SElf-timed DATapath synthEsis (SEDATE)

Case for Support: Proposed Research

1. Summary

This is a joint proposal by leaders in asynchronous synthesis and test methodologies at the Universities of Manchester, Newcastle and Edinburgh. This project will lay the theoretical and engineering foundations for novel algorithms to be incorporated in tools for the automatic synthesis of self-timed datapaths.

2. Background

2.1 Industrial Context

Parameter variations for evolving CMOS processes go from 25% on geometry and 10% on threshold voltage for 250nm, up to 45% and 15% for 70nm respectively [16]. Variability in non-conventional processes such as strained Si is expected to be even greater. In general, variability forces industry to rely on statistical timing analysis and speed binning of chips. A hard problem resulting from variability is timing closure. Typically the following overheads are added within standard synchronous (globally clocked) design flows: 45% for worst-average case, 25% for signal integrity, 30% for variability, 10% for clock skew compensation and 20% for non-balanced stages. Moreover, measuring variability is unreliable because ring oscillators on the same die can exhibit up to 15% delay variation. In practice, many designs can work at least twice as fast as the sign-off speed.

The convergence of architectural, logical and physical design [23] requires huge computational effort from the EDA tools, which must optimise designs (on power, speed and area) under the effect of parametric faults and variation in order to achieve timing closure. Lack of precise data from emerging processes, prevents designers from rapidly adjusting their designs to exploit these new fabrication processes robustly.

Self-timed logic design methodologies, ignored until recently, are now increasingly seen by industry [17] as inevitable complements to globally clocked systems. If well supported by CAD tools, such methodologies are seen as a way of rapidly prototyping existing designs and IPs on new processes. Companies such as ARM, Boeing, Epson, FTL Systems, Intel, Infineon, MBDA, Philips and Sun are either making products containing asynchronous technology or have active research groups in the area. Start-up companies exploiting self-timed technology include Fulcrum Microsystems, Handshake Solutions, Theseus Logic, Silistix and Situs Logic.

2.2 Current research Activity

Much recent research has focused on the synthesis of self-timed control circuits where methods and tools such as Petrify [7], Minimalist [12] etc. have matured into versatile synthesis environments capable of producing circuits often out-performing manual designs of similar complexity. Unfortunately, these tools are not suitable for datapath synthesis.

Balsa [10] and Tangram [1] synthesise complete systems from behavioural specifications. However, the syntax-directed nature of the compilation means that the resulting circuits are somewhat inefficient and are not suited to high performance. BESST [18] uses Petri-net based control, but synthesises datapaths converted from synchronous synthesis tools and cannot be directly applied to pipeline structures.

Martin's work at Caltech on process decomposition using CHP, a CSP subset for hardware specification, to target high performance dynamic pipeline structures has influenced several groups. The CAST System [26] synthesises behavioural descriptions into dynamic Quasi-Delay-Insensitive pipelines. A similar approach is also adopted by Fulcrum Microsystems. CAST targets non-standard cells, which require either hand-drawn layout or a cell compiler and are not compatible with standard cell libraries. The TAST System [9] also synthesises CHP specifications into QDI circuits, however, unlike CAST the flow targets standard cells and manual functional decomposition is required.

A less radical approach, *desynchronisation* [13, 3, 19], employs synchronous design compilers to convert synthesised designs into self-timed implementations. Desynchronisation techniques suffer because of the extra complexity required to make synchronous datapaths self-timed. The structure of the desynchronised datapaths is determined by conventional synthesis tools; architectural techniques that can exploit the properties of self-timed designs and reduce the overheads, such as early-evaluation, relative-timing or ultra-fine-grained pipelining, have limited applicability.

3. Description of the Proposed Research

3.1 Project Objectives

This work builds on over 10 year's research undertaken by the applicants into asynchronous system synthesis (Balsa) [10], Petri-net based synthesis (Petrify, BESST) [7,18], test approaches for asynchronous circuits [11] and preliminary work into delay-insensitive combinatorial logic synthesis [21]. However, rather than incremental advances on previous work, we will deliver new theory, algorithms and case studies.

The focus of the proposed research will be the automated generation of self-timed datapath structures such as pipelines and low-latency combinational blocks targeted at standard cell libraries. The framework will allow differing design styles within a single flow. A designer will be able choose from a range of implementations – from those that are completely insensitive to delays within components to those which are aggressively-timed using relative timing constraints based on actual layout parameters. Design-for-test techniques and relative timing constraints will be fully exploited by incorporating them into the datapath architecture. New algorithms will be developed for self-timed datapath decomposition and implementation. The system will be more flexible (allowing logic synthesis in both multivalued and binary domains) and more automated than those based on Martin's work and will be more sympathetic to the exploitation of asynchronous architectures than approaches based on desynchronisation. The measurable objectives of the work are:

1: *Algorithms for timing-aware synthesis of self-timed de-*

signs.

