The Story of the Amulet: A Brief History of Asynchronous Events in Manchester

Doug Edwards, Will Toms, Steve Temple, Luis Plana, Jim Garside and Steve Furber

University of Manchester

Abstract. This presents an overview of the significant achievements in Asynchronous Logic over the last quarter of a century at the University of Manchester. Including the Amulet (and subsequent) asynchronous Microprocessors, the synthesis tools Balsa and Teak, the MARBLE and Chain on-chip Interconnects, together with research arising from external collaborations with the University of Newcastle.

1 Pre-history - an Overview by Steve Furber

The Amulet story begins in 1989. Through the 1980s I (SBF) worked at Acorn Computers Ltd in Cambridge, UK, where I was a designer of the BBC Microcomputer and the ARM (then "Acorn RISC Machine") 32-bit RISC microprocessor. We didn't publish much, but one paper we published was on the ARM3 - the first ARM with an on-chip cache - at VLSI'89 in Munich [7]. At that conference I was struck by a paper [14] presented by Craig Mudge, then CEO of Austek Microsystems Ltd based in Adelaide. This was my first exposure to asynchronous design, and of course Mudge's paper cited Ivan Sutherland's Turing Award paper [19] which really sold the concept to me.

Over the following year I doodled some ideas on the possible design of an asynchronous version of the ARM based on Sutherland's micropipelines approach. This was around the time the EU launched the OMI - Open Microprocessor systems Initiative - which was in part stimulated by input from Acorn. When I accepted the ICL Chair at Manchester, to which I moved on 1st August 1990, Acorn was generous in allowing me to take with me to Manchester a part of an early OMI project that I had been involved in developing at Acorn - the OMI-MAP project. OMI-MAP provided the early funding that got the Amulet (asynchronous ARM) research off the ground at Manchester. The University allocated a lectureship appointment linked to my chair, to which Jim Garside was appointed, and OMI-MAP funded the appointment of Nigel Paver and Paul Day as post-docs. In addition, existing academic staff joined in - Doug Edwards, Linda Brackenbury and Viv Woods, bringing in additional perspectives and new funding, all of which enabled the Amulet group to grow to critical mass.

Why "Amulet"? Well, in searching for a group or project name you have to start somewhere, and writing down keywords on a whiteboard is one way to start. "Asynchronous", "Manchester University","Low-Energy Technology"?

The name stuck and worked well, even though the acronym expansion did not see much light-of-day!

2 Amulet Processor Series

Before engineers had learned the "correct way" to build digital circuits - by inventing a global clock to slow everything down - there was considerable experimentation with different design styles. However to attempt anything complex in a clockless circuit was clearly a bad idea. Was it even possible to challenge the decades of synchronous development with such a radical alternative?

Few people had tried asynchronous VLSI, largely because the two terms are from different eras; asynchronous digital circuits largely predated VLSI. However, conventional devices were exhibiting some worrying trends, particularly increasing power consumption in parallel with the rise of mobile computing devices and shedding the clock seemed like a way to lose one of the major drains on a battery, particularly as the clock waggled continuously but did no actual computing.

Requiring a demonstrator, a microprocessor seemed an appropriate size. Regarding an architecture, a custom ISA was already being addressed at Caltech [13] so cloning a commercial ISA seemed to pose some additional - and well-defined - challenges. Respecting Steve Furber's background, and arguably influenced by already being a low-power champion - the demonstrator ended up as an ARM.

Thus was born Amulet 1, an attempt to produce an ARM7 including everything but the clock. With respect to the literature, particularly Sutherland's 'Micropipelines' paper [19] this was done using transition (two-phase) signalling and lots of "exotic" standard cells such as a variety of Muller C-elements [15]. Time elapsed and the the device appeared, made largely 'by hand' on a 1 µm 2LM process. Happily it was functional and could run standard ARM code with various asynchronous features such as varying execution speed as the supply voltage was changed. Rather less happily it did this at about half the speed of a contemporary ARM7, despite managing a 'CPI' figure of 0.0. On the plus side, energetically it was at least comparable with the synchronous circuit; as a minus the transition signalling proved extremely painful to interface to.

