Programmable Logic Core Based
Post-Silicon Debug for SoCs

Bradley R. Quinton and Steven J.E. Wilton
University of British Columbia
Vancouver, B.C., Canada

What this talk is about:
Enhancing ASIC debug using embedded FPGA cores
- Use the embedded FPGA to implement debug circuitry

This talk:
1. Our basic debug architecture
2. Network architecture for “tapping” internal signals
   a) Network topology: concentrators
   b) Synchronous vs. asynchronous networks
3. Bus Interface Architecture
4. Overall Area Overhead estimates
Part 1: Our Debug Architecture

Baseline IC
**High Level Architecture**

Observability:

1. Select signals using the network
2. Process these signals with the PLC
3. Return the test results
High Level Architecture

Signal Control:
1. Create circuits in the PLC that interact with the device
2. Selectively override signals using the network
3. Observe results

High Level Architecture

Correct/Change:
1. Interrupt block output signals
2. Manipulate these signals using the PLC logic
3. Create new device behaviour
Part 2: Network Topology

Network Definition/Details
This network needs to be:
- Small and fast
- Non-blocking

We can take advantage of the fact that each PLC pin is equivalent
Concentrator Networks

A network that exactly matches these requirements has been defined in previous network theory research.

A concentrator network provides full connectivity and takes advantage of the I/O flexibility of the PLC.

an \((n,m)\)-concentrator is defined as:

a network with \(n\) inputs and \(m\) outputs, with \(m \leq n\), for which every set \(k \leq m\) of the inputs can be mapped to some \(k\) outputs, but without the ability to distinguish between those outputs.

The area is lower than a permutation network
Depth half that of a permutation network


Part 3: Network Implementation:
Synchronous vs. Asynchronous
Asynchronous Networks

In modern process technologies wire delay can be significant with respect to gate delay, this makes communication that spans the entire die more complex.

Classic Synchronous Solution: Pipelining
- Difficult global clock construction

Asynchronous Techniques: Self Clocking
- Do not need a global clock
Two methods:

1. Bundled-data
   - control signaling is separate from the data
   - requires delay-matching*

2. Delay-insensitive
   - control signaling encoded with the data
   - no delay-matching* required

* Arbitrary delay-matching is a difficult CAD problem, and is not supported by most tools.

We use ‘dual-rail’ encoding to minimize the depth of the control decode.
Compare Synchronous and Asynchronous

we created 9 ICs based on the TSMC 0.18µm

- 3 core die sizes:
  - 3830x3830 µm (~1 million gates),
  - 8560x8560 µm (~5 million gates),
  - 12090x12090 µm (~10 million gates)

- 3 different block partitions:
  - 16 blocks
  - 64 blocks
  - 256 blocks

Improved throughput without a global clock
Compare Synchronous and Asynchronous

For large, high-speed ICs it is possible to achieve a high throughput with asynchronous interconnect while avoiding a global clock for pipeline registers.

However, the advantage does not justify the added complexity of dealing with asynchronous logic, therefore for the remainder of our work we will use synchronous interconnect.

Detailed Results:
Part 4: Programmable Logic Interface

Interface Challenges

Circuits implemented in a PLC will inevitably have **lower timing performance** and logic density than fixed function circuits.

This fundamental mismatch in performance makes the **interface** between the PLC and the rest of the SoC a challenging problem.
**PLC Modifications**

Our goal is to maintain the standard *island-style PLC architecture* while enhancing some of CLB structures.

**CLB Enhancements**

We use the ‘shadow cluster’ concept to ensure that the new circuits will integrate into the existing *routing architecture*, and to reduce the effective *area overhead*. 
PLC Interface Conclusions

Improves interface timing by $36.4\%$, reduces CLB usage by $7.9\%$ and improves routability by $28.8\%$ for circuits that require system bus interfaces.

Area overhead is less than $0.5\%$ for circuits that do not require system bus interfaces.

Detailed Results:

Part 5: Post-Silicon Debug
Area Overhead / Cost
To understand the area overhead of our scheme for a range of ICs we created a set of parameterized models.

We used a 90nm standard cell process.

We targeted the 90nm IBM/Xilinx PLC with a capacity of approximately 10,000 ASIC gates.

The network was implemented using standard cells.

All area numbers are post-synthesis, but pre-layout.

Area Overhead - Overall

- 20M gate device, 7200 signals for ~ 5% overhead
Conclusions

We have shown that it is feasible to integrate a PLC in a fixed-function IC in such a way that it could be used to assist post-silicon debug.

Key: Flexible network to connect PLC to chip
- Based on Concentrator network
- Can be synchronous or asynchronous

Also important to have bus interface support

We have shown that for many ICs the area overhead of this scheme is well below 10%.