2: *A methodology for testable self-timed datapaths.*

3: *A synthesis flow for self-timed data paths*.

4: *A fabricated chip to evaluate the developed techniques.*

3.2 Background

3.2.1. Timing Regimes

The execution of self-timed circuits is initiated by the arrival of their operands, thus they are tolerant to variations in propagation delay: operations wait until all data arrives. The data-driven nature of self-timed circuitry adapts well to large differences in arrival times of signals and several architectural techniques for constructing self-timed datapaths such as *bit-level pipelining* [15] or *anti-tokens* [2] exploit and even propagate such differences.

Different self-timed design methodologies are characterised by the assumptions that are made about delays within circuit components. The *unbounded-delay model* (UDM) assumes unbounded (but finite) propagation delays in circuit components. The most robust of all self-timed methodologies is the *delay-insensitive* (DI) approach, where all circuit components (including wires) adhere to the UDM. In general, this is not practical so DI components are used for communication between circuits employing weaker delay constraints. *Quasi-Delay-Insensitive* and *Speed-Independent* models assume unbounded-delays within gates but bounds on the propagation delays of wires. In deep-sub-micron technologies, bounds on wire propagation delay are not viable over large areas of silicon and so speed-independent and QDI methodologies are usually limited to small circuit areas connected by DI communications. UDM circuits require that the environment of a circuit can determine that all internal transitions have taken place – a process known as *indication.* A hierarchy of timing models may be discerned – in descending order of safety (and complexity):

strong-indication (SI) circuits require all signal transitions to be indicated explicitly, giving limited scope for optimisations as each gate output must indicate transitions on all of its inputs. Recent work at MU [21] has developed synthesis techniques for this model.

weak-indication (WI) circuits allow transitions to be acknowledged implicitly. Each transition indicates only a subset of its input transitions and a set of transitions collectively indicates all of its input transitions. Weak-Indication allows variations in the logical complexity of Combinational-Logic (CL) blocks to be exploited using techniques such as *early-evaluation*, where certain "fast" outputs indicate the minimum possible set of transitions to reduce their latency and "slower" outputs indicate the remaining outputs. General synthesis algorithms for this model have not yet been developed although theoretical models based on Petri nets are already available at NU [25].

relative-timed (RT) circuits [20] exploit known ordering constraints between signals to reduce the size and latency of CL blocks. By applying ordering constraints between signals, the complexity of circuits is reduced by limiting the number of possible signal orderings that need to be considered in an implementation. The ordering relations do not put absolute bounds on the delay of signal transitions and so, crucially, circuits employing relative-timing still adhere to the UDM. Many existing systems employ simple models[8], such as unit gate-delays, to generate ordering constraints. In current CMOS technologies such models are unreliable; post-layout verification is required to ensure that all constraints are met.

monotonic circuits [6] are hazard free circuits that assume the delay of each CL block is less than the cycle time of the circuit within its environment. Methods for the synthesis of dual-rail monotonic circuits have been developed: the work proposed here will allow arbitrary encoded CL circuits to be synthesised.

3.2.2. Encoding

In order to implement self-timed datapaths, validity information must be encoded within the data. Unordered codes [24], where no code word is contained within any other, allow the receiver of data to determine the arrival of valid data unambiguously. There are many unordered codes, however most existing approaches adopt simple codes, such as dual-rail or 1-of-4 codes, allowing easy mapping between conventional binary functions. It has been shown [22] that conventional state-encoding techniques may be employed to generate unordered codes; such techniques have not been employed within synthesis flows due to the lack of effective tools for synthesis of self-timed combinational logic.

4. Work Plan and Methodology

A complete synthesis flow for self-timed datapath circuits will be created that removes the need for conventional CAD synthesis tools and allows datapath architectures to be created that can exploit the properties of self-timed circuits. Many of the architectural techniques used for self-timed pipelines are applicable to all self-timed methodologies. The proposed synthesis flow will encompass existing self-timed methodologies allowing them to be applied across a single design. An evolutionary approach to self-timed synthesis can then be adopted, where designs are initially synthesised using robust delay-models such as QDI. Once the circuit is synthesised, analysis may be undertaken to determine areas (such as critical paths) where timing constraints may be applied to increase the overall performance of the circuit

To achieve the project objectives, the research is split into five work-packages, each of which has several sub-tasks.

