Weaver-DC: Asynchronous Micropipeline Synthesis Flow

ECE, Boston University

The problem

Asynchronous design has long been proposed as a solution to clock and timing convergence related problems but [so far] never occupied any significant niche on the market. The biggest problem of its acceptance has been identified in the absence of industrial quality EDA support. With relatively low academic resources rich EDA support cannot realistically be created from scratch. Even if it would be feasible the paradigm shift (in specification formats, synthesis methods and operational principles etc) is too big comparing to immediate benefits.

The solution

Chameleon tools take advantage of industry standard synthesis engines to obtain RTL implementation from high-level HDL specification and further re-synthesize the RTL into self-timed clockless implementation. When the RTL is synthesized using an RTL library known to the tool the circuit functionality can be easily identified and low computational complexity substitution-based re-implementation can be applied.

By using local handshaking instead of global synchronization self-timed implementations well sustain manufacturing and environmental variations thereby increasing yield and robustness. Graceful performance degradation at lower voltage simplifies the performance power consumption trade-off implementation.

The tool

Weaver-DC – is a chameleon flow for synthesis of asynchronous micropipeline circuits from high-level HDL specifications. Weaver-DC works in integration with Synopsys Design Compiler used for RTL synthesis and solving fan-out violations in the final netlist.

The tool consists of a set of TCL scripts guiding the synthesis flow and implementing the user commands functionality and the Weaver engine written in C++ and using SAVANT VHDL compiler to interface with the RTL synthesis engine using VHDL and Synopsys Liberty parser to Support standard library specifications.

Internally Weaver-DC uses VHDL but the Synopsys Design Compiler front end allows handling variety of specification formats.

Weaver-DC features

Weaver is under development and the on-going work is primarily targeted at supporting the area-performance trade-off by allowing variable pipeline granularity and at architectural power saving techniques. Currently (in version prerelease v0.94) the flow features:

- micropipeline implementations is shown to be flow equivalent to the RTL implementation synthesized by the RTL synthesis engine i.e. the sequences of data values appearing on any given micropipeline data channel to the sequence of values settled during setup intervals on the channels (data wires) in the original RTL implementation
- low computational and spatial complexity re-implementation algorithms
- support for various micropipeline pipelining protocols and implementations including binary dual-rail and ternary single-rail data encoding through library approach
- automatic gate-level pipelining with hierarchical slack matching (similar to retiming in RTL) and micropipeline optimization in many cases improves the circuit performance proportionally to its depth
- test bench automation
- industry standard Synopsys Design Compiler front end provides familiar tool interface (Weaver-Dc extends the set of DC commands and runs within its shell) and allows support for variety of input specification formats with no changes or extensions
- micropipeline library specification using industry standard Synopsys Liberty extensible format extended with micropipeline specific attributes
- simulation automation (with Mentor Graphics ModelSim)
- automatic library installation minimizes the library developer effort

Micropipeline libraries

Although Weaver-DC supports compound micropipeline stages (stages composed of more than one physical library cell) and covers the vast majority of handshaking protocols as of the prerelease v0.94 Weaver does not provide any support for stage delay calculation or any control over routing. That makes it safest to use the protocols with delay insensitive (DI) inter-stage communication. Quasi-delay insensitive (QDI) micropipelines comprise one very extensively researched group of protocols and implementations well suited for the Weaver flow.

We developed “proof of concept” QDI micropipeline libraries: dual-rail with dynamic domino-style logic implementation (similar to PCHB), balanced dual rail-dynamic library for side channel attack resistant implementations and an experimental ternary logic based library with single-rail data channel using TSMC 0.18um process. UCL made available their Single-Track and PCHB libraries through MOSIS (at this point we are unaware if those have been tried with Weaver flow).

With unbalanced libraries experimental results on average show x6 area overhead and fixed close to 700MHz performance due to the gate level pipelining depending primarily on the library cells cycle time.

 http://async.bu.edu/weaver