Somewhat undaunted, Amulet 2 followed, hopefully incorporating the lessons learnt. This involved going "four phase" to simplify the circuit design and incorporating more features - notably a cache memory (operating asynchronously, of course) on chip. This achieved its objectives in that when the chip arrived it worked and was a lot easier to use. It was also somewhat faster; unfortunately so were the contemporary synchronous ARMs though which meant it was still about half their speed. However there was another interesting observation made which was that the lack of temporal coherence reduced the electromagnetic noise emitted by the system.

It was this last characteristic which was a tipping point in having a third attempt at the architecture, named, unsurprisingly, Amulet3. This was done in

parallel with Hagenuk GmbH for integration onto Draco - a Digital Radio Controller. This time only half the chip (the Manchester half) was clockless but this included the processor, memories, DMAC etc. all communicating across MARBLE [2], an asynchronous bus. This was now to compete with the ARM9 and more features - such as branch prediction and out-of-order, parallel execution - were incorporated into the microarchitecture. The goal was to achieve a commercially available system before the year 2000. The chip was manufactured in time; unfortunately also in time for Hagenuk to go out of this business. However the technical design was vindicated in that this would run at effectively the same speed as an ARM9 manufactured on the same process. Honours on energy use was about even; the EMI was startlingly low.

Amulet 4 may have addressed more sophisticated architectural issues such as superscalar execution ... but it never happened. In the synchronous world new techniques - such as gating the annoying, power-consuming clock - were being adopted and it was becoming apparent that the potential advantages of future asynchronous processors were unlikely to be great enough to cause a major switch in the industry. However the chips did demonstrate that most, perhaps all, synchronous microarchitecture can be duplicated competitively without a clock.

3 SPA - A Synthesised Amulet

The Amulet processors followed the design style used in the early, synchronous ARM hard macrocells - full-custom datapaths with standard cell control blocks. A different approach was required for the next collaborative project, which investigated the ability of asynchronous systems to offer improved security through increased resistance to non-invasive attacks on smartcards, such as power and timing analysis. A new processor, based on a more secure asynchronous technology, was needed and there was no time to design it in the rather laborious, full-custom style used previously. Balsa had proved its value in the Amulet3-based DRACO chip and presented the group with the opportunity to produce a synthesised, asynchronous, ARM-compatible core, named SPA [16]. In fact, a complete asynchronous smartcard System-on-Chip, shown in figure 1, was synthesised using Balsa. The only clock used in the system is the smartcard interface clock, driven by the smartcard reader. All the components of the system-on-chip were connected through CHAIN, the on-chip interconnect technology also developed at Manchester. Balsa was used to describe and synthesise the system-on-chip, which incorporated synchronous memories with asynchronous wrappers. Cadence CAD tools were used to implement the chip in TSMC 180nm technology. The chip, shown in figure 2, occupied an area of approximately 33 mm^2. Prototypes were received from the manufacturer in October 2002 and, after a connection in the memory wrapper was repaired using FIB technology, were fully functional. SPA required a completely different design approach from the Amulets. The strategy to make SPA robust against power and timing attacks was to make sure that all operations consumed the same energy and took the same

