The Design and Test of a Smartcard Chip Using a CHAIN Self-timed Network-on-Chip

W.J.Bainbridge, L.A.Plana, S.B.Furber
Computer Science Dept, The University of Manchester, UK
jbainbridge@ieee.org, lplana@unexpo.edu.ve, sfurber@cs.man.ac.uk

Abstract

The CHAIN self-timed Network-on-Chip (NoC) architecture provides a flexible, clock-independent solution to the problems of system-on-chip (SoC) interconnect. In this paper we look at the use of CHAIN in a low-performance, smartcard chip to connect two self-timed processors and a range of memories and peripherals. Key design-time advantages provided by the use of CHAIN in this design included the ability to operate a very-narrow, high-frequency network fabric using serial communication without the need for high frequency clocking, rapid assembly in the final stages of the design and the avoidance of the need to perform timing analysis or validation on the SoC interconnect. Additionally we describe a bare port that provided direct access to the CHAIN fabric which was instrumental in testing and debugging the smartcard chip.

1. Introduction

The question of how to connect the many components of a system-on-chip (SoC) is becoming increasingly difficult to answer as systems become more complex and as the relative delay costs of wires become more significant with every CMOS feature size shrink.

One answer to this question is to use a self-timed interconnect mechanism to connect a mixture of IP blocks which may themselves be constructed using either self-timed or synchronous design styles. In fact, the ITRS roadmap recognizes that such Globally Asynchronous, Locally synchronous (GALS) systems may be inevitable in the not too distant future. This approach is a far cry from today's approach of using legacy buses, such as AMBA [1] and CoreConnect [2], designed around a processor interface.

This paper describes the use of a self-timed network-on-chip to meet the interconnect needs of a secure smartcard chip. It also describes several test components that were incorporated into the chip and that allowed successful testing and debugging of the fabricated chip.

2. The G3card smartcard chip

Figure 1 shows a block diagram of the major components of an entirely self-timed system-on-chip including two variants of the SPA processor [3], a complete implementation of the ARMv5T ISA [4], three on-chip memory blocks (for program and/or data), a Memory Protection Unit (MPU), timers, a random number generator (RNG), a smartcard UART [5] for communication with a card-reader and a small CHAIN network-on-chip connecting these together. The chip was designed and fabricated to evaluate the circuit-level effects of self-timed design on the security of the system, but in doing so provided a vehicle to:

- demonstrate the syntax directed translation method of the Balsa synthesis tool [6] on an entire processor core;
- prove the capabilities of the CHAIN self-timed Network-on-Chip in a low-performance system.

This paper presents measurements and an analysis of the CHAIN network implementation used on the chip and a description of the support it provided for debugging implementation errors in the final silicon.

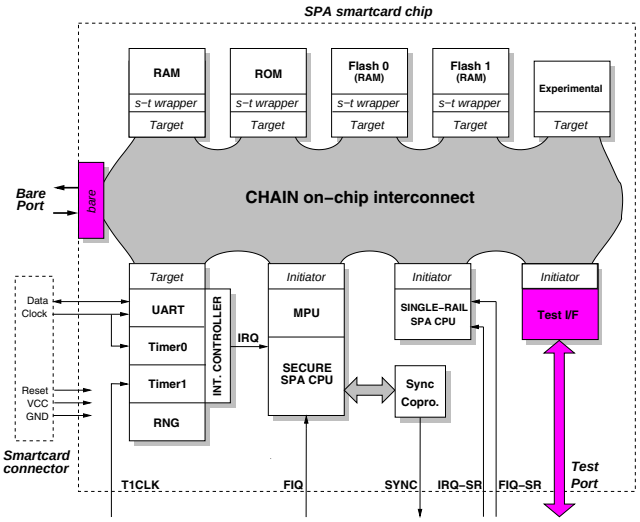

Figure 1: Smartcard system architecture

3. The CHAIN architecture

CHAIN [7] is an approach to SoC interconnect using a network of low-cost switches and narrow links all operating using self-timed protocols. The links and switches, which can be ganged together for increased throughput, are then used in conjunction with well-defined packet formats to provide virtual connections between client devices.

