFOS leer sind (z.B. während Reset). Sie kontrollieren, wann das Output -STOP-Signal jedes Ports aktiviert wird:

SEL0	SEL1	Bedeutung
0	0	Der Input Port hat mindestens noch 21 freie Bytes im Input-Puffer
0	1	Der Input Port hat mindestens noch 13 freie Bytes im Input-Puffer
1	0	Der Input Port hat mindestens noch 9 freie Bytes im Input-Puffer
1	1	Der Input Port hat mindestens noch 5 freie Bytes im Input-Puffer

2. Die Crossbars

Die Bilder 3,4 und 5 zeigen die internen Zwischenverbindungen der 16x16 Crossbars. Die Schnittstellen der Crossbars sind mit den Links des Knotenrechners identisch. Der Input-Link wird durch einen Input-Puffer, einem 32 Byte tiefen FIFO, Router und Requester mit dem Input Port verbunden. Der Output-Link wird durch einen Multiplexer und Arbiter mit dem Output Port verbunden. Alle Arbiter, Multiplexer und Links arbeiten gleichzeitig, parallel und in beiden Richtungen. Jeder Input Port kann eine Anforderung (request) an alle Output Ports stellen. Jeder Output Port hat einen Arbiter. Der Arbiter überprüft alle Request Signale der Input Ports und stellt die notwendigen Verbindungen zwischen den korrespondierenden Input- und Output Ports her.

Die Kontrolle der Crossbar-Komponenten ist maximal verteilt. Es gibt keine zentrale Kontrolle. Jede Zelle in jedem Port hat ihre eigene Kontrollinstanz, die unabhängig vom Rest des Crossbars arbeitet.

2.1. Crossbar - Funktion

Nach “Reset” besteht im Crossbar keine logische Verbindung mehr, und alle Puffer (FIFOs) sind leer. Wenn ein Output Port keine logische Verbindung hat, dann sind alle Output Signale hoch (1, inaktiv), und seine Inputs werden ignoriert. Jedes Daten-/Befehlswort, das bei einem Input Port ankommt, wird im Input-Puffer aufgeschrieben,unabhängig davon, ob der Input Port in einer logischen Verbindung steht oder nicht, außer wenn der Input-Puffer schon voll ist. Sollte das der Fall sein, geschehen Datenverluste. Logische Verbindungen werden durch einen Routing-Befehl aufgebaut und durch einen Close- oder Abortbefehl beendet. Die Befehle “close” und “abort” sind im Crossbar identisch.

Der Router des Input Ports nimmt Wörter vom Input FIFO. Wenn der FIFO leer ist, wartet er, bis etwas ankommt. Hat der Input Port noch keine logische Verbindung zu einem Output Port, nimmt und verwirft er alle Wörter aus dem FIFO bis ein Routing-Befehl ankommt. Der Router nimmt und verarbeitet nur den ersten Routing-Befehl. Er macht eine Anforderung (Req signal active) an den entsprechenden Output Port und wartet, bis er die Erlaubnis bekommt, weiterzumachen (Ack signal des Output Ports). Das verwendete Routing Word wird nicht weitergeleitet, sondern konsumiert. Es besteht nun eine logische Verbindung zwischen dem Input Port und dem entsprechenden Output Port.Der Router nimmt und sendet dann alle anderen Wörter vom FIFO (einschließlich anderer Routing- und Schließbefehle) und sendet sie zum entsprechenden Output Port, bis ein Schließbefehl ankommt. Der Schließbefehl wird zum Output Port gesendet, und die logische Verbindung wird geschlossen (Das Req Signal ist inaktiv). Der Input Port wartet, bis der entsprechende Output Port das Ack Signal inaktiviert hat, und dann wiederholt er die Schleife, um auf einen Schließbefehl zu warten.

Steht ein Input Port in logischer Verbindung, nimmt (und sendet) er nur Wörter vom Input FIFO, wenn das Stop Signal des Output Ports nicht aktiv ist. Wird das Signal aktiv, hört der Input Port in weniger als 4 Taktperioden auf, Wörter zu senden. Nach einem Schließbefehl sendet der Input Port denselben Schließbefehl weiter zum Output Port und wartet, bis er aus dem Crossbar herausgekommen ist. Dann inaktiviert er das Req Signal, um das Ende der Botschaft zu signalisieren, und die logische Verbindung zu schließen.

Der Arbiter des Output Ports schaut sich alle Req Inputs an. Er teilt alle Anforderungen (requests) einer Round Robbing Form zu. Ist der Port frei, stellt er logische Verbindungen her. Der Arbiter bestätigt die Verbindungsanforderung (connection request) mit dem internen Signal Ack, und alle Output-Signale des Output Ports werden mit dem entsprechenden Input Port verbunden. Die Verbindung bleibt bestehen, bis das interne Signal Req inaktiviert wird (nach einem Schließbefehl).

Pin und Gate Count

Input Ports 16x:		Output Ports 16x:
9x Daten	Input	9x Daten
-STOP	Output	-STOP

I/O Signale für 16 Ports 16* (10*2)	240
Crossbar Kontrolle	
(Reset, CLK, Test_out, Halt, SEL0, SEL1)	5
I/O Signale	245
VCC/Ground	> 100
Insgesamt	391

```
Setup-time:            min. -2.8 ns
Hold-time:             min. 4.8 ns
clock-to-out:          2.5 to 13.5 ns
```