

Ein RMNC3 System-on-a-Chip: Ein Migrationspfad zu schnellerer RMNC-Hardware?

Thomas Sailer, HB9JNX/AE4WA

11. April 2001

1 Einleitung

Das RMNC-Digipeatersystem erfreut sich grosser Beliebtheit in Westeuropa. Die aktuelle Hardware, der RMNC3 [1], wurde jedoch schon 1993 vorgestellt und hat seither leider immer noch keinen Nachfolger gefunden. Die Hardware erfüllt die Ansprüche heute nur noch eingeschränkt.

Feldprogrammierbare Logikbausteine (Field Programmable Gate Arrays, FPGA) haben sich als Ersatz von applikationsspezifischen IC's für kleinere Stückzahlen etabliert. Heute sind grosse FPGA's recht günstig erhältlich, so liegt die Idee nahe, die gesamte RMNC3-Hardware mit einem FPGA zu realisieren. Darüber soll in diesem Artikel berichtet werden.

Begonnen hat dieses Vorhaben als Spaßprojekt. Ich wollte einmal moderne Konzepte der Prozessortechnik [2] selber ausprobieren, und hierzu ist der 6809 als Beispiel nicht die schlechteste Wahl.

Die Schaltung könnte als Migrationsstrategie auf eine neue RMNC-Hardwareplattform nützlich sein. Ein neuer RMNC wird auf jeden Fall ein FPGA enthalten. Das RMNC-Busprotokoll entspricht keinem Industriestandard, wofür es fertige IC's geben könnte. Daher eignet sich ein FPGA ideal, einen Buscontroller zu implementieren, der weniger Arbeit vom Prozessor abverlangt als die RMNC3-Implementation. Eine neue RMNC-Plattform wird also neben einem neuen Prozessor, wohl einer 32bit RISC CPU, auch ein FPGA enthalten. Mit der vorliegenden Schaltung könnte nun Hard- und Softwareentwicklung für die neue Plattform entkoppelt werden. Die Hardware könnte erst mal als RMNC3 mit der bestehenden Software betrieben werden, wobei der RISC-Prozessor arbeitslos wäre, und dann per Softwareupgrade zum vollwertigen Nachfolger zu mutieren.

Eine weitere denkbare Strategie ist die Implementation einer 32bit RISC CPU direkt im FPGA.

Die Schaltung erlaubt es der RMNC3-Plattform überdies, wieder auf der Moore-Kurve zu reiten, d.h. wieder direkt von der Leistungssteigerung in der Halbleiterindustrie zu profitieren. Währenddem keine neuen 6809-Bausteine entwickelt werden, werden laufend neue FPGA-Familien und neue, schnellere Mitglieder bestehender Familien vorgestellt. Ein RMNC3 in VHDL lässt sich daher mit wenig Aufwand auf neue Bausteine portieren.

2 Der Softwaresimulator

Begonnen hat das Projekt mit dem Erstellen eines Softwaresimulators der RMNC3-Hardware. Ein Softwaresimulator erlaubt es, Ideen beinahe in Echtzeit auszutesten. Ausserdem kann ein Softwaresimulator Instruktions traces generieren, die dann mit der Simulation der Schaltung verglichen werden können.

Der Softwaresimulator verzichtet bewusst darauf, die Peripheriebausteine vollständig zu simulieren, sondern lediglich die Fähigkeiten, die FlexNet benutzt. Der Softwaresimulator hat es ermöglicht, erst einmal herauszufinden, was wirklich gebraucht wird.

3 Die Schaltung

Figur 1 zeigt das Blockschaltbild der RMNC3-Hardware und damit auch dieser Schaltung. Sie besteht aus dem Prozessor M6809 [6], dem Serial Communications Controller (SCC) Z8530 [8], dem Versatile Interface Adapter (VIA) 6522 [5], und dem Modem, welches auf Original-RMNC3 grösstenteils diskret aufgebaut war.

