Automatic Clock: A Promising Approach Toward GALSification

Mahdi Jelodari Mamaghani*, Milos Krstic† and Jim Garside*
*School of Computer Science, The University of Manchester, Manchester M13 9PL, UK
†IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany
Email: *Mahdi.Jelodari@manchester.ac.uk, †Krstic@ihp-microelectronics.com, *Jim.Garside@manchester.ac.uk

***Abstract*—Hardware design abstraction has significantly favoured productivity in the recent years. The clock is known to be the beating heart of every digital design which coordinates the communications and computations. Due to the critical role of this signal a proper management of it is essential. Newly emerged high-level synthesis and hardware construction tools either reflect this responsibility to the designer at high level or make some general assumptions based upon critical paths which may also require the designer to re-architecture the design when the assumptions encounter failure. This can exert a profound impact on designer's productivity. We propose the AutoCLK technique to handle the clock automatically which calls for specific properties, such as 'slack elasticity' and distributed control flow, in the target architecture. Our experiments demonstrate that both low-level and high-level factors have to be taken into account for efficient clock management.**

I. Introduction

The rapidly growing complexity of System-on-Chips (SoCs) has drawn the attention of the industry to the productivity factor more than before. Recent advancements in chip design by the EDA industry has paved the way for deeper and faster exploration of hardware systems. Bluespec and recently Chisel have emerged to address the productivity issue by raising the level of abstraction to Haskell- and Scala-like languages. These are designated as hardware construction languages which yet require the designer to be knowledgeable in hardware design.
This has also attracted the software community who would like to leverage hardware computing techniques for energy or performance purposes. In this regard, VivadoHLS (formerly AutoPilot) and LegUp have emerged as High-Level Synthesis (HLS) flows to enable C programmers to use FPGA systems without need for having hardware design skills.

The productivity offered by HLS has two major sources: i) provides the designer with parametrisable libraries of pre-designed IPs to be reused in the development process, ii) allows separation of timing from functionality; Although this largely simplifies the development process for the designers with bear hardware knowledge, clock management yet remains an obstacle. A conventional approach is to use HLS flow to generate RTL. Then, one can define clock domains and manually insert FIFOs and synchronisers into the design accordingly. Alternatively, Chisel or Bluespec can be leveraged to define clock domains at higher abstraction level. This may need the designer to re-architect the design when the timing constraints are not met.

To tackle this problem we propose the Automatic Clock (AutoCLK) technique which aims to handle clocking in a digital design automatically. In this respect the eTeak synthesis framework [1] is exploited as it has the essential properties for automatic clocking (see Section II). eTeak is employed to generate FIFO-based GALS architectures [2]. To determine their sizes the worst-case scenario is usually considered. This may impose area and energy overhead to the system [3]. AutoCLK can avoid this by reconsidering the elastic boundaries with regard to the timing requirements which may lead to more appropriate FIFO sizing, and thus less expense.

II. Automatic Clock

We define AutoCLK as the process of introducing a common timing discipline to a high-level synthesis flow. This overcomes the lack of familiarity of the designer at software level with the hardware-level details such as timing. On the other hand, it enables automatic clock management whilst taking *low-level constraints* (e.g. critical path delays, EMI, etc.) and high-level patterns (e.g. critical rate, architecture, input dataset) into account. Therefore it is necessary for the synthesis tool to have control over these factors.

In this respect two main properties are essential: i) a distributed control/data architecture with relaxed global timing assumptions. This removes the concerns with the activation/compilation of the tasks when automatically refining the control flow by re-sizing the buffers and ii) a slack elastic [4] network which allows any bounded degree of storage on the communication channels. This property enables the tool to re-architecture the circuit through buffer insertion and buffer re-sizing which can influence the local/global cycle times and impact the critical path delays, consequently.

Equation 1 proposes Clk_{mgr} the core function of the AutoCLK process. It takes low-level $constraints$ and high-level $patterns$ as input and returns two sets $arch$ and $freq$ which are the FIFO sizes with every link and the clock frequencies associated with every domain, respectively. Note that $patterns$ are extractable from the high-level code.

$$freq, arch = Clk_{mgr}(patterns, constraints) \quad (1)$$

III. Experimental Results

A. Case Studies: SSEM Processor & Hash Search

Two case studies are exercised to study the properties of the dataflow networks toward GALSification: i) a three-stage 32-bit processor (SSEM [5]) and ii) a Hash Join function as a producer/consumer model. Both are implemented as

independent procedures in the CSP-like language of Balsa and synthesised using eTeak. The generated fine-grained dataflow networks consist of 146 and 42 combinatorial primitives including Join, Fork, Steer, Merge, Variables and Operations which are connected via 200 and 65 channels, respectively. In these networks high-level variables are transformed to pipeline stages with read/write ports. Therefore the longest path from a read port of variable V_{n-1} to the write port of the successor variable V_n is designated as $criticalpath$. Both systems have access to external memories modelled in Verilog. The instruction/data memory associated with the processor is loaded with a GCD(12, 8) program with 30 RISC instructions while the hash function uses a hash table with 64K 48-bit buckets and a load factor of 0.6.

eTeak uses SELF [6] to adopt a synchronous elastic timing discipline to its fine-grained dataflow circuits. The SELF-adopted circuits of eTeak are considered as baseline architecture operating on single clock running at $\frac{1}{CriticalPathDelay}$ frequency.