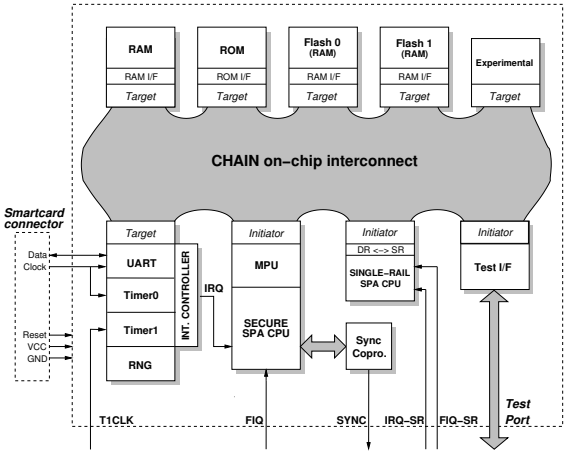

Fig. 1. SPA Block Diagram

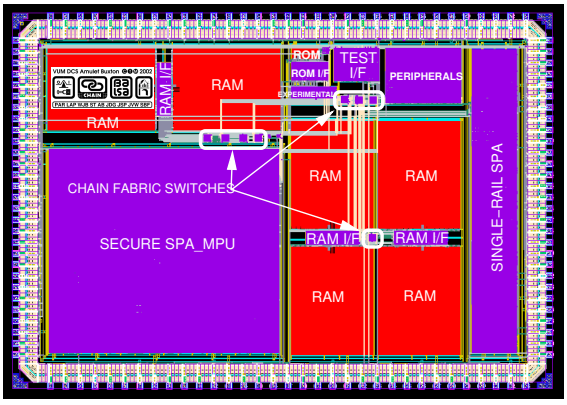

Fig. 2. SPA Chip Plot

time, irrespective of the actual data. This went against a basic asynchronous designer's strategy to take advantage of average case performance. The only sensible way to achieve this was to design the processor to operate in worst-case time and energy consumption. SPA was a success from the point of view of security but was distinguished with the unenviable accolade of being the slowest ARM ever.

4 Latch Controllers

Amulet1 was an implementation of the ARM based pretty faithfully on Sutherland's micropipelines style. However, we found the two-phase control pretty messy and slow for complex control structures, so for Amulet2 we decided to try four-phase micropipelines - a very similar bundled-data approach but with return-to-zero (RTZ) signalling. After some early exploratory work [6] we decided that the number of options in terms of the interleaving of the two handshakes was too complex to do entirely by hand, so with a little help from Alex we adopted an STG approach to devise a range of solutions with performance/complexity trade-offs. The paper published in IEEE Trans.VLSI [8] remains Steve Furber's most cited paper of all time, with just under 300 citations to date pretty evenly spread over the last 20 years! Further developments optimised the controllers for pipelines using dynamic logic [10]. Steve pulled a lot of this together in the unpublished (though still cited!) "A small compendium of 4-phase micropipeline latch control circuits", which has kept Graham Birtwistle and Ken Stevens (constructively) amused for many years since exploring the outer limits of this space.

As our designs became more complex, the need to use formal tools to get things right increased. On Amulet3 [9] we acknowledge extensive use of Alex's Petrify tool in the design. Although a wide range of latch controllers are used in the design, the most terrifying aspect of it was the register reorder buffer. Here Jim Garside came up with an initial circuit by hand, but no-one else could understand it, so Steve used Petrify to verify Jim's design. This turned out to be non-trivial - specifying an STG with as much concurrency as Jim's brain was a very challenging and incremental process! But in the end, with a little help from Alex, we got there. Petrify proved Jim's design correct, and even managed to remove one transistor from it!

5 AntiTokens and Wagging Logic

While working in the optimisation of dual-rail self-timed logic, Charles Brej formulated the idea of "anti-tokens" [5]. To fully exploit the benefits of "early-evaluation" (or "Or-causality") anti-tokens are sent in the opposite direction to the flow of data to destroy data that is no longer required. The concept was subsequently adopted by Cortadella et al for use in Elastic Systems.

Charles also developed the "Wagging-Logic" style where logic-blocks are replicated (into slices) and each successive datum are applied to successive blocks in a round-robin fashion. This allows the return-to-zero phase of a slice to be

overlapped with the data-phase of the succeeding slice. By extending the concept to the units of a Microprocessor, data forwarding occurs between slices, reducing the interlocking penalty and maximising the throughput of independent instructions. This design style cumulated in the Utopium, an 11-way wagging 8051 microcontroller. Sceptical comments from reviewers were etched into the top layer metal (figure 3)!