WP 1: Datapath Decomposition

In this work-package, techniques for specifying the high level data structure of the datapath will be developed. The initial phase of the synthesis procedure will decompose the behavioural description into a set of smaller inter-operating processes. The behaviour of the datapath and the relationship between the processes will be modelled by Petri-nets where places between events represent data-buses and tokens data-values. Because functional decomposition abstracts binary-values to tokens and events to value changes, it can be applied regardless of the self-timed design methodology. At the same time, Petri-net models with OR causality [25] will allow us to capture early-evaluation techniques at a relatively high level of abstraction.

Once the circuit is decomposed, each block can be analysed. With a wide variety of logic synthesis methods available, the choice between them can be made on the basis of this structural analysis – for example, early-evaluation blocks may

be used to generate relative-timing information to simplify circuits, but in critical areas where the cycle-time is short, more robust methodologies may be required.

To ensure that pipeline stages are balanced, it is possible [15] to analyse the circuit at a high level to insert or remove latches or to move logic through latches (slack matching). A tool will be produced to automate this task.

The output of this stage will be a synthesised communication architecture using a range of pipeline templates and a set of combinational logic block *macros*, to be encoded and synthesised by software designed and implemented in WP 2.

Task 1.1: Functional Decomposition

Aim: to develop techniques for decomposing RTL descriptions into a set of communicating processes.

Method: the datapath will be modelled using petri-nets allowing the causal relationship between processes to be explored.

Deliverable: algorithms for datapath modelling and decomposition (D.1.1).

Risks: *Medium* – efficient decompositions and modelling of process parameters may be more complex than anticipated.

Task 1.2: Architecture Generation

Aim: to perform high-level structural analysis of pipeline structures and to assign appropriate self-timed protocols to individual processes. Processes will be decomposed further into synthesisable macros and communication architectures.

Method: to use performance analysis to determine critical parts of the circuit to choose self-timed methodologies for processes.

Deliverables: software to control the synthesis process based on performance analysis (D1.2.1).

Risks: *High* – requires a deep understanding of the complex behaviour of datapath architectures and self-timed methodologies. Input to this task is split across several sites.

Task 1.3: Structural Synthesis

Aim: to synthesise the communication architecture between combinational logic processes.

Method: a range of pipeline templates, including anti-token and NCL-X style architectures will be employed. Techniques such slack-matching will be used to increase performance.

Deliverables: a synthesis tool for pipeline templates including automation of slack-matching techniques (D1.3.1).

Risks: *Low* – pipeline templates contain little data-dependent complexity. Slack-matching techniques are well documented, no public asynchronous tools have been released to evaluate their effectiveness, but tools do exist for clocked systems.

WP 2: Datapath Synthesis

This work-package builds on preliminary work at Manchester [21] into delay-insensitive datapath compilation. In order to take advantage of different DI-codes, processes are initially specified using multi-valued variables and then decomposed into synthesisable macros which must be encoded. The most common methods of state encoding perform multi-valued symbolic minimisation on state-functions to increase the efficiency of the encoding technique. It is therefore preferable to incorporate some form of multi-valued synthesis within the design-flow. As multi-valued synthesis takes place before encoding and so does not deal with individual binary bits, the constraints required to ensure full indication are much less severe than for conventional binary synthesis, thus existing multi-valued synthesis systems such as MV-SIS [4] may be used with little or no modification. The encoding techniques however have to be modified to produce DI-codes.

A range of different combinational-logic synthesis tools are required by the system in order to implement different self-timed methodologies. There are very few combinational logic synthesis systems for self-timed methodologies and so this is an important area of new research. For each methodology there are several different techniques that may be employed to optimise circuits, such as layout aware synthesis, timing driven synthesis and synthesis for test, which are of particular importance in the self-timed domain due to the complexities of testing and verifying timing constraints.

Task 2.1: Multi-Value Logic Synthesis

Aim: to encode macros in a DI-code using modified synchronous multi-level logic synthesis and state-encoding algorithms.

Method: existing multi-value logic synthesis (MVSIS [4]) will be used where possible; new state-encoding techniques, based on existing state encoding technology developed.

Deliverable: State encoding software for DI codes (D2.1.1).

Risks: *Medium* – datapath architectures may be too complex to apply efficient custom encodings. If modifications to multi-level synthesis are required, much time will be needed to become familiar with the algorithms employed.

Task 2.2: Binary Logic Synthesis

Aim: to synthesise encoded macros in a variety of different self-timed methodologies, with varying properties, such as synthesis for testability.

Method: existing algorithms will be modified to suit self-timed methodologies as in [21]. Algorithms to determine test coverage, or layout models to provide better heuristics for synthesis procedure will be developed.

Deliverable stand-alone software tools for synthesis of self-timed circuits – WI synthesis (D2.2.1), relative-timing based synthesis (D2.2.2) and monotonic synthesis (2.2.3). SI synthesis has already been completed [21].

