Asynchronous Design Seminar at University of Verona – Lecture Notes 3

De-Synchronization

De-synchronization Theory and Fundamentals
Prior work

- **Micropipelines** (Sutherland, 1989)

- **Local generation of clocks**
  - Varshavsky et al., 1995
  - Kol and Ginosar, 1996

- **Theseus Logic** (Ligthart et al., 2000)
  - Commercial HDL synthesis tools
  - Direct translation and special registers

- **Phased logic** (Linder and Harden, 1996)
  (Reese, Thornton, Traver, 2003)
  - Conceptually similar
  - Different handshake protocol (2 phase vs. 4 phase)
Flow equivalence

[Guernic, Talpin, Lann, 2003]
Flow equivalence = Cycle Accuracy!

Flow equivalence = Cycle Accuracy!

Asynchronous Control Circuit Design - L3
Synchronous Circuit

Clock

Asynchronous Control Circuit Design - L3
De-Synchronized Circuit

Two latches replace every flip-flop

Clock generators manage data flow

Delay Elements
Track combinational cloud delays

Clock generators manage data flow
Asynchronous Control Circuit Design - L3
Can we increase concurrency?

not flow-equivalent

data overrun

data lost
Can we reduce concurrency? How much?

(8 states)

(6 states)

(5 states)

(4 states)

Asynchronous Control Circuit Design - L3
fully decoupled (Furber & Day)

semi-decoupled (Furber & Day)

GasP, IPCMOS

de-synchronization model

non-overlapping
Asynchronous Control Circuit Design

(semi-decoupled 4-phase protocol)
Asynchronous Control Circuit Design

(semi-decoupled 4-phase protocol)
Asynchronous Control Circuit Design - L3

(semi-decoupled 4-phase protocol)
(semi-découpled 4-phase protocol)
Which protocols are valid/fast/effective for de-synchronization?

Asynchronous Control Circuit Design

L3
Theorem: the de-synchronization protocol preserves flow-equivalence

Proof: by induction on the length of the traces

Induction hypothesis: same latch values at reset
Induction step: same values at cycle $i \Rightarrow$ same values at cycle $i+1$
Theorem: any reduction in concurrency preserves flow-equivalence

Any hybrid approach preserves flow-equivalence!
Flow-equivalence is preserved, ... but ...

- **semi-decoupled**
- **non-overlapping**
- **fully decoupled**
Live-ness?

- Preservation of flow-equivalence:
  
  *all the generated traces are equivalent*

- Are all traces generated?
  (Is the marked graph live?)

  *Not always!*

---

Semi-decoupled 4-phase handshake protocol

Liveness: all cycles have at least one token [Commoner 1971]
Results regarding Live-ness

- At least three latches in a ring are required with only one data token circulating [Muller 1962]

- **Theorem:**
  any hybrid combination of protocols is live if the simple 4-phase protocol is not used

  **Proof:** any cycle has at least one token
Asynchronous Control Circuit Design - L3

Valid for de-synchronization

decoupled (Furber & Day)

non-overlapping

semi-decoupled (Furber & Day)

fully decoupled (Furber & Day)

Asynchronous Control Circuit Design - L3

Initial state: E1, A1, R1, Ro = 0

Initial state: A0, O1, R0, Ro = 0
De-synchronization on FPGA
ASPIDA FPGA Implementation

- Xilinx Spartan IIE FPGA on a Diligent 2DE board
- FPGA contained:
  - De-synchronized DLX,
  - Processor memories
  - VGA driver
- Implemented Xilinx ISE
- Technology-portable Verilog design
- The full integer ISA and interrupt support is included
- DLX runs the “Game of Life” Algorithm
  - Fully-asynchronous
- VGA is fully synchronous


De-synchronized DLX on FPGA
ASPIDA ASIC Design

ASPIDA IC

- 32-bit RISC CPU
- EU funded research project
- Two fully-functional ICs were manufactured with IHP 0.25um technology
- Runs in both synchronous and de-synchronized modes,
  - Direct comparison of results
- Measurements
  - Performance, Voltage scaling and EME measurements
  - On-tester functional tests
  - Lab analysis
ASPIDA IC Testing

ASPIDA PCB Design

- 32-bit RISC CPU with DFV and adaptive (P, V, T) timing
- DFV embedded automatically to RISC CPU Verilog netlist using NanoSync V0 tool
- 700K Transistor Design, 0.25um CMOS process
- Full-scan testable, Adaptive Timing operation
- DFV Voltage Scaling from 3.3V down to 0.95V (2.5V process nominal)
- DFV Speed Scaling period from 18ns cycle @ 2.5V to 4,000ns @ 0.95V
ASPIDA TEM Cell Measurements

ASPIDA Results
ASPIDA Measurements

- **Performance results**
  - Synchronous: 52 MHz
  - De-Synchronized: all chips worked above 63 MHz
  - ~20% Lower-power through Voltage Scaling!

- **Power results**
  - Scaling: 200 mW (50 MHz) dropped to 98 mW (30 MHz)
  - Chips could scale down to 0.95 V (250 KHz), well beyond allowed voltage

- **EME reduction**
  - 30 dB (average)
  - 50 dB (max)
  - EME measurements were done using a TEM cell to guarantee accuracy

ASPIDA Schmoo Plot

- **Period vs. VDD for desynchronized operation**
- Fully functional on any voltage above 0.95 V
ASPIDA EME Results

Factorial

- sync
- async

ASPIDA running Factorial on TEM

Matrix

- sync
- async

ASPIDA running Matrix on TEM
Energy Spreading Effect

Asynchronous Control Circuit Design

Synchronous Pipelined MIPS

Asynchronous Control Circuit Design - L3
9/6/2016
Synchronous Design Flow and Study

STA only at $V_{dd}$ NOM (libraries calibrated) ➔ Iterative simulations for lower voltages

Synchronous Operation at 0.2V

$V_{dd} = 0.2V$

![Graph showing waveforms and current over time at 0.2V](image)
De-Synchronizing the MIPS

De-Synchronized Bundled-Data MIPS for sub-VT (uaMIPS)
Post Silicon Tunable Asymmetric Delay Line

- Asymmetric Delay Lines, for fast Return to Zero

- Post-Si delay line tuning: Mitigate process variation

Asynchronous Design Flow and Study

- RTL
  - Synthesis
  - STA (set DL)
  - De-Sync
  - SPICE NET

- LOWER V_{DD}
- Spice

WORKS?
  - no
  - yes
  - Save

DONE
Asynchronous Operation at 0.2V

\[ V_{dd} = 0.2V \]

- No clock

Asynchronous vs. Synchronous
uMIPS vs. uaMIPS Performance

uaMIPS Results

- Post-silicon tuning of delay lines:
  - No need to accommodate for process variations

ULP Embedded Processor Landscape
Conclusions

- Asynchronous is NOT a Religion!
  - Stop evangelizing goodness Axioms
  - It is NOT about asynchronous OR synchronous!
  - It is about clocking selectively!

- Need Pragmatic Design Approaches and Flows
- Need New EDA Tools
- Need New EDA Algorithms
- Don’t need new Library Cells for ASIC/SoC
- Don’t need new Silicon Architecture for FPGAs
- Killer Apps are here to stay; Understand them!
  - SoC synchronization
  - Low-Power
  - Variability