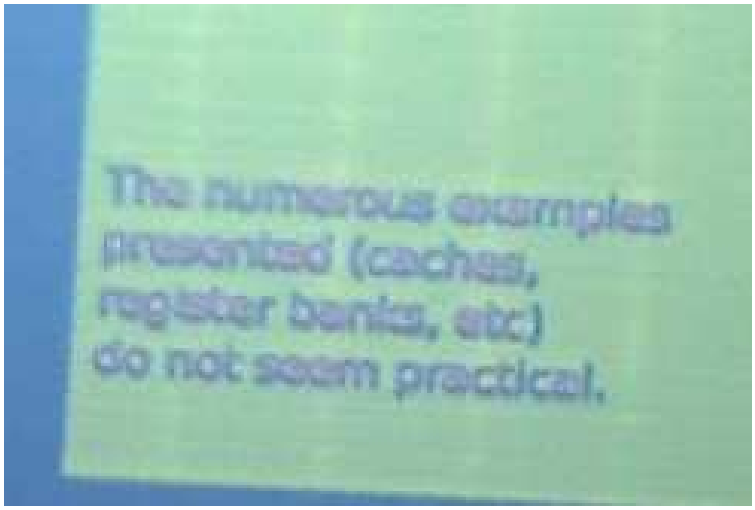

Fig. 3. Utopium Die Plot

6 Balsa and Teak

As part of the EU funded project, EXACT, we were introduced to Tangram, developed by Philips Research Labs, wihich was a CSP-based tool using handshake circuits to synthesise complex asynchronous hardware. it was clear that Tangram was likely to be extremely useful for the design of the systems that the Amulet group might be interested in building. However, it had a number of disadvantages: the language was unfamiliar to most engineers; more importantly, it was proprietary software and was not possible to experiment with language extensions and optimisations and it was not clear if Manchester would be able to use it after the end of EXACT project and. Doug Edwards, who led the Manchester effort on EXACT, persuaded Andrew Bardsley, then a final year student, to become interested in the idea. Andrew produced a prototype for his prize-winning project and then produced a complete working system, Balsa [4], for his M.Phil.

Balsa was used to design the DMA controller for Amulet3. This was a matter of necessity: for various reasons, it would not have been possible for a hand-

designed controller to meet the tape-out deadline; the design that Andrew produced in short order was a convincing proof of the effectiveness and reliability of the tool. Balsa was then used to successfully design a complete processor - SPA by an engineer, Peter Riocreux, who had no previous experience of the Balsa paradigm. SPA has a reputation of being slow, but this is in part due to the design requirements of the processor.

Although there were good reasons for SPA's performance, strenuous efforts were made by Andrew and Luis Tarazona to improve the speed of Balsa generated circuits resulting in an improvement of more than an order of magnitude. Nevertheless, performance could not match that achieved by a conventional synchronous implementation. Attention was directed towards generating data-driven circuits rather than the control-driven circuits of Balsa. Sam Taylor's system, based on Balsa, gave promising results, but unfortunately existing circuit descriptions could not be automatically translated into his new language.

Andrew then began developing Teak which is capable of transforming the timeless concurrent specifications in the CSP-based Balsa language into a data-driven network. The Teak generated dataflows enjoy a set of architectural properties in communication and computation including slack elasticity, distributed control and data-driven behaviour which pave the way for further optimisations such as retiming and re-synthesis. These all make Teak a practical EDA framework in the asynchronous domain suitable for exploring fine-grained elasticity toward tackling the energy issue in large-scale SoCs

Recently eTeak has been introduced to enable the synchronous designers to exploit the powerful properties of the Teak networks including scalability. eTeak adopts a synchronous library (specifically the synchronous elastic protocol) to introduce a common timing discipline to the asynchronous dataflows of Teak [12]. This way clocked commercial EDA is employable for retiming and resynthesis in the synchronous domain. Latest explorations toward automatic GALS synthesis leverage these advantages, particularly retiming of eTeak circuits, to study the impact of the fine-grained partitioning on performance.

Balsa research has had many points of intersection with Alex Yakovlev's group at Newcastle, from STG specification of handshake circuit to collaboration on funded research projects such as GAELS, SEDATE, VERDAD. It has been used as a teaching and research tool in many institutions globally.