Risks: *High* – synthesis algorithms may be complex or inefficient particularly when implementing synthesis for testability techniques, or synthesis with relative-timing constraints

Task 2.3: Technology Mapping

Aim: to map technology-independent circuits into defined cell libraries, for various different self-timed methodologies.

Method: modify existing algorithms to suit self-timed methodologies.

Deliverables: stand-alone technology mapping tools – SI mapping (D2.3.1), WI mapping (D2.3.2), relative-timing based mapping (D2.3.3) and monotonic mapping (2.3.4) (Ph.D. topic).

Risks: *Medium* – Should the Ph.D. not progress, logic synthesis is technology independent allowing less efficient gate-mapping to be used.

WP 3: Datapath Analysis

This work package relates the physical aspects of VLSI design to the synthesis procedure. In deep sub micron technologies, interconnect delay has a great impact on system performance and interconnect performance estimation models (IPEM) [5] based on actual layout parameters must be integrated into the highest level of synthesis. In the self-timed datapath synthesis flow, IPEMs will be used within the architecture generation phase not only to increase the performance of datapaths but also to create relative-timing constraints and determine the appropriate self-timed methodology for each CL block.

Once designs are synthesised, post-layout verification must be undertaken to analyse the performance and behaviour of the circuits. Timing constraints generated by the synthesis procedure must be verified. A mixture of static and dynamic timing analysis techniques will be employed; dynamic analysis, being complex and time consuming, will be used only in critical areas. Performance analysis is particularly important, because most designers are unfamiliar with self-timed methodologies. Feed-back from the behaviour of synthesised circuits is essential for educating such designers so that they may create circuit specifications that may be best exploited by self-timed implementations.

Task 3.1: In-synthesis Analysis

Aim: to develop tools to influence the synthesis procedure by analysing the physical attributes of candidate design choices.

Method: create performance estimation models based on technology parameters to generate accurate delays for architecture generation and binary synthesis procedures

Deliverables: performance estimation models (D3.1.1).

Risks: High – performance estimation models are widely used although they are complex; analysing them for self-timed synthesis techniques is difficult and has not been attempted.

Task 3.2: Post-Synthesis Analysis

Aim: to develop tools to allow timing constraints generated by the synthesis procedure to be verified. To generate tools to analyse the performance of self-timed systems to provide feedback to designers.

Method: use static and dynamic analysis techniques for performance and timing using layout information generated from existing CAD tools and simulation engines.

Deliverables: software for verification of relative-timing constraints for self-timed systems (D3.1.1). Performance analysis – a Ph.D. topic – functional decomposition models (D3.1.2), pipelines models (D3.1.3), and analysis results (D3.1.4).

Risks: Medium/High – verification of generated timing constraints is essential to ensure correct operation of circuits employing relative-timing. Accurate timing analysis is complex, particularly in deep sub micron technologies. The Ph.D. topic, performance analysis, is a difficult research area, but failure (unless total) would not be fatal to the project.

WP 4: Design for Test

Efthymiou [11] shows that sequential devices such as C-elements, often used in self-timed design, can be fully tested by applying sets of test patterns without requiring each element to be connected to a scan chain. By implementing a suitable design-for-test architecture throughout the datapath, and by implementing testing-driven logic synthesis, it may be possible to achieve full test coverage. The macros within the architecture will provide sufficient granularity for test pattern generation.

Work on test coverage and automatic test pattern generation for self-timed combinational logic is already being undertaken by Efthymiou under the EPSRC first grant scheme. The work undertaken in this proposal will not only extend these ideas and provide a suitable framework in which they may be evaluated, but also approach the problem from a top-down perspective. DfT permeates all levels of the design flow and the work will feed into Task 1.2, Task 2.2 and Task 2.3.

Task 4.1: Structural DfT

Aim: to incorporate design for testability techniques into architecture generation and structural synthesis procedures.

Method: Create a DfT architecture: break feedback loops, insert scan latches into pipeline registers, decompose large pipeline stages to increase testability

Deliverables: DfT additions to WP 1 software (D4.1.1).

Risks: *Low* – Structural DfT methods for synchronous circuits already exist. Some of these are applicable to asynchronous circuits, but no general method nor any publicly available tools exist for asynchronous circuits.

Task 4.2: Synthesis for Testability

Aim: to develop synthesis and technology mapping software using test coverage to select between possible alternatives.

Method: implement scan paths within CL blocks, using a range of metrics, such as area, performance etc, to minimise the overheads of full testability. Apply a range of scan technologies to different self-timed methodologies.

Deliverables: DfT additions to software packages of WP 2 for binary logic synthesis (D4.2.1) and technology-mapping (D4.2.2).