3.1. Self-timed link protocol

The individual CHAINlinks, illustrated in figure 2, comprise five forward-going signals and an explicit acknowledge signal to allow self-timed flow control. The data-encoding used on these links is a 1-hot code allowing either 2-bits of data or a control marker, end-of-packet (eop), to be sent on each communication. A return-to-zero protocol is used so that after each communication, the link returns to the idle state. There are thus four signal transitions (two data, two acknowledge) for every two-bit communication.

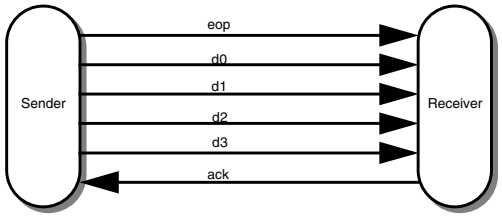

Figure 2: CHAINlink

3.2. Scalable performance

The significant delays incurred due to long wires on small feature-size CMOS processes mean that repeater stages are often required to repower the signal, resulting in propagation delays proportional to the connection length.

These repeaters can easily be modified so that they behave as pipeline latches, allowing the cycle-rate of the links to be increased. This type of change can be easily accommodated since there is no (slow) global clock limiting the fabric's performance. A suitable circuit of a CHAIN pipeline latch is shown in figure 3. The key features in the circuit are the datapath latches (actually Muller C-elements [8], similar to transparent latches) the OR-gate detecting the presence of a valid, latched output and generating an input-acknowledge.

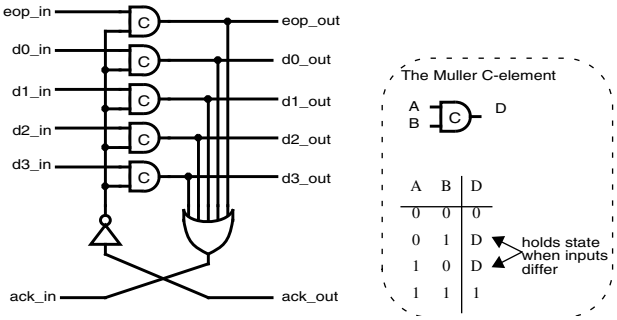

Figure 3: CHAIN pipeline latch

The variation in the cycle-rate of a CHAINlink stage for different link lengths, based on SPICE RC-network simulations for a TSMC 0.18µm CMOS process with aluminium wires, is shown in figure 4. For a 2mm link, it can be seen that the cycle-rate is about 380MHz, giving a throughput of about 0.75Gb/s/link. For higher throughputs, links can be ganged together to form a wider datapath. It should be noted that the repeaters used were only 2x standard drive cells. The 2mm link cycle-rate can be improved to about 500MHz using cells with stronger outputs.

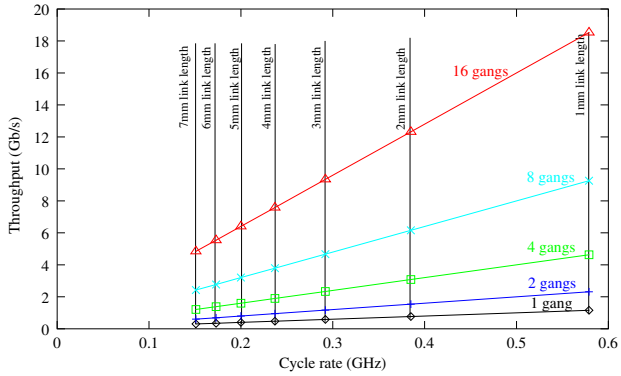

Figure 4: CHAINlink cycle-rate v length

3.3. Packet switching

The pipeline latch/repeater shown in figure 3 can be extended [7], through the addition of extra logic, to support steerable forks and joins between links. To control the steering of data by these units, which form a distributed network, the data is formatted into packets, as shown in figure 5. Each packet contains a route, one or more messages and an end-of-packet marker.

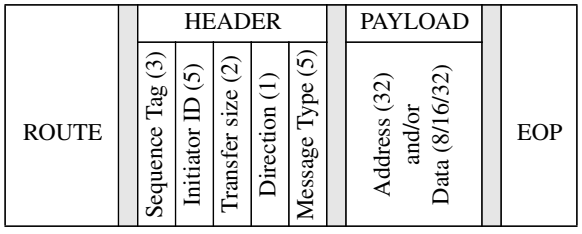

Figure 5: Packet format

Where a connection in a network uses a gang of links as described above, the route and end-of-packet symbols are replicated on all of the links in the gang, and the packet body (the messages) is spread across the links in the gang.