7 Asynchronous Interconnect

The Manchester AsynchRonous Bus for Low Energy (MARBLE) [2] was developed by John Bainbridge for use in the Amulet3H system. MARBLE was a dual-channel pipelined bus with centralised arbitration and address decoding, using an asynchronous four-phase bundled data protocol. The interfaces (figure 4) were specified using STG's and speed-independent implementations were synthesised using Newcastle University's Petrify tool, except for two signalling modules in which timing assumptions were unavoidable. John was awarded the BCS Distinguished Dissertation award for his PhD based on MARBLE.

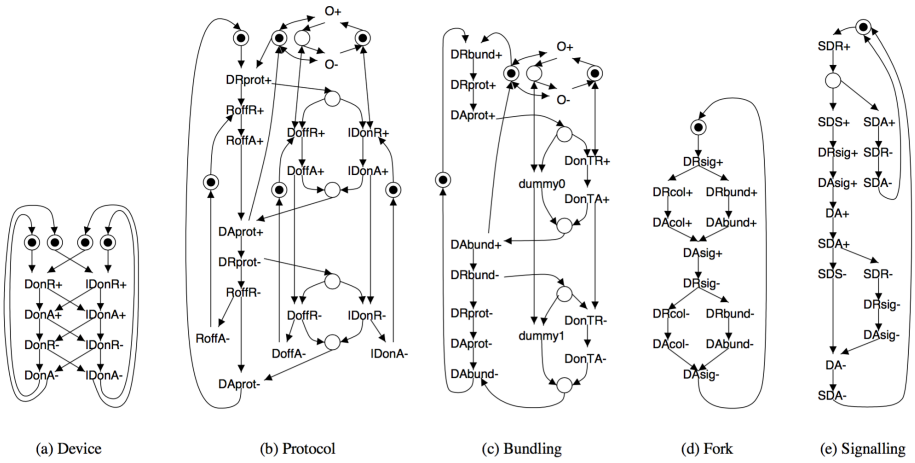

Fig. 4. MARBLE Interface STGs

To overcome the limitations of a shared bus, MARBLE was re-implemented as a packet-switched Network-on-Chip, using a Delay-Insensitive 1-of-4 data encoding [3]. This was further developed in to the CHip Area INterconnect (CHAIN) Network fabric [1] using a range of incomplete m-of-n codes designed to minimise encoding and decoding overhead. The CHAIN Network became the core of Silistix, which produced a range of AMBA and AXI compatible asynchronous network-on-chip solutions.

8 SpiNNaker

In addition to leading the research into asynchronous systems, in the late 1990s Steve Furber also became actively interested in how the brain functions. This was to lead to the SpiNNaker project [11] which received its first funding in 2006 and has continued to the present time. SpiNNaker is based around a multi-core processing chip [17] which has a novel routing system for small packets (40 or 72 bits) which represent the spikes emitted by real neurons when they fire. The chips are designed to tile together to create a massively parallel compute engine for simulating networks of spiking neurons. Currently, a 500,000 core machine has been constructed, which consumes around 40kW when running flat out. It is planned to extend this machine to 1 million cores.

It was clear from the outset that the design effort in making the SpiNNaker chip required the use of as much pre-existing IP as possible. Our Amulet cores were not readily process-portable so in collaboration with ARM we obtained processor and memory controller IP (synchronous, of course) and from Silistix, which was spun out from the group to commercialise the CHAIN system, we obtained asynchronous networking IP. This meant that we could concentrate our

efforts on the novel IP required for SpiNNaker which was based around a custom router which facilitated efficient multicast routing of the small packets used to represent neural spikes. The chip was built as a GALS system which meant that we could harden the various synchronous IP blocks and obtain timing closure on them relatively easily, while the GALS architecture meant that timing closure at the top level was much easier than a fully synchronous system of similar size.

