Asynchronous Circuits and Systems

“An architectural approach”

Marc Renaudin
TIMA Laboratory/CIS group
Marc.Renaudin@imag.fr
http://tima.imag.fr
http://tima.imag.fr/cis/

• Introduction
• Asynchronous circuits design principles
• Asynchronous circuit classes
• Asynchronous circuits’ architecture design
• Asynchronous circuits have very nice properties!
• TAST design flow
• Design experiments
• Conclusion and prospects
CIS group of TIMA laboratory

TIMA (http://tima.imag.fr)

About 120 people

Six research groups
- MNS : Micro and Nano Systems
- RMS : Reliable Mixed Signal Systems
- SLS : System Level Synthesis
- VDS : Verification and modeling of Digital Systems
- QLF : QuaLiFication of circuits
- CIS : Concurrent Integrated Systems

CIS group of TIMA laboratory

“CIS” Group (http://tima.imag.fr/cis)

About 20 people

Research topics
- Asynchronous circuits design and prototyping
- CAD Tools for Asynchronous circuits and systems
- Formal verification of asynchronous designs (Coll. with VDS D. Borrione)
- Hardware-software design for low power
- Secure circuit design for Smart-card applications
- SoCs and Smart devices design
- Mobile communication processors (Coll. with SLS A.A. Jerraya)
- Arithmetic operators
Concurrent Integrated Systems

Main results and partnerships

• Asynchronous Processors
  ASPRO (16 bit RISC)
  MICA (8 bit CISC)
  (STMicroelectronics, FT-R&D)

• Contactless Smart Card
  (FT-R&D, STMicroelectronics, Gemplus)

• Secure chip design (DES, AES)
  (SGDN/DCSSI, STMicroelectronics, Gemplus, Leti)

• TAL asynchronous std cell library
  (LIRMM)

• TAST framework (modeling, synthesis, validation)
  (CEA-LETI, STMicroelectronics, Technion, TUCS)

Asynchronous Circuits and Systems

• Introduction
• Asynchronous circuits design principles
  • Asynchronous circuit classes
  • Asynchronous circuits’ architecture design
  • Asynchronous circuits have very nice properties!
• TAST design flow
• Design experiments
• Conclusion and prospects
Asynchronous Circuits Design Principles

- Data-flow instead of control-flow

If rising_edge of clock then
    send output = f(inputs)
Else
    output remains unchanged
End if

Wait for inputs valid
    output = f(inputs)
Complete input transactions
Wait for output ready to receive
    send output
Complete output transaction

- At the scale of an individual hardware module

- every clock cycles
    trigger the computation

- data availability
    trigger the computation

→ Global Clock distribution replaced by local channels (handshaking)
Asynchronous Circuits Design Principles

• Composing hardware modules

• Synchronous circuits: balance the pipelines (worst case approach)

→ Circuit: add latency, increase power consumption
→ Design Meth: need to know the state of the whole architecture in each cycle
→ What happen if the system is very complex?
→ Difficult to exploit input data stream irregularities
Asynchronous Circuits Design Principles

• Asynchronous circuits: ensure data flows

\[
\begin{align*}
\text{Instr} & \quad \rightarrow \quad \text{Out} \\
\text{Op1} & \quad \text{Op2} \\
\end{align*}
\]

→ Circuit: latency is always minimum, as well as power consumption
→ Design Meth: no need to know the state of the whole architecture
   pipelining do preserve functional correctness
→ Easy to compose a complex system using simple modules
→ Free to exploit input data stream irregularities

Asynchronous Circuits Design Principles

Features of a basic asynchronous cell

- Bit level
- Arithmetic function
- Complex function

• Maximum speed: minimum forward latency
• Maximum throughput: minimum cycle time
• Respect the handshake protocol
→ design issues solved locally \(\Rightarrow\) cells are easy to reuse
Asynchronous Circuits Design Principles

Design of a FIR filter

The C-Element or Muller gate

Symbol

\[ C \]

Truth table

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Z \text{ or } Z \downarrow</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Z \downarrow</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ Z = XY + Z(X+Y) \]

- State holding
- Reset
Asynchronous Circuits Design Principles

- Protocol
  - Two phases
  - Four phases
- Signaling
  - Data encoding / Request
    - Three states
    - Four states
  - Acknowledgement