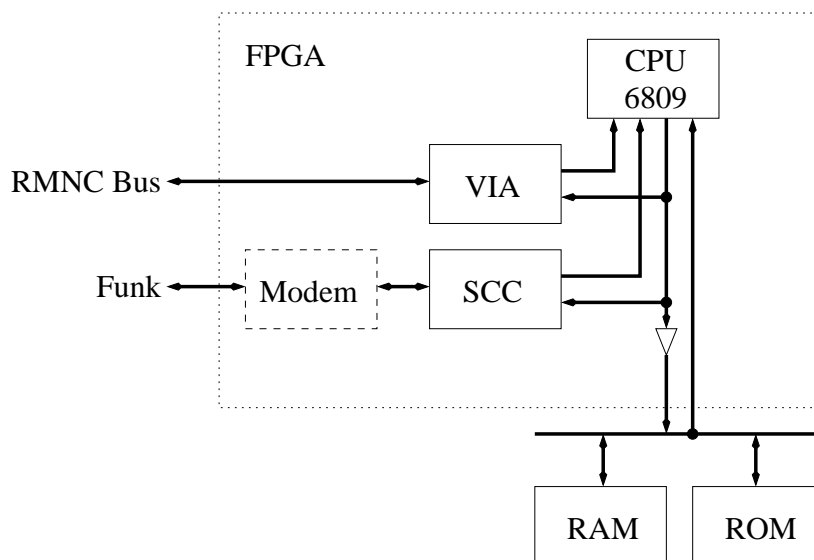


Abbildung 1: Blockdiagramm des Systems

3.1 Der Prozessor

Das zentrale und komplizierteste Bauteil ist der Prozessor. Er implementiert den Befehlssatz des 6809, ist aber keine 1:1 Kopie. Hauptaugenmerk war die Geschwindigkeit, die Grösse der Logik war eher zweitrangig.

Figur 2 zeigt das Blockschaltbild des Prozessors. Der Prozessor besteht aus vier Pipeline-stufen. Die beiden Decoderstufen zerlegen die zum Teil recht komplizierten 6809-Instruktionen

3.1 Der Prozessor

in mehrere $\mu 68$ Operationen, d.h. Instruktionen einer einfachen RISC-Maschine, die dann in der Ausführungseinheit ausgeführt werden können.

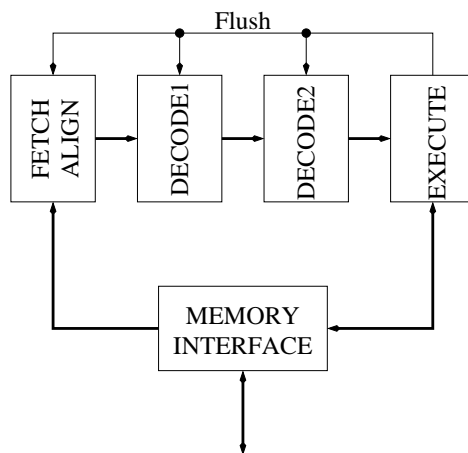


Abbildung 2: Blockdiagramm des Prozessors

Fetch/Align Die Fetch/Align Stufe lädt erst einmal Instruktionen aus dem Speicher in ein 64bit-Register. Ein Schatten-Programnzähler liefert dafür die Adresse. Eine 6809 Instruktion kann neben dem Opcode-Byte, welches den Befehl identifiziert, noch ein Index-Byte, welches den Adressierungsmodus bestimmt, ein oder zwei Byte Konstanten und ein Präfixbyte enthalten. Die Fetch/Align-Stufe zerlegt eine Instruktion in ihre Bestandteile und bestimmt die Länge. Die Länge ist in dieser Stufe nötig, um den Schatten-Programnzähler inkrementieren zu können.

Decode1 Die Decode1-Stufe berechnet aus der zerlegten 6809-Instruktion alle $\mu 68$ -Instruktionen, die zur Ausführung der 6809-Instruktion nötig sind.