Risks: *High* – self-timed synthesis techniques are complex and the number of possible outcomes is severely constrained, the use of synthesis for testability techniques may not give much improvement over post synthesis techniques.

Task 4.3: Post Synthesis Testability and Test Pattern Generation

Aim: to analyse synthesised circuits in order to generate full test coverage and test-vectors.

Method: calculate test-coverage of combinational logic blocks and insert scan-latches where necessary to achieve full test coverage. Use conventional ATPG tools to generate full test patterns for synthesised circuits.

Deliverables: software to calculate test coverage of combinational logic blocks and create test patterns for synthesised circuits (D4.3.1)

Risks: *Low* – groundwork for this task is underway.

WP 5: Test Cases

The research work planned will deliver stand-alone tools. However, for comparison with existing techniques, these tools will be integrated into existing synthesis tools developed by the investigators. The synthesis flow will be evaluated on sev-

eral examples and a test chip fabricated. Candidates are:

Conventional microprocessors will allow the effectiveness of self-timed architectures for pipelined systems to be evaluated. An ALU is an ideal test case for evaluating early-evaluation and relative-timing.

DES & AES encryption algorithms. Strongly-indicating DI methodologies may increase security of systems but the performance of such systems is generally poor. Synthesising encryption blocks within a unifying self-timed framework would allow security-critical strongly-indicating components to be implemented with full testability, but also would allow the design to be partitioned so that non-security critical blocks could be implemented in more efficient methodologies.

GALS (Globally Asynchronous Locally Synchronous) Architectures. The requirements of GALS interconnects are very different from other self-timed systems: combinational logic requirements are relatively small and techniques such as early-evaluation are not readily applicable. Timing optimisations may also be more complex due to the low cycle-time of interconnect stages, thus either a robust self-timed methodology must be employed or timing assumptions must be rigorously verified.

Neural network processors are ideal tests for self-timed systems due to their inherent concurrency and non-standard (threshold logic) combinational logic functions. Research into self-timed hardware implementation of neural-networks has shown m-of-n delay-insensitive codes to be effective.

Task 5.1: Tool Integration

Aim: to integrate the research results into Balsa and Petrify.

Method: suitable APIs will be provided in stand-alone tools.

Deliverables: enhanced Balsa and Petrify systems (D5.1.1)

Risks: *Low* – we have intimate knowledge of both systems.

Task 5.2: Evaluation of Case Studies

Aim: to choose suitable demonstrators and evaluate the effectiveness of the research.

Method: existing asynchronous implementations will be examined to choose those with interesting properties as discussed above.

Deliverables: demonstrator chip (D5.2.1)

Risks: *Medium* – a number of self-timed examples already exist, e.g. Balsa descriptions of ARM v5 and MIPS. Chip manufacture is risky, but the groups are all experienced in successful chip fabrication. Simulation is a fall-back position.

5. Project Management

5.1 Risk Management

The risks involved in each task have been considered. Many of the tasks outlined in the proposal represent fundamental research into synthesis of self-timed circuits; they will be useful as stand-alone tools to other researchers and industry alike. Even if the initial ambitious goal of a complete system is not fully realised within the project, the aims of novel research in self-timed datapath synthesis will still be achieved.

5.2 Project Co-ordination

The principal investigators know each other well and have collaborated previously. The interdependencies between the task have been considered carefully and tasks allocated to minimise risks. A supervisory board of PIs and industrial collaborators (FTL/Silistix/Cadence/Intel) will be established and will meet quarterly with regular extended evaluation visits. Use will be made of video conferencing technology for weekly progress checks.

6. Relevance to Beneficiaries & Timeliness

The research will benefit companies involved in the design and manufacture of microelectronic systems based on high-end semiconductor technologies, in particular deep-submicron CMOS, strained Si, SOI etc. It will be particularly beneficial to companies transferring between IC processes facing challenges of high parametric variability (e.g. temperature, supply voltage, and on-die and die-to-die process variability), and problems with timing closure (due to uncertainty in timing data, incompletely characterised cell libraries etc), yet which require re-design and multiple re-spins. The semiconductor industry is highly competitive: to remain ahead of the field, companies must continually improve their products requiring an increased re-use of IP at the RTL and logic level which must be suitable for quick re-targeting to the new processes, combining increased reliability and high performance with reduced size, cost and power consumption.

The development of better algorithms and tools for asynchronous circuit design would thus be of great benefit to the microelectronics industry in the future, and would help create confidence amongst the CAD tool and test equipment vendors, who still consider asynchronous design an exotic option with little demand from practitioners. Continuing research in asynchronous synthesis and testing will play an important role in the training of tomorrow's chip designers through its impact on future undergraduate and postgraduate training. By continuing to interact with such companies as Philips, Sun, Intel, IBM and Atmel we aim to achieve a rapid transfer of the knowledge gained within the project into industry. Silistix and FTL would be immediate beneficiaries of the research.

