2016 Symposium on Asynchronous Circuits and Systems

Automatic Clock: A Promising Approach Toward GALSification

Mahdi Jelodari Mamaghani*, Milos Krstic[P], Jim Garside*

*University of Manchester, UK [P]IHP, Germany

Outline

- Introduction to Elasticity
- GALSification and the Benefits
- Conventional GALS Design Methodologies
- *Automatic Clock:* An Architectural Exploration Technique
  - eTeak synthesis framework
  - Patterns and constraints classification
  - Some results
- Conclusion

Why "Elasticity" Matters

- Elasticity at system level: the degree to which a system autonomously adapts its capacity (resources) to workload over time.
- Elasticity at circuit level: the degree to which a circuit autonomously tolerates *unexpected* timing variation in communication and computation.
- Elasticity:
  - can handle non-determinism and uncertainty
  - can be achieved using fine-grained synchronisers

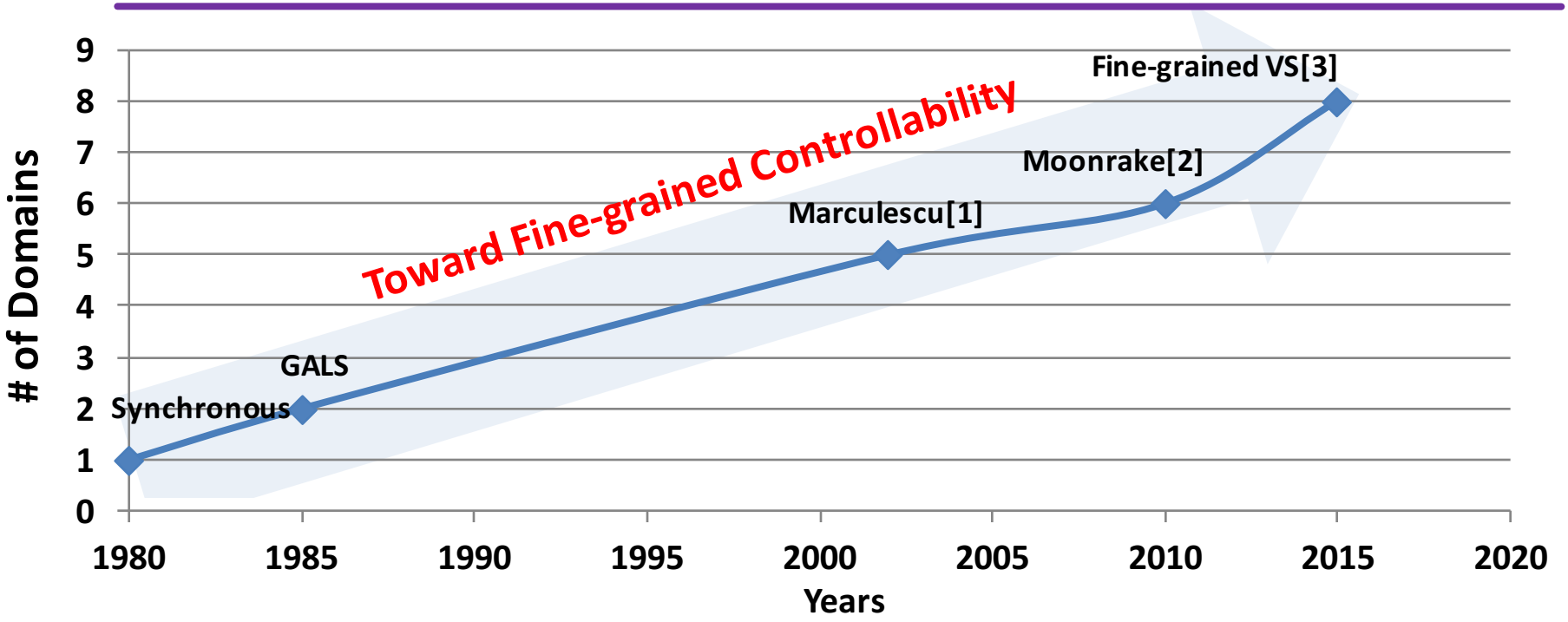
Elasticity could introduce an overhead in terms of area and performance!