So while SpiNNaker generally uses synchronous IP there is a significant asynchronous element to the design. The chip has two independent asynchronous networks (figure 5). The System NoC is the main system bus for all of the cores, on-chip peripherals and memories. It was generated using Silistix's tools and based around multiple 3-of-6 RTZ channels running in parallel to achieve the necessary bandwidth, which is of the order of 1 Gbyte/s. Silistix provided synchronous interface IP for this network which presented ARM AMBA interfaces (AXI, AHB and APB) to the synchronous IP blocks. The second asynchronous

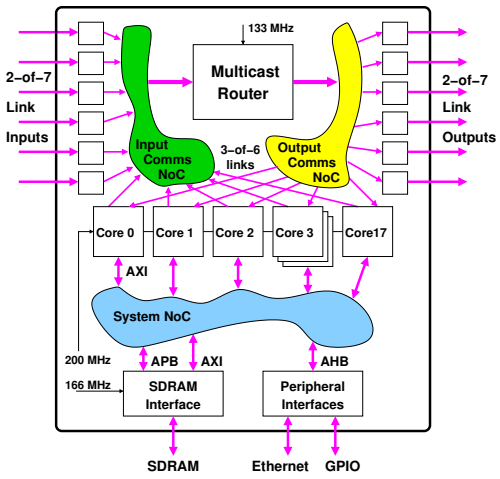

Fig. 5. SpiNNaker asynchronous NoCs

network, the Communications NoC, is used to carry spike packets around the chip both between cores and the router and between the router and the off-chip packet interfaces which transport packets between chips. The Comms. NoC links are single 3-of-6 RTZ channels as the bandwidth requirement for packets is relatively modest. To carry packets from chip to chip, we convert 3-of-6 to an asynchronous interface based on 2-of-7 NRZ signalling. This only requires 3 transitions (two data and acknowledge) to convey data as a 4-bit flit and provides a very low-power interconnect, albeit at the cost of 8 pins per channel. The bandwidth across these links is around 250 Mbit/s and significant design effort was

expended in making them resistant to glitching (and therefore deadlock-free) as they may span significant distances on a PCB [18].

Two chips were fabricated in a UMC 130nm process through Europractice. A prototype (MPW) chip in 2009 was followed by the production chip in 2011. The production chip figure 6 is approximately 10x10mm and houses 18 ARM968 processing cores each with 96 Kbytes of local RAM. The packet router is in the centre of the die and there is also an SDRAM interface block at the left hand side which interfaces to a 128 Mbyte LPDDR memory die which is housed in the same package as the SpiNNaker die. As some of the on-chip asynchronous links span considerable distances on the die, pipelined repeaters were inserted every 0.5mm to maintain throughput. The System NoC is laid out as a sea of gates surrounding the router. Standard synchronous CAD tools were used throughout the design with a minimum of manual intervention to maintain correct operation of the asynchronous components. An estimated 30 man-years went into the

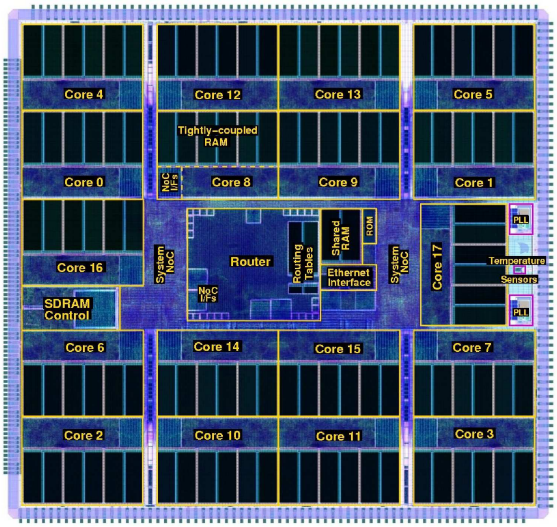

Fig. 6. SpiNNaker chip die plot

design of the SpiNNaker chip, a figure which will soon seem insignificant when the associated software effort is calculated!