7. Dissemination and Exploitation

The results generated by the project will be disseminated via publication of academic papers in appropriate international journals and international conferences. We will also endeavour to disseminate the knowledge through industrial channels, in particular through our existing contacts with several UK and European based companies involved in the design of IP cores, SoC and EDA tools for greater variability and DfM. It is envisaged that the success of the project will lead to further research, and industrial collaboration, and enhancement to undergraduate and postgraduate curricula in the area of electronic design and test. Tools developed will be made available under an open-source licence to encourage early adoption. Exploitable results arising from the programme of work will be dealt with through the Universities' Offices of Technology Transfer and Industrial Liaison. Informal collaborations on asynchronous circuit synthesis and testing exist with FTL Systems, Silistix, academic colleagues from Cambridge, Politecnico di Torino, UPC Barcelona, DTU Denmark, TIMA-INPG, Grenoble. These and other links are expected to result in the publication of the next editions of technical reports on the status of asynchronous design in

industry and tools in academia and industry [26], as well a textbook on asynchronous circuit synthesis in which the proposers will play the key role. We wish to organise a school on asynchronous design following the success of the ACiD-WG 2005 Winter School in Cambridge [26]. It is proposed that these collaborations be continued and strengthened under this research programme, and in the areas of information exchange and research results dissemination. Throughout the duration of the project we aim to liaise with various related research groups and projects in the UK and abroad.

8. Resources

9. References

[1] C.H. van Berkel, J. Kessels, M. Roncken, R. Saeijs, F. Schalij, "The VLSI-Programming Language Tangram and its Translation into Hanshake Circuits", *Proc EDAC91*, 1991

[2] C.F. Brej, "Early Output Logic using Anti-Tokens", *Proc. IWLS*03, 2003

[3] I. Blunno, J. Cortadella, A. Kondratyev, L. Lavagno, K. Lwin, C. Sotiriou, "Handshake protocols for de-synchronization", *Proc ASYNC04* ,2004

[4] D. Chai, J.-H. Jiang, Y. Jiang, Y. Li, A. Mishchenko, R. Brayton, "MVSIS 2.0 User Manual", *Tech. Report*, Dept EE & CS, University of California at Berkeley, 2003

[5] J. Cong, An Interconnect-Centric Design Flow for Nanometer Technologies", *Proceedings of the IEEE*, vol. 89, No. 4, pp 505-528, April 2001

[6] J. Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou. "Coping with the Variability of Combinational Logic Delays". *Proc. ICCD04* , pp. 505-508, Oct 2004.

[7] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev. "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers", *IEICE Trans on Information and Systems*, Vol. E80-D, No. 3, 1997.

[8] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Taubin, and A. Yakovlev, "Lazy transition systems: application to timing optimization of asynchronous circuits" *Proc. ICCAD*98 1998.

[9] A.V. Dinh Duc, J.B. Rigaud, A. Rezzag, A. Sirianni, J. Fragoso, L. Fesquet, M. Renaudin. "TAST CAD Tools", *Proc. of the 2nd Asynchronous Circuit Design Workshop*, 2002.

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

[11] A Efthymiou, J. Bainbridge, D. Edwards, "Adding Testability to an Asynchronous Interconnect for GALS SoC" *Proc. IEEE Asian Test Symposium*, 2004.

[12] R. M. Fuhrer, S. M. Nowick, "MINIMALIST: An Environment for the Synthesis and Verification of Burst-Mode Asynchronous Machines" *Proc. 7th International Workshop on Logic and Synthesis (IWLS)*, 1998.

[13] A. Kondratyev, K. Lwin, "Design of Asynchronous Circuits Using Synchronous CAD Tools", *IEEE Design and Test of Computers*, Vol. 19, No. 4, 2002.

[14] A. J. Martin. "The Limitations to Delay-Insensitivity in Asynchronous Circuits", *6th MIT Conference on Advanced Research in VLSI Processes*, 1990.

[15] A. J. Martin, A. Lines, R. Manohar, M. Nystrom, P. Penzes, R. Southworth, U. Cummings, T. K. Lee, "The Design of an Asynchronous MIPS R3000 Microprocessor", *Proc. ARVLSI*, 1997.

[16] S. R. Nassif, "Modeling and Forecasting of Manufacturing Variations", *Proc ASP-DAC01*, 2001.

[17] Semiconductor Industry Association "International Technology Roadmap for Semiconductors, 2004 Update", *http://www.itrs.net/Common/2004Update/2004Update.htm*, 2004.