GALS vs. Elastic Design

- GALS design approach is similar to elastic approach, however with elasticity reduced to LS blocks
- GALS allows that system components run at different clock frequency resulting in power and performance advantages
- Fine-grained GALS is more energy efficient
- High-level GALS synthesis could facilitate design space exploration and optimisation

GALS is more beneficial when applied at finer level of granularity!

Heterogeneity over Time



[1] Iyer, Anoop, and Diana Marculescu. **"Power and performance evaluation of globally asynchronous locally synchronous processors**." *Computer Architecture, 2002*

[2] M. Krstic *et al*., "**Moonrake chip - GALS demonstrator in 40 nm CMOS technology**," *System on Chip (SoC), 2011 International Symposium on*, Tampere, 2011, pp. 9-13.

[3]. Waclaw Godycki et al. **"Enabling Realistic Fine-Grain Voltage Scaling with Reconfigurable Power Distribution Networks."** *MICRO* 2014

Conventional GALS Design

- Prior art: Clock boundaries defined by the designer
- Coarse-grained GALS, a collection of synchronous IPs

$$Freq = Clk_{mgr}(Constraints)$$

where constraints are the critical path delays

- High-level GALSification: Bluespec/Chisel which provide the designer with *clock type* definition

Clock definition is reflected to the synchronous designer which has productivity drawbacks

Automatic Clock: Our Contribution

- Clock is important and has to be handled automatically
- Another degree of freedom is considered: Architectural exploration of GALS systems
- Fine-grained exploration of GALS partitions is possible
- Architectural changes may impact the clock frequency

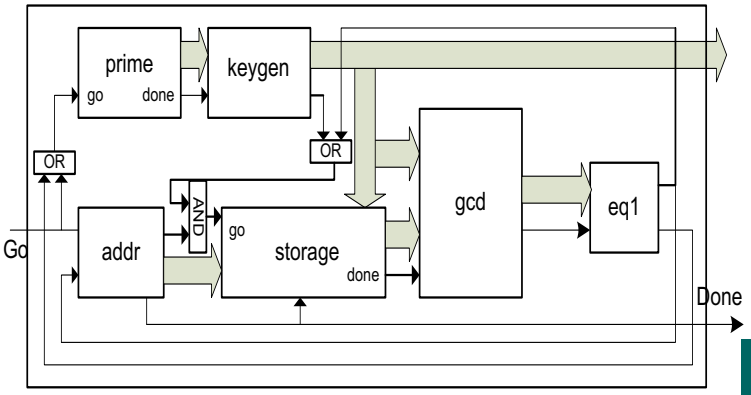
Freq, Arch = Clk$_{mgr}$(Constraints, Patterns)

A generalised function for handling the clocks

AutoCLK considers an architectural degree of freedom to the GALS space exploration.

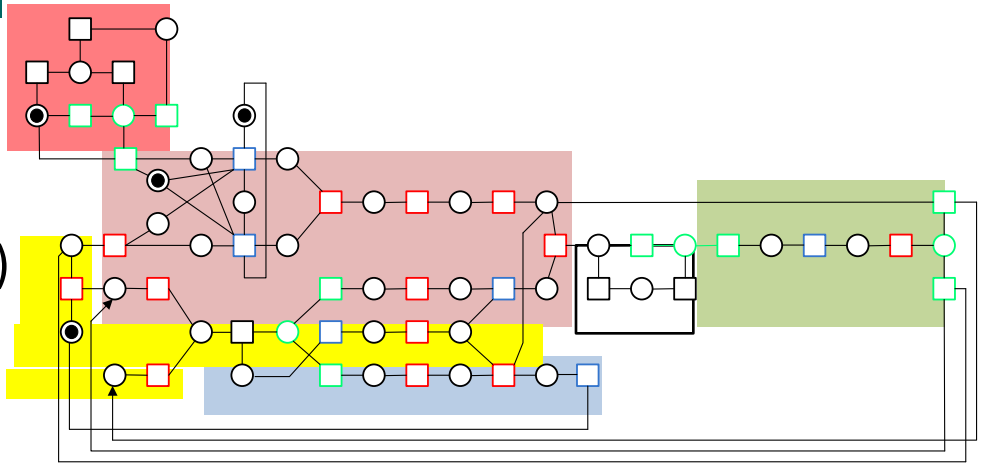
Automatic Clock (AutoClk)