Decode2 Die Decode2-Stufe serialisiert die in der Decode1-Stufe berechneten $\mu 68$ -Instruktionen und schickt sie in der richtigen Reihenfolge zur Ausführungseinheit. Der Decoder kann eine $\mu 68$ -Instruktion pro Takt generieren, falls die Fetch/Align-Stufe Daten bereithält.

Execute Die Ausführungseinheit führt die vom Decoder produzierten $\mu 68$ -Instruktionen aus. Das Herzstück der Ausführungseinheit ist ein Register File mit acht 16bit-Registern, zwei Read-Ports und einem Write-Port. Im Register-File sind einerseits die in der 6809 Architektur sichtbaren Register, andererseits weitere Register, die vom Decoder z.B. zur Adressberechnung verwendet werden. Alle $\mu 68$ -Instruktionen mit Ausnahme der Load/Store Befehle werden in einem Takt ausgeführt. Die Anzahl Taktzyklen für Load/Store-Befehle hängt vom Speicherinterface ab. Sprungbefehle und andere Instruktionen, die in den Programnzähler

schreiben lösen einen “Pipeline flush” aus, d.h. die Pipeline wird geleert und der Schattenprogramm Speicher in der Fetch/Align-Einheit geschrieben.

Memory Interface Das Speicherinterface verbindet die Ausführungseinheit und die Fetch/Align-Einheit mit der Aussenwelt. Das Speicherinterface ist 32bittig ausgelegt, damit auch 16bit-Lade- und Schreibebehle grösstenteils nur einen Buszyklus benötigen.

Dieser Prozessor besitzt einige Unterschiede zum Original. Zum einen ist die Geschwindigkeit deutlich grösser. Der Original-68C09 kann zwar mit 12 MHz getaktet werden, benötigt aber vier Takte für einen Buszyklus. Die schnellsten Instruktionen benötigen aber zwei Buszyklen, sodass der Prozessor bestenfalls 1.5 MIPS liefern kann. Der vorgestellte Prozessor läuft derzeit mit etwa 25 MHz auf einem Xilinx Spartan2 XC2S150-5. Der kritische Pfad liegt im Align-Netzwerk, und sollte mit etwas Optimieren noch verkürzt werden können. Der Prozessor verarbeitet eine μ 68-Instruktion pro Zyklus, für sinnvolle Befehle sind mindestens zwei μ 68-Instruktion nötig, sodass der Prozessor 12.5 MIPS liefern kann. Die FlexNet-Software enthält zwar einige Timing-Schleifen, um den Prozessor-Takt und den SCC-Takt zu bestimmen, um damit Teilfaktoren für Zeitgeber und Modembitrate zu berechnen, diese Schleifen haben aber eine Begrenzung, sodass die Software auf dieser Hardware einfach immer die im Original-RMNC3 vorgesehenen Maximalfrequenzen annimmt. Die von FlexNet geschriebenen Teilfaktoren können daher in der Hardware korrigiert werden.

Ein weiterer Unterschied zum Original besteht im Verhalten bei Selbstmodifizierendem Code. Da die Fetch/Align-Einheit der Ausführungseinheit um einige Instruktionen vorausseilt, werden Modifikationen im Instruktionsstrom unmittelbar vor der ausgeführten Instruktion u.U. nicht gesehen. Die RMNC/FlexNet Software enthält aber mit Ausnahme der Kopierrou-tinen vom ROM ins RAM keinen selbstmodifizierenden Code, und dieser Fall bereitet keine Probleme, da ein Sprungbefehl zum kopierten Code die Pipeline leert.

3.2 Die Peripherie

Vom VIA und dem SCC ist nur implementiert, was FlexNet auch wirklich benötigt. Beim SCC ist insbesondere nur der HDLC-Modus implementiert.

3.3 Resultate

Die Schaltung wurde für den Xilinx Baustein XC2S150-5 [7] synthetisiert. Dieser Baustein kostet derzeit etwa 24 US Dollar (Einzelstückpreis bei Avnet/Marshall, Stand Februar 2001) und liegt also durchaus in dem Bereich, was M6809, Z8530 und 6522 zusammen auch kosten, wenn man sie denn überhaupt kriegt.