[18] D. Shang, F.Burns, A.Koelmans, A.Yakovlev, F. Xia. "Asynchronous system synthesis based on direct mapping using VHDL and Petri nets", *IEE Proceedings, Computers and Digital Techniques*, Vol. 151, No.3, May 2004, pp. 209-220.

[19] A.Smirnov, A. Taubin, M. Su and M. Karpovsky. "An Automated Fine-Grain Pipelining Using Domino Style Asynchronous Library", *Proc. ACSD'05*, 2005.

[20] K. Stevens, R. Ginosar, S. Rotem, "Relative Timing", *IEEE Transactions on VLSI Systems*, Vol 11 , No. 1, 2003.

[21] W. B. Toms "Synthesis of Quasi-Delay-Insensitive Datapath Circuits". *PhD. thesis*. Dept. of Computer Science, Manchester University, 2004.

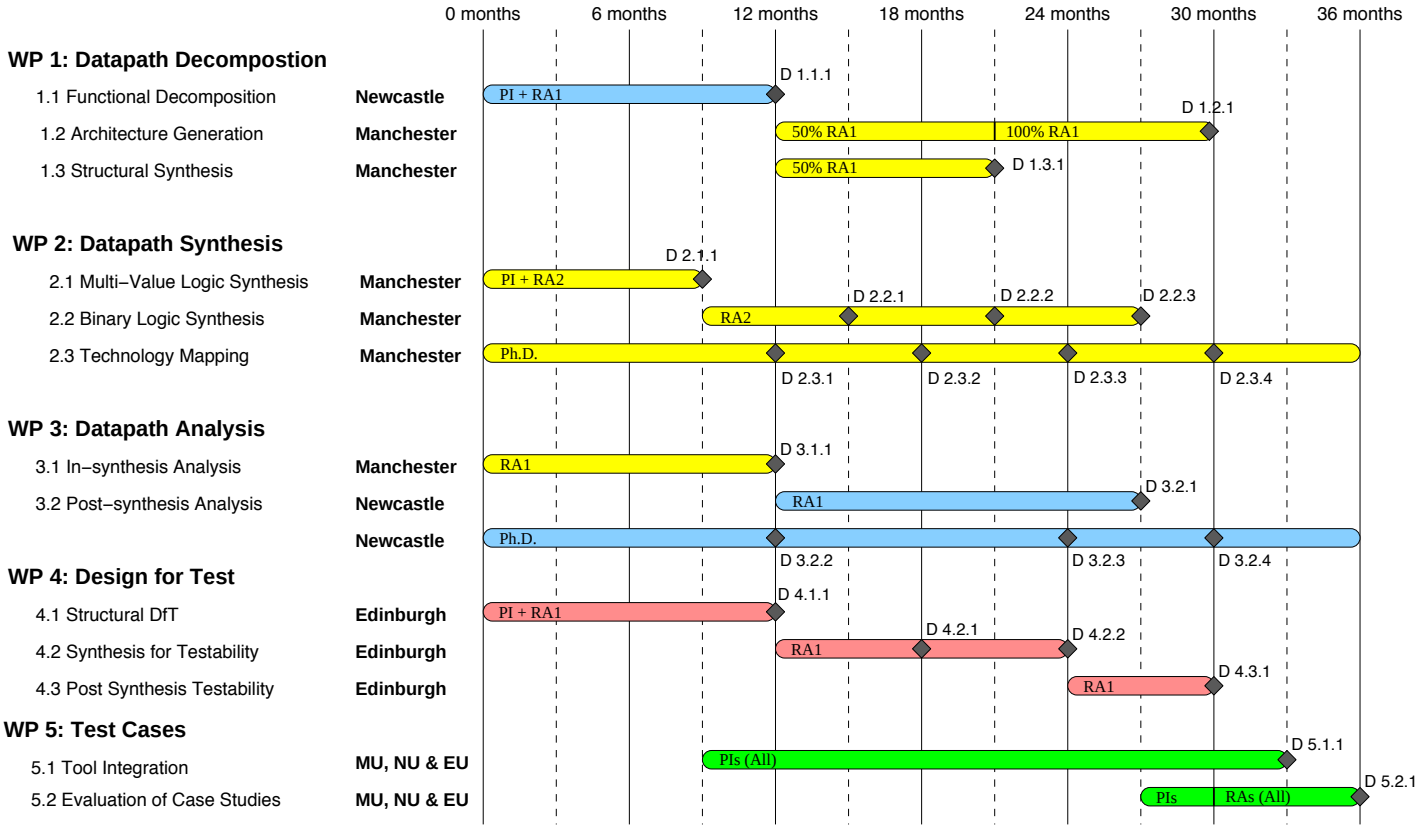
[22] V.I. Varshavsky, ed. "Self-Timed Control of Concurrent Processes: The Design of Aperiodic Logical Circuits in Computers and Discrete Systems", *Kluwer Academic Publishers*,1990.