B. Low-level Constraints and High-level Patterns

In this study critical path delays are assumed as low-level constraint and levels of granularity (component and stage) as high-level pattern. To explore AutoCLK single and multiple clocking policies are considered. In the multiple clocked designs clocks are asymmetric and are set based upon the 'local' critical path delays; asynchronous cascaded buffers are employed as FIFOs for communication between the grains.

C. Results and Discussion

Figure 1 shows the difference between the multiple clocked processor and the single clocked counterpart with FIFOs on every link. The design with multiple domains exhibits an smooth increase in execution time compared to that of the SELF-adopted design where the delay of the synchronous FIFOs drastically influences the execution time. The same behaviour is observable when the level of granularity is coarser and variables are expanded as FIFOs. The global algorithmic cycle in SSEM is the reason for this behaviour as it constrains the number of active tokens in the design. Whilst this constraint does not exist in the Hash system where LFSR and the search loop are running independently. Although figure 2 shows a coarse-grained GALS pipeline can outperform the fine-grained ones, an improvement by 20% may occur (FIFO size = 1) if FIFOs are sized appropriately. Note that for every run FIFO depths are incremented equally. This factor can vary for every FIFO with regard to level of granularity and input data. Determining the exact depths is subject for future research.

This experiment shows how high-level information can influence the decisions made by the tool for re-timing the design. These explorations are not applicable to a commodity pipeline without considering modifications to its global timing requirements unless it is de-synchronised and transformed to a latency-insensitive design. Cycle-accurate information at RTL prevents a system to be modelled using dataflows as the connections between datapath and control registers are not always explicit. This constraints the possible architectural explorations.

IV. Conclusion

This work proposed the automatic clock technique which exploits the distributed and latency-insensitive properties of

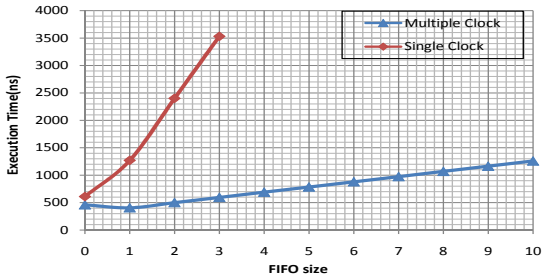

Fig. 1. FIFO re-sizing impact on a single and multiple clocked dataflow processor at a fine level of granularity.

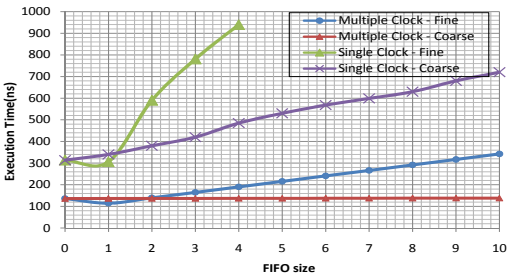

Fig. 2. FIFO re-sizing impact on a single and multiple clocked Hash Join at different levels of granularity.

the eTeak generated circuits to explore GALSification. This study shows how high-level patterns such as architecture, input dataset and the level of granularity along with low-level constraints can impact the performance of a GALS system. Using this information the tool can re-architect the design through re-timing to satisfy the user defined constraints. The proposed technique can largely influence designer's productivity and it can be exploited by the reconfigurable industry where a design has to be flexible for refinements based on the input enquiries. As future work we will consider a broader range of scenarios to collect enough data for training the AutoCLK process.

V. Acknowledgement

We would like to thank anonymous reviewers for their feedback. This work is supported by UK EPSRC and the industrial partners, IHP GmbH and Infineon Technologies, under grant GAELS (EP/I038306/1). Mahdi is also holding an EPSRC Doctoral Prize Fellowship.

References

[1] M. Jelodari Mamaghani, W. Toms, and J. Garside, "eTeak: A data-driven synchronous elastic synthesiser," in *13th International Conference on Application of Concurrency to System Design, PhD Forum*. IEEE, 2013, pp. 134–137.

[2] S. Suhaib, D. Mathaikutty, and S. Shukla, "Dataflow architectures for GALS," *Electronic Notes in Theoretical Computer Science*, vol. 200, no. 1, pp. 33–50, 2008.

[3] M. Krstic, E. Grass, F. Gurkaynak, and P. Vivet, "Globally asynchronous, locally synchronous circuits: Overview and outlook," *Design Test of Computers, IEEE*, vol. 24, no. 5, pp. 430–441, 2007.

[4] R. Manohar and A. J. Martin, "Slack elasticity in concurrent computing," in *Proceedings of the Fourth International Conference on the Mathematics of Program Construction*. Springer-Verlag, 1998.

[5] S. Lavington, "A History of Manchester Computers (2nd ed.), Swindon: The British Computer Society," 1998.

[6] J. Cortadella, M. Kishinevsky, and B. Grundmann, "Self: Specification and design of synchronous elastic circuits," in *International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU)*, 2006.