Freq, Arch = Clkmgr(Patterns, Constraints)



High-level Patterns:

- Communication rates
- Functional Behaviour
- Level of Granularity



Low-level Constraints:

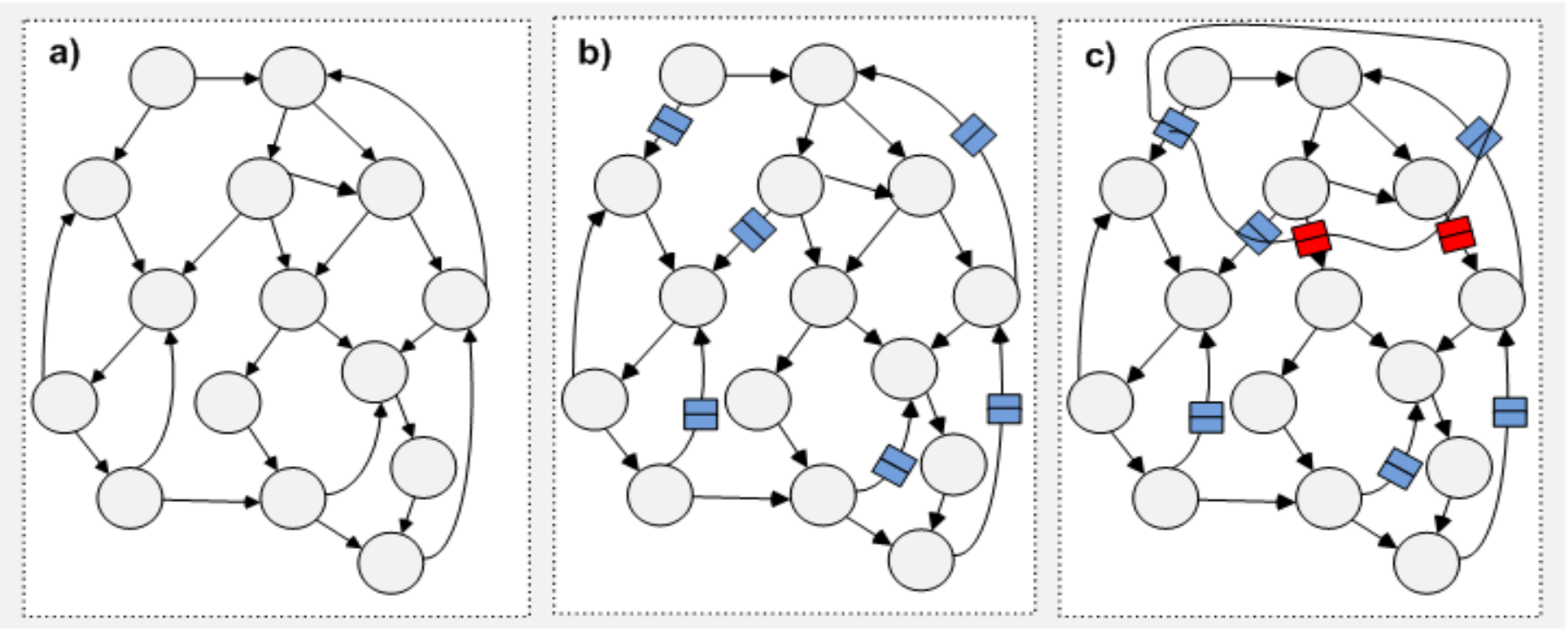
- Place & Route
- Timing information (critical path delays)
- IO bandwidth

eTeak Synthesis Framework

eTeak is a Synchronous/GALS Dataflow Synthesis backend for the Balsa language:

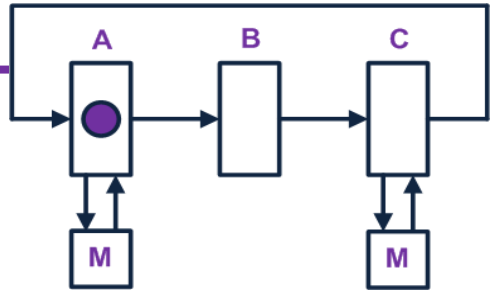
- **Communication**:
  - **Point-to-point communication** between computation blocks.
  - **Slack elastic** channels are capable of storing 'any number' of tokens.
- **Computation**:
  - **Macro-module** style with separate **Go** and **Done** activation signals.
  - **Dataflow** which realises data-dependent computation

AutoCLK Adoption into eTeak

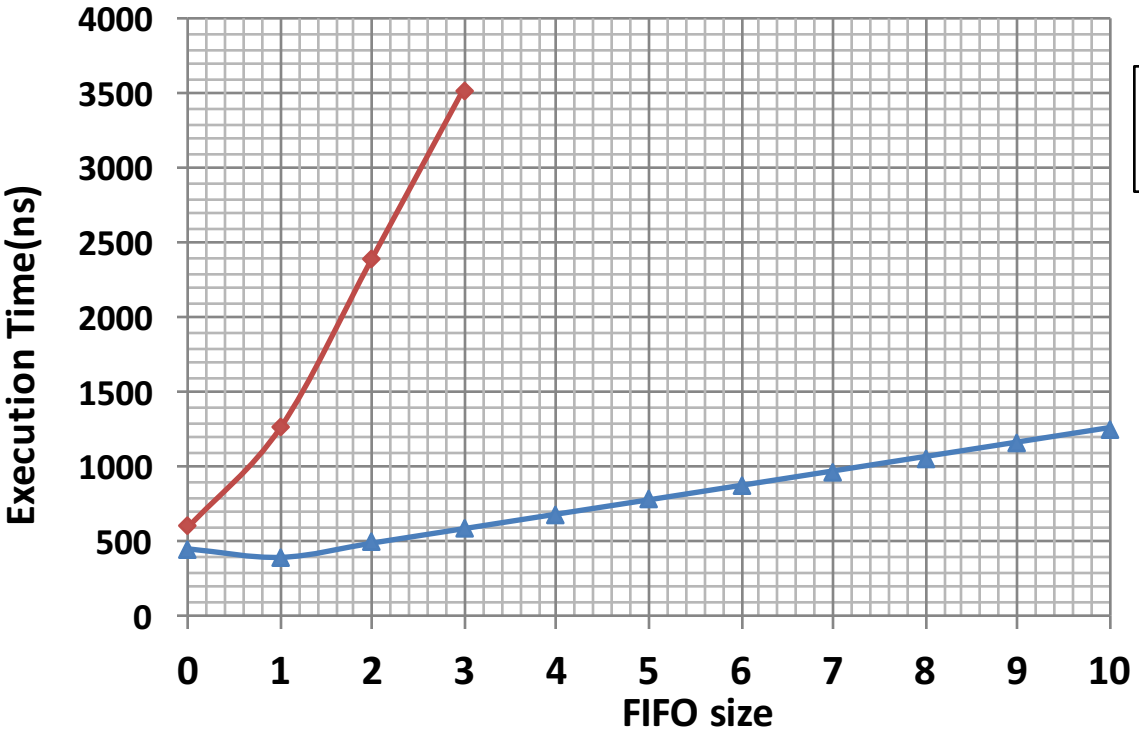


- ✓ eTeak generated Synchronous Elastic Clocked Circuits
- ✓ (Data) **FIFOs** inserted to ensure deadlock-freedom
- ✓ Local **FIFOs** converted for interfacing
- ✓ Time-safe **FIFOs** inserted for interfacing