[23] Ted Vucurevich, "Future EDA Challanges", *Keynote address, IEE EDA Tools Forum*, 2004.

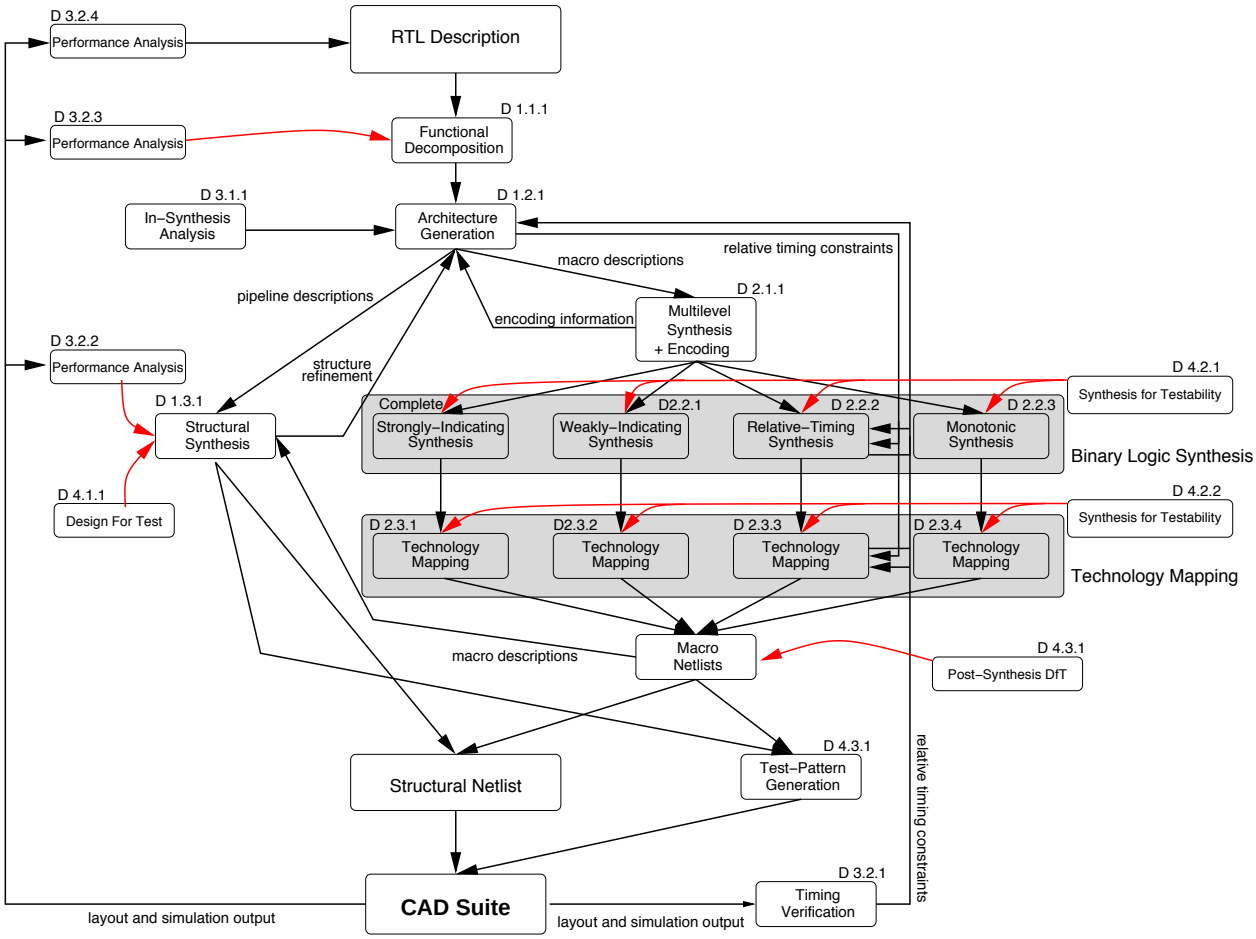
[24] T. Verhoeff. "Delay Insensitive Codes - an Overview." *Distributed Computing Vol. 3*, 1988.

[25] A. Yakovlev, M. Kishinevsky, A. Kondratyev, L. Lavagno and M. Pietkiewicz-Koutny. "On the Models for Asynchronous Circuit Behaviour with OR Causality". *Formal Methods in Systems Design* (Kluwer), Vol. 9, No. 3, Nov. 1996, pp. 189-234

[26] *http://www.scism.sbu.ac.uk/ccsv/ACiD-WG/*



Work-package/Task/Deliverable Gantt Chart



Task Dependency Relationships (within design flow)