3.3.1 6809

Der Prozessor läuft mit etwa 25 MHz auf obigem Baustein und benötigt etwa 1000 von total 1704 Configurable Logic Blocks (CLB's).

Die VHDL-RMNC3-Hardware wurde und wird ausgiebig simuliert und mit dem Softwa-resimulator verglichen.

3.3.2 32bit RISC Prozessor

Um eine Vorstellung davon zu kriegen, wie gross und wie schnell ein 32bit RISC Prozessor auf dem Xilinx FPGA ist, wurde eine solche CPU implementiert. Die CPU implementiert die meisten Befehle der Hitachi SuperH Prozessorfamilie [3, 4], mit Ausnahme des Multiplikationsbefehles und einigen Befehlen, die der Multiprozessorsynchronisation dienen. Ausserdem verzichtet die CPU auf Illegal Instruction und Address Exceptions. Die CPU enthält ausserdem einen 2kByte zweivegeassoziativen Cache.

Der Prozessor läuft auf dem XC2S150-5 mit 24 MHz und benötigt etwa 800–1000 CLB's, je nachdem, ob eine vollständige Instruktionsdecodierung gewünscht wird oder nicht, d.h. ob undefinierte Befehle als NOP's ausgeführt werden sollen oder ob sie undefinierte Resultate haben dürfen.

Die meisten Instruktionen benötigen dieselbe Zyklenzahl wie die Hitachi-CPU's. Die Ausnahme sind die Lade- und Speicherbefehle. Ladebefehle benötigen 2 Zyklen bei einem cache hit und Speicherbefehle 3 Zyklen. Der Grund hierfür ist die Architektur der Xilinx Block-RAM's, welche die Cache-Daten und Tags enthalten.

4 Zusammenfassung

Dieser Artikel stellte eine synthetisierbare VHDL-Beschreibung der RMNC3-Hardwareplattform vor. Diese Beschreibung kann dazu benutzt werden, einen RMNC3 auf einem einzigen FPGA als System-on-a-Chip (SoC) realisiert werden. Die Rechenleistung des Prozessors ist deutlich grösser als die des Original-6809.

Die Schaltung wurde ausgiebig simuliert, um die Funktionsfähigkeit zu garantieren. Eigentliche Hardware, d.h. ein FPGA auf einer Eurokarte und einem RMNC-Busstecker, existiert aber nicht, weil dem Autor derzeit die Lust aufs Platinenmachen abgeht.

5 Ausblick

Es wäre durchaus interessant, diese Schaltung mal in Realität laufen zu lassen. Dazu müsste jemand mal einen Prototypen aufbauen.

Aber auch wenn diese Schaltung nie in Realität benutzt wird, so hats dennoch Spaß gemacht und liefert hoffentlich Gunter, (D)K7WJ eine Entscheidungsgrundlage für die Architektur der nächsten RMNC-Generation.

Literatur

- [1] DK7WJ Gunter Jost. RMNC3 – einer für Alle(s). *Adacom Magazin*, 6, 1993.
- [2] John L. Hennessey and David A. Patterson. *Computer Architecture, A Quantitative Approach*. Morgan Kaufmann, second edition, January 1996.
- [3] Hitachi. *SH7032/SH7034 Hardware User's Manual*, September 1994.

LITERATUR

- [4] Hitachi. *Hitachi SuperHTMRISC engine SH-2E Programming Manual*, March 1999.
- [5] MOS Technology, Inc. *MCS6522 Versatile Interface Adapter*, November 1977.
- [6] Motorola, Inc. *MC6809-MC6809E 8-Bit Microprocessor Programming Manual*, March 1981.
- [7] Xilinx, Inc. *Spartan-II 2.5V Family Field Programmable Gate Arrays*, March 2000.
- [8] Zilog, Inc. *Serial Communication Controllers Product Specifications Databook*, 1995.