4. Smartcard chip integration using CHAIN

Integration of the components described in section 2 and the additional test/debug support described later in section 5 required a CHAIN network supporting a total of 3 initiators, 6 targets and the bare port. The construction of the fabric is described below.

4.1. The network fabric

Since the required performance of the smartcard system was only 10 MIPS, there was no need for a complex, high-performance network in this system. This allowed exploitation of the low-end of CHAIN's scalability in three areas:

- Link-length - low performance requirements allowed links of around 2mm to be used giving around 0.75Gb/s throughput per link.
- Gang width - a gang of two links gives an aggregate throughput of 1.5Gb/s, which gives an upper bound on message throughput of about 31 million messages/second (for a 48-bit read-command/read response message) and a fabric-imposed latency of 32ns over-and-above the route-setup time.
- Topology - minimal concurrency was required since the two processors would not (in normal circumstances) be simultaneously enabled, and did not support memory-system pipelining. A simple concentrate/expand topology, analogous to a shared bus, was adequate.

This meant that the fabric was composed of connections formed from two outgoing links to carry the commands, and two returning links to carry the responses with the switching components connected as shown in figure 6.

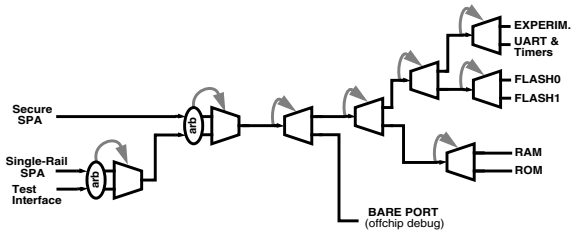

Figure 6: Network fabric composition

4.2. Network interface adapters

All connections between client devices and the CHAIN fabric are made by either an initiator or a target network interface. These are responsible for a number of tasks including:

- performing the serial<->parallel conversion necessary to transmit the 16-bit message headers, and 32-bit addresses, read-data and write-data payloads across the narrow connections provided by the gangs of two CHAINlinks.
- determining the routing setup symbols required to steer a packet from the sender to the receiver and transmitting these at the start of the packet.
- constructing and processing the 16-bit message header
- transmitting the end-of-packet symbol after the last message to clear-down the connection.

The internal structure of an initiator network interface adapter is shown in figure 7.

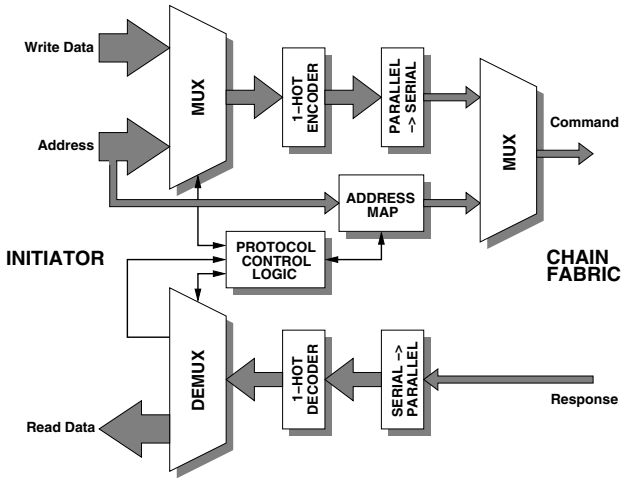

Figure 7: Initiator Network Interface Adapter

In a more demanding application, the network interface adapters would also manage the end-to-end flow control to restrict the number of outstanding transactions permissible by an initiator and would provide any packet reordering required to ensure the correct operation of the system.

The interfaces provided by the network interface adapters in this system all used the dual-rail data design style. The secure SPA processor, peripherals and test interface could use this interface directly, but small hand-designed protocol adapter wrappers were required for devices such as the single-rail SPA and the ROM and RAM memories.

4.3. Component placement

The final layout of the chip is shown in figure 8 with the CHAINlinks and the fabric switches highlighted. The only constraint on the floorplanning stage was to roughly balance the lengths of the CHAINlinks, keeping the long links below the 2mm length, which was easy to achieve.

