Ten years of petrifying: where are we now?

Jordi Cortadella, Univ. Politècnica Catalunya, Barcelona, Spain
Alex Yakovlev, Univ. of Newcastle upon Tyne, UK

With thanks M. Kishinevsky, A. Kondratyev and L. Lavagno
Topics

• What does Petrify do?
• Where has it been used?
• How does it compare with other tools?
• What is the future of async logic synthesis?
Design Methodology and CAD tools are essential to cope with complexity

Source: Sematech
Role of logic synthesis

• Automatic logic synthesis has been a major success story in (synchronous) EDA in the 80s and 90s
• The key model behind it is FSM
• Why auto-synthesis for asynchronous circuits
  – to help widespread use of async circuits
  – async circuits are highly concurrent – even harder to design manually
  – The key model is Petri nets (captures concurrency in natural form)
Basic research
(well-founded theory)

Applied research
(tools and experiments)

Theory for the synthesis of speed-independent circuits and relative-timing

petrify
ACiD-WG workshop, Turku, June 2004
Specification

Implementation

synthesis
Signal Transition Graph (Petri net)

State Graph

Encoded State Graph

synthesis
ACiD-WG workshop, Turku, June 2004

Encoded State Graph

synthesis

DTACK DSr DLDTACK

LDS = 0

0 1 0

-- -- --

00 01 11 10

0 0 0

-- -- --

00 01 11 10

00 01 11 10

Encoded State

Graph

00000

01000

10010

10110

10111

11111

01110

01110

DTACK

D

DSr

LDS

csc

LDTACK

dsr+

D-

DTACK+

DTACK-

LDS+

LDTACK+

LDTACK-

LDS-

LDTACK

DSw+

DSw-

DTACK-
ACiD-WG workshop, Turku, June 2004

Complete State Coding (CSC)

Encoded State Graph

synthesis
**Boolean equations:**

\[
\text{LDS} = D \lor \text{csc} \\
\text{DTACK} = D \\
D = \text{LDTACK} \\
\text{csc} = \text{DSr}
\]
Main Features

• Synthesis of circuits from STGs
  – State encoding and decomposition
  – In complex gates, and in given libraries
  – Under timing assumptions

• Synthesis of Petri nets from Transition Systems
  – Extraction of concurrency (using theory of regions)
  – Visualisation and backannotation
Dissemination of results

*Books, tutorials, journals, conferences, ...*
Previous books by the team:

M. Kishinevsky, A. Kondratyev, A. Taubin and V. Varshavsky,
*Concurrent Hardware: The theory and practice of self-timed design*,
John Wiley & Sons, 1994

L. Lavagno and A. Sangiovanni-Vincentelli,
*Algorithms for Synthesis and Testing of Asynchronous Circuits*,
Tutorials

Summer schools:
- Lyngby, 1997
- Grenoble, 2002

Academic institutions:
- Samsung Labs, 1996
- Acorn Networks Inc., 1999
- Fujitsu Labs, 1996
- Dagstuhl, 2001
- Eilat, ASYNC-00
- Aarhus, ICATPN 2000
- Bangalore, VLSI-02

Locations:
- San Jose, ICCAD-95
Publications and Ph.D. thesis

• 13 journal papers

• 23 conference papers
  – ICCAD, DAC, ASYNC, ICATPN, EDTC, ...

• Invited lectures
  – DAC, ICATPN, CSD, HDPN, ACiD-WG

• 4 Ph.D. thesis and 8 in progress
Practical impact of the research

Prototypes,
Teaching and
Research
Used in several prototypes

- AMULET microprocessors (Manchester University)
Used in several prototypes

- AMULET microprocessors (Manchester University)
- Instruction-length decoder (Intel Corporation)
Used in several prototypes

- AMULET
- Instruction-length decoder (Intel Corporation)
- DSP for hearing aids (Tech. Univ. of Denmark)
Used in several prototypes

- AMULET microprocessors (Manchester University)
- Instruction-length decoder (Intel Corporation)
- DSP for hearing aids (Tech. Univ. of Denmark)
- Smart cards (Cambridge University)
Used in several prototypes

- AMULET microprocessors (Manchester University)
- Instruction-length decoder (Intel Corporation)
- DSP for hearing aids (Tech. Univ. of Denmark)
- Smart cards (Cambridge University)
- Arbiters and GALS circuits (McGill University)
- and others
  - Theseus Logic
  - AT&T Cambridge
  - ...
Teaching (besides our groups)

- One chapter of Furber & Sparso’s book
- Technical University of Denmark
- Technion (Israel)
- University of Kaiserslautern (Germany)
- Boston University (USA)
- Indian Institute of Technology (Mumbai, India)
- University of Aizu (Japan)
- McGill University (Canada)
- Oregon State University (USA)
- ...
Used in research

- University of Manchester (UK)
- South Bank University (UK)
- IIT Mumbai (India)
- CSEM (Switzerland)
- University of North Carolina (USA)
- KAIST (Korea)
- University of Augsburg (Germany)
- Universidad de Zaragoza (Spain)
- ... and many individuals using the tool by their own
Comparison with other tools doing control synthesis

• Minimalist: based on an FSM model, uses fundamental mode (Petrify uses input-output mode), less power in terms of handling concurrency

• ATACS: uses event-based notation, but does not seem to have the power of solving state encoding problems; allows relative timing but without backannotation and visualisation
Other tools complementing Petrify design flow

- Graphical STG editor: VSTGL
- HDL front end: Pipefitter
- CSC visualization tool: ConfRes
- Verification with timing assumptions: Transyt
- Large STG analysis using unfoldings: Punf/CLP
- Direct mapping of STG: OptiMist
- DI process algebra front-end: di2pn
- STG Decomposition by contraction: DESI
- STG+XBM co-synthesis: CASCADE
Input specification: VSTGL
(technical University of Denmark)
Possible future design flow (based on pipefitter)

- System Spec (Behavioural Verilog)
  - Control/Data Splitting and Optimisation
    - Control Unit Spec (STG)
      - Logic Synthesis (Petrify)
        - Control Unit Implementation (Verilog Netlist)
      - Direct Mapping (David Cells)
        - Datapath Implementation (Verilog Netlist)
  - Datapath Synthesis
    - Datapath Spec (Synthesizable Verilog)
      + Logic Synthesis + Timing Analysis + Delay Insertion
  - Functional Simulation
  - Timing Simulation

- Testbench (Behavioural Verilog)
  - System Timing Info (Back-annotated Verilog Netlist)
    - Extraction of Timing Info

- System Spec
  - System Timing Info Extraction
  - System Implementation (Verilog Netlist)
    - System Layout
      - Placement & Routing

- Simulation
  - Control/Data Splitting and Optimisation
  - Datapath Synthesis

ACiD-WG workshop, Turku, June 2004
Future for Petrify

• Better front-end (CSP-like and HDLs)
• Better state space management (unfoldings, structural methods)
• Better use of direct mapping (at appropriate levels)
• Decomposition at various levels (CSP, labelled PN, STG, logic)
• Applications: desynchronisation and GALS, interfaces, control logic
Desynchronisation (Async 2004)

- The de-synchronization model provides an abstraction of the timing behavior
A+                   B+ A- B-
A+                   B+ A- B-
A+                   B+ A- B-
A+                   B+ A- B-

**de-synchronization model**

**fully decoupled**

*(Furber & Day)*

**simple 4-phase**

**semi-decoupled**

*(Furber & Day)*

**non-overlapping**

GasP, IPCMOS

ACiD-WG workshop, Turku, June 2004