References

1. Bainbridge, J., Furber, S.: Chain: a delay-insensitive chip area interconnect. IEEE Micro (5), 16–23 (2002)
2. Bainbridge, W., Furber, S.B.: Asynchronous macrocell interconnect using MARBLE. In: Advanced Research in Asynchronous Circuits and Systems, 1998. Proceedings. 1998 Fourth International Symposium on. pp. 122–132. IEEE (1998)

3. Bainbridge, W., Furber, S.B.: Delay insensitive system-on-chip interconnect using 1-of-4 data encoding. In: Asynchronus Circuits and Systems, 2001. ASYNC 2001. Seventh International Symposium on. pp. 118–126. IEEE (2001)
4. Bardsley, A., Edwards, D.: The Balsa asynchronous circuit synthesis system. In: Proceedings FDL 2000. pp. 37–44 (2000)
5. Brej, C., Garside, J.: Early output logic using anti-tokens. In: International Workshop on Logic Synthesis. pp. 302–309 (2003)
6. Day, P., Woods, J.V.: Investigation into micropipeline latch design styles. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 3(2), 264–272 (1995)
7. Furber, S., Thomas, A., Oldham, H., Howaid, D., Urquhart, J., Wilson, A.: ARM3-32b RISC processor with 4kbyte on-chip cache. In: Proceedings IFIP VLSI'89 international conference. vol. 10, pp. 35–44 (1989)
8. Furber, S.B., Day, P.: Four-phase micropipeline latch control circuits. IEEE Transactions on Very Large Scale Integration(VLSI) Systems 4(2), 247–253 (1996)
9. Furber, S.B., Edwards, D.A., Garside, J.D.: AMULET3: a 100 MIPS asynchronous embedded processor. In: Computer Design, 2000. Proceedings. 2000 International Conference on. pp. 329–334. IEEE (2000)
10. Furber, S.B., Liu, J.: Dynamic logic in four-phase micropipelines. In: Advanced Research in Asynchronous Circuits and Systems, 1996. Proceedings., Second International Symposium on. pp. 11–16. IEEE (1996)
11. Furber, S.B., Galluppi, F., Temple, S., Plana, L.A.: The SpiNNaker project. Proceedings of the IEEE 102(5), 652–665 (2014)
12. Mamaghani, M.J., Garside, J., Edwards., D.: De-elastisation: from asynchronous dataflows to synchronous circuits. In: Proceedings of the 2015 Design, Automation and Test in Europe. EDA Consortium (2015)
13. Martin, A.J.: The Design of a Delay-Insensitive Microprocessor: An Example of Circuit Synthesis by Program Transformation. In: Leeser, M., Brown, G. (eds.) Hardware Specification, Verification and Synthesis: Mathematical Aspects. vol. 408, pp. 244–259 (1989)
14. Mudge, J.C.: An Illustration of Micropipelines using Two-Dimensional Fourier Transform Architectures . In: Musgrave, G., Lauther, U. (eds.) Proceedings of VLSI 89. pp. 359–368 (1989)
15. Muller, D.E.: Asynchronous Logics and Application to Information Processing. In: Symposium on the Application of Switching Theory to Space Technology. Stanford University Press (1962)
16. Plana, L., Riocreux, P., Bainbridge, W., Bardsley, A., Garside, J., Temple, S.: SPA-a synthesisable Amulet core for smartcard applications. In: Asynchronous Circuits and Systems, 2002. Proceedings. Eighth International Symposium on. pp. 201–210. IEEE (2002)
17. Plana, L.A., Clark, D., Davidson, S., Furber, S., Garside, J., Painkras, E., Pepper, J., Temple, S., Bainbridge, J.: SpiNNaker: Design and Implementation of a GALS Multicore System-on-Chip. J. Emerg. Technol. Comput. Syst. 7(4), 17:1–17:18 (Dec 2011), http://doi.acm.org/10.1145/2043643.2043647
18. Shi, Y., Furber, S.B., Garside, J., Plana, L.A.: Fault tolerant Delay Insensitive Inter-Chip Communication. In: 2009 15th IEEE Symposium on Asynchronous Circuits and Systems. pp. 77–84. IEEE (2009)
19. Sutherland, I.E.: Micropipelines. Communications of the ACM 32(6), 720–738 (1989)