4.4. Design validation and timing closure

The system was extensively simulated using extracted transistor/capacitance netlists of the two processors with functional models for the memories and basic, non-extracted transistor netlists (no R/C values) for the CHAIN network in order to validate the functionality and correct

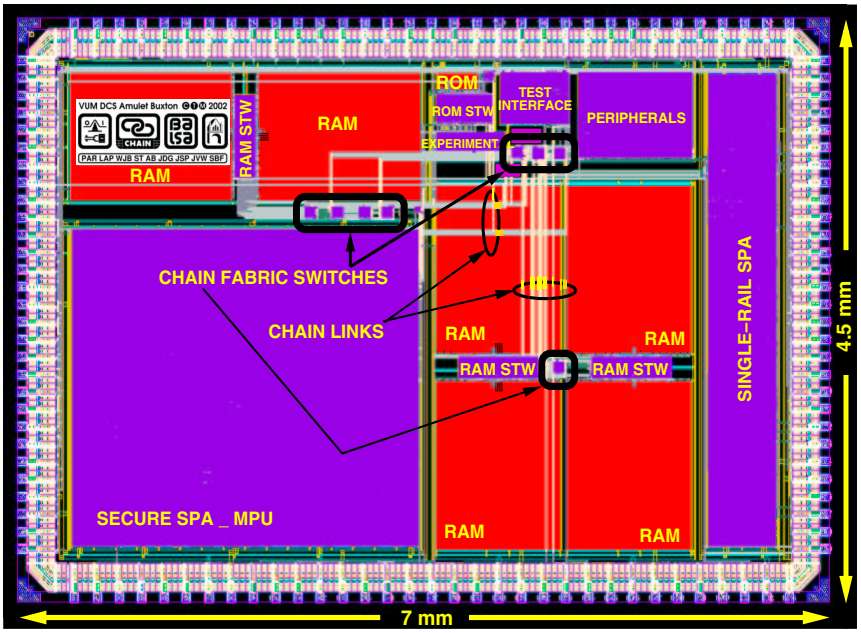

Figure 8: The SPA smartcard chip

timing behaviour of the computational blocks.

The CHAIN network fabric was never extracted prior to device fabrication, since with the self-timed design guarantees correct-by-construction timing behaviour and the network offered a significant margin over the required system performance level.

With the fine-grained control over the network performance provided by the ability to choose abitrary link-lengths and gang-widths as described earlier, it is expected that system-level interconnect timing validation should be unnecessary in the majority of systems built around CHAIN where sufficient margins can be designed in, although in extremely high performance systems, some simulation might still be desirable to validate the network performance.

4.5. Tool flow

Most of the components of the smartcard, including the two processors, the memory protection unit and the peripherals, were synthesised using the Balsa asynchronous synthesis system. Commodity memories were used and a few very small interfaces and the on-chip interconnect were hand-designed.

Balsa compiles the circuit description and produces a Verilog standard-cell netlist. The hand-designed parts were then incorporated into the same structural netlist which was then fed through standard commercial tools (Cadence, SiliconEnsemble, Arcadia, Nanosim, etc.) for gate-level simulation, static timing analysis, layout and routing, extraction and transistor-level simulation.

A conventional standard cell library was used for this work, but it was augmented with a few special self-timed cells (a Mutex [9] and Muller C-elements). Whilst such cells can be constructed using conventional synchronous cells, they were added as custom primitives to improve performance and reduce area costs.

5. Test components in the smartcard chip

Since the smartcard system does not need an external memory interface, its only general-purpose communication route with the outside is through the dedicated UART. This severely limits the controllability and observability of the system. Although this is desirable in a production smartcard for security reasons, it is not convenient in an experimental chip. To aid the testing and debugging of the system and the CHAIN network, the following interfaces were included in the design.

5.1. External Test Interface

A synchronous test interface was included on the chip to provide external access for applying test patterns and initialising the contents of the on-chip memories. The external interface is similar to the one which was successfully employed on Amulet3i [10] and operates fully synchronously as this style is best suited to current automated chip test machinery.

The test interface consists of a 32-bit bidirectional bus, a 3-bit control-bus and a clock input and supports reading and writing at arbitrary addresses in the on-chip address

space through its own dedicated initiator interface to the CHAIN fabric. This allows the memories and peripherals to be tested directly, and the processors to be tested indirectly by loading code into the memory, running the code and then reading back a checksum from the memory.