Results: SSEM Processor

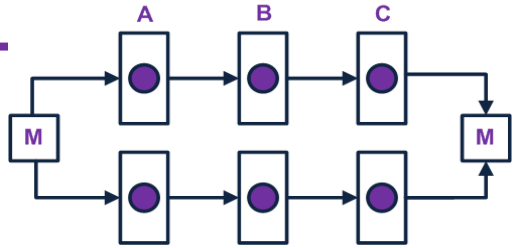


- A fine-grained GALSified Processor

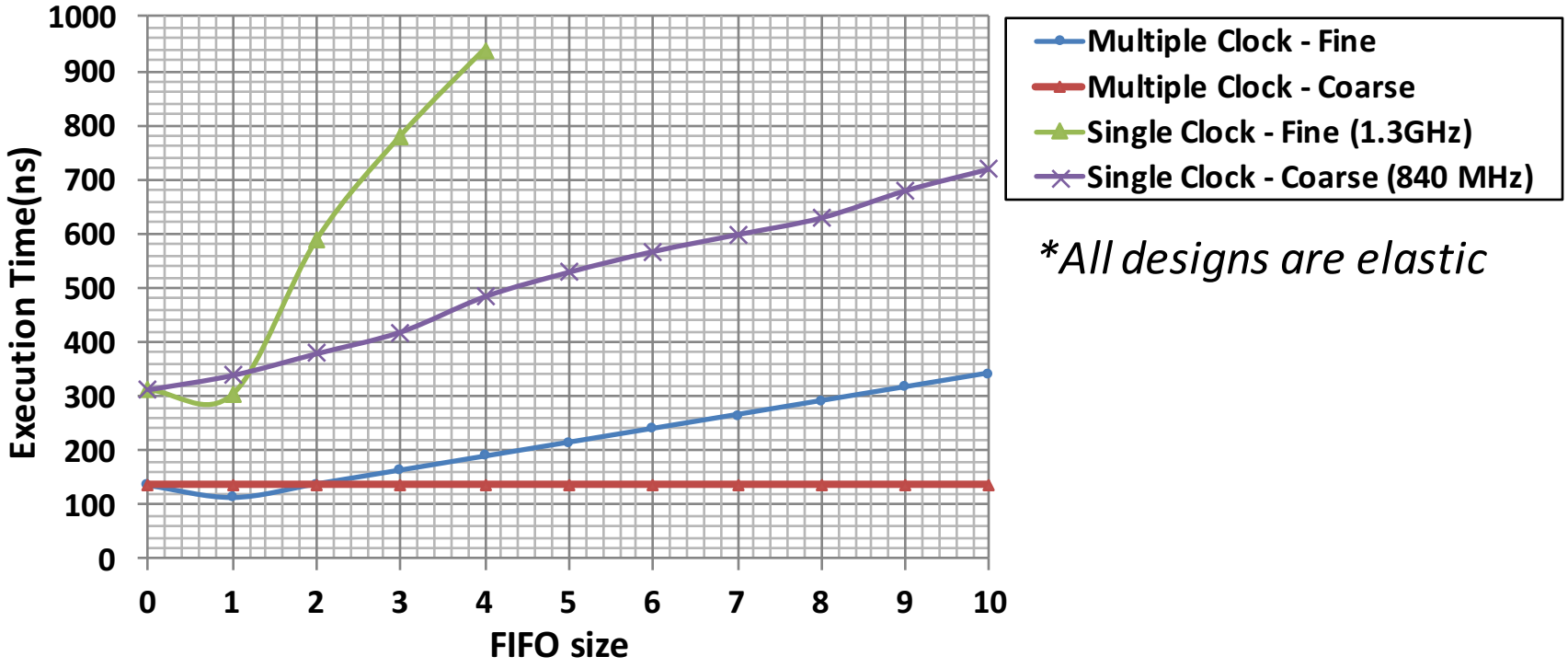


*All designs are elastic

Results: DBMS Benchmark



- A fine- vs. Coarse-grained GALSified Database Search Probe



*All designs are elastic

Conclusion

- Elasticity/GALS could introduce the overhead in terms of area and performance.
- Careful high-level approach is required in handling GALS
- Partitioning has to be handled automatically by taking high-level patterns and low-level constraints into account.
- AutoCLK considers an architectural degree of freedom to the GALS space exploration.

Questions

***eTeak is open-source: https://github.com/balangs/eTeak**