5.2. Bare fabric port

To allow further debugging support and memory expansion, a bare port was added to the network fabric. This bare port brings the underlying fabric to external pins, without an initiator or target interface. As the fabric uses 2 ganged CHAINlinks and we wanted the bare port to bring single CHAINlinks off chip, a small controller was implemented using a multiplexer/demultiplexer with some buffering. The initiator interfaces route-mapping tables (which generate the route-symbols at the head of a packet) were modified so that all addresses not corresponding to a memory-mapped peripheral would be directed to the bare port.

The symmetry between a read-command and a read-response (route, 16 bit header, 32 bit address or data, eop) allows the bare port to be used in a loop-back configuration. The self-timed nature of the command and response links and their identical structure (5 forward and 1 acknowledge signals) allows the loop-back to consist of simply wiring the pins together.

5.3. Other test features

Two external pins set the bootstrap configuration of the chip. The levels on these pins control which of the four on-chip memories is used for the first instruction fetch after a reset. This allows us to boot the processor from ROM during testing and from non-volatile memory when in service.

Support was also added to the initiator interfaces to allow an external pin to override the route-mapping tables so that ALL network traffic is routed to the bare port, thus effectively giving a direct connection from the processor to the outside world. The override can be removed part way through a transaction causing the route-mapping tables to revert to their normal operation for subsequent transactions.

6. Debugging the silicon using CHAIN

The chips were fabricated through MOSIS on a TSMC 0.18µm 6 layer metal CMOS process which runs the core cells at 1.8V and the pad ring at 3.3V. No production testing was performed on the chips.

As expected, all the major self-timed components on the chip operated correctly. Unfortunately, the received parts had a small number of faults in the hand-designed interfaces to the synchronous components which were later mostly identified as design errors as described in the following subsections.

6.1. Test interface

The test interface was supposed to allow reading to and writing from arbitrary locations in the memory address space using a simple synchronous interface. In reality, it was impossible to use for any practical purpose since on read operations it always returned the all-ones pattern. This was a serious problem since its main purpose was to allow testing and initialisation of the on-chip memories.

Further testing of this interface, by writing-to and reading-from the CHAIN bare port showed that in fact write and read operations were being performed correctly, but there was an error in the logic controlling the timing of the tristate drivers on the bidirectional off-chip pins meaning that they were enabled during the wrong phase of the interface's clock cycle.

6.2. RAM self-timed wrapper

The test interface should have been the simplest method for initialising the memories in order to bootstrap the processor, but it was not the only avenue available.

As described in section 5.3, the memory map also allowed the first instruction to be fetched from the ROM, which contained code to bootstrap the system using the UART to load code into the memory from off-chip. However, this did not work either, because (as was later identified) the memory self-timed wrapper had a delay-fault which had not been apparent in simulation.

The final, but least attractive option available was to use the CHAIN bare port with the route-mapping table override active so that the processor would ignore its on-chip memory system and run all code from off-chip. This of course has performance implications.

Using this approach, the basic functionality of the processor was proven. Using the route-mapping table override allowed the processor to be bootstrapped into running code from the unmapped region of the address space (which of course is always directed to the bare port as described in section 5.2) and then the removal of the override allowed the on-chip memory subsystem to be probed by the processor. The failure of the RAM self-timed wrapper then became apparent.

6.3. External timing reference signal

An internal signal was brought off-chip to act as a timing reference to aid the security analysis. This signal was generated internally by a coprocessor (sync copro in figure 1) connected to one of the SPA processor cores but, unfortunately, the pulse width was too narrow and the pulse degraded in passing through the pad buffer and was not visible to the outside world. This posed the problem of how to

provide a suitable timing reference to aid the security analysis.

The solution adopted was to use the bare port loop-back feature, described in section 5.2, without any additional hardware. This results in a single pulse on one of the interface signals which is an acceptable alternative to the coprocessor pulse.

6.4. Measurements and results

Some of the above problems were worked-around, whilst others had to be fixed using a focused ion beam, but eventually a fully functioning chip was obtained which was used to perform the security analysis reported elsewhere [11] and to examine the on-chip CHAIN networks behaviour.

Initial performance measurements place the secure SPA at 6 MIPS and the single-rail SPA at 11 MIPS. The performance of the secure version is not as high as had been expected. Several sources of inefficiency have already been identified in the SPA architecture, the Balsa code and the implementation of the secure back-end but it is too early to predict what performance improvement may be achieved by correcting these problems.

Measuring the performance of the high-speed network fabric embedded in the low performance smartcard chip has proven more difficult than first anticipated.

To aid debugging of the network fabric and interface adapters, a small experimental target was included in the design which could introduce sequences of transaction deferrals and transaction abortions.

By using the solution described in section 6.3 to measure the time difference between a set of CHAIN transactions that were deferred compared to the same set of transactions without any deferral, it was found that the fabric did in fact operate at just over the nominal 500MHz cycle-rate predicted by simulations, illustrated in figure 4, for the longest link on the chip which is about 1.5mm in length.

7. Conclusions

This paper has shown how a self-timed network-on-chip has been successfully deployed in a simple system-on-chip design. The self-timed nature of the interconnect meant that there was no need to perform post-layout timing validation of the network after the final stages of system integration.

A further advantage from the self-timed operation was the ability to use only a very thin network fabric operating at a much higher cycle rate than the processing and memory subsystems that were to be connected, without having to provide (and timing validate) a high-speed clock. The fabric's bare port proved so useful in debugging the chip that we are currently enhancing its capabilities and performance for use in future designs.

The fabricated chip has demonstrated how a self-timed network-on-chip can perform all of the functions offered by a more conventional system bus such as allowing test access and debugging of the IP blocks within the system, whilst giving the significant advantages of self-timed operation, rapid timing closure and greater design flexibility/ scalable performance.

8. Acknowledgements

Funding for the work on CHAIN was provided by Theseus Logic Inc. and the EPSRC (GR/R533340/1). The development and fabrication costs of the SPA smartcard chip were funded by the EU G3CARD project (IST-1999-13515).

9. References

[1] ARM Ltd. AMBA, Advanced Microcontroller Bus Architecture Specification, Rev 2.0, May 1999.

[2] IBM Corporation, CoreConnect Bus Architecture, product brief. URL: http://www.chips.ibm.com/news/1999/990923/pdf/corecon128_pb.pdf

[3] L.A. Plana, P.A. Riocreux, W.J. Bainbridge, A. Bardsley, S. Temple, J.D. Garside and Z.C. Yu, "SPA - a Secure Amulet Core for Smartcard Applications," *Microprocessors and Microsystems*, vol. 27, no. 9, Oct. 2003, pp. 431-446.

[4] D. Jaggar and D. Seal, "ARM Architecture Reference Manual", Addison Wesley Publishing Company, 2000.

[5] International Standards Organisation, "Identification Cards - Integrated Circuit(s) with Contacts - Part 3: Electronic Signals and Transmission Protocols," ISO/IEC 7816-3:1997, 1997.

[6] D. Edwards and A. Bardsley, "Balsa: An Asynchronous Hardware Synthesis Language," *The Computer Journal*, vol. 45, no. 1, Jan. 2002, pp. 12-18.

[7] J. Bainbridge and S. Furber, "Chain: a Delay-Insensitive Chip Area Interconnect," *IEEE Micro*, vol. 22, no. 5, Sep./ Oct. 2002, pp. 16-23.

[8] M. Shams, J.C. Ebergen and M.I. Elmasry, "Modelling and comparing CMOS implementations of the C-element," *IEEE Transactions on VLSI Systems*, vol. 6, no. 4, Dec. 1998, pp. 563-567.

[9] C.L. Seitz, *System timing*, in C.A. Mead and L.A. Conway, editors, *Introduction to VLSI Systems*, chapter 7, Addison-Wesley, 1980.

[10] S.B. Furber, D.A. Edwards and J.D. Garside, "AMULET3: A100 MIPS Asynchronous Embedded Processor," *Proc. Int'l Conf.Computer Design* (ICCD), IEEE CS Press, Los Alamitos, Calif., 2000, pp. 329-334.

[11] Z.C. Yu, S.B. Furber and L.A. Plana, "An Investigation into the Security of Self-Timed Circuits," Proc. 9th IEEE Int'l Symp. Asynchronous Circuits and Systems (Async), IEEE CS Press, Los Alamitos, Calif., 2003.