Two Phase Protocol

Data

Ack

Phase 1

Phase 2

Com. "n"

Com. "n + 1"
Asynchronous Circuits Design Principles

**Four Phase Protocol**

- Data
- Ack
- Phase 1
- Phase 2
- Phase 3
- Phase 4
- Com. "n"
- Com. "n+1"

=> Several derivatives exist

Asynchronous Circuits Design Principles

**Data Valid/Invalid Signaling**

- Three state coding
- Four state coding

Asynchronous Cell

Asynchronous Cell
Asynchronous Circuits Design Principles

Data encoding: Three states (dual rail)

Invalid state

00

01

Data bit = 0

Data bit = 1

ADD

Ack

D0

D1

Data

00 01 00 10 00 10

Ack

01

Data

00 01 00 10 00 10


Asynchronous Circuits Design Principles

Data encoding: Four states (dual rail)

00

01

10

11

Data = 0

Parity: Even

Data = 0

Parity: Odd

Data = 1

Parity: Odd

Data = 1

Parity: Even

ADD

Ack

D0

D1

Data

00 01 11 10 11 10

Ack

01 11 10 10 11 10

10
Asynchronous Circuits Design Principles

Completion Signal Generation

- Internal clock
- Use of a delay model
- Current sensing
- Use the data encoding

Asynchronous Cell

Invalid Data Valid Data Invalid Data

Acknowledge

Asynchronous Circuits Design Principles

Completion Signal Generation

- Using data encoding (dual rail)

Three state encoding

Output

S0 S1

OR Ack

Acknowledge

Invalid Data Valid Data Invalid Data

Four state encoding

Output

S0 S1

XOR Ack

Acknowledge

Invalid Data Valid Data Invalid Data
Asynchronous Circuits Design Principles

• Conclusion
  – Asynchronous circuits communicate using an handshaking protocol
  – Data/Request have to be encoded
  – A completion signal is required

→ The implementation may be delay insensitive
→ Hazard free logic is required
→ Hardware overhead ?

Asynchronous Circuits and Systems

• Introduction
• Asynchronous circuits design principles
• Asynchronous circuits’ architecture design
• Asynchronous circuit classes
  • Asynchronous circuits have very nice properties !
  • TAST design flow
  • Design experiments
  • Conclusion and prospects
Asynchronous circuit classes

• Synchronous circuits
  - Time is discrete
    → combinational logic is simple (hazards ignored)
    → trivial communication mechanism
    → worst case approach

  Data → Reg → Log. → Reg → Log. → Reg → Clock

  Global clock
  ⇒ Global timing assumption

Asynchronous circuit classes

• Asynchronous circuits
  - Delay insensitive circuits
  - Quasi delay insensitive circuits
  - Speed independent circuits
  - Huffman / Burst-mode circuits
  - Micropipeline
  - ...
  - Synchronous

  Hazard Free Logic → Reg. → Hazard Free Logic

  No global clock = no global timing assumption
  Sequencing is based on Handshaking
  ⇒ Hazard free logic is required

  Robustness & Complexity are decreasing:
  more timing assumptions
Asynchronous circuit classes

- Hazard free logic
  - QDI / Speed Independent Circuits
    - Data path: a dual-rail OR Gate
    - Sequencing: the Q-Element
  - Bounded delay circuits
    - Huffman circuits / Burst mode circuits
    - Sequencing: the Q-Element
  - Micropipeline circuits

Hazard free logic required

- Static hazards
- Dynamic hazards
- Combinatorial hazards
- Functional hazards
- Sequential hazards
Hazards : introduction (1)

- Hazard = spurious signal change
- Hazards appear during processing
  - it is then related to signal dynamics and component delays (wires and gates)

- Avoiding hazard implies:
  - to characterize the interaction between the circuit and its environment
   (define assumptions)
  - to characterize wire and gate delays
   (define assumptions)

- Static hazard 0
- Static hazard 1
- Dynamic hazard 0
- Dynamic hazard 1

No hazard  With hazard

Hazards : introduction (2)

Avoiding hazard is a Karnaugh Map covering problem!

1 to 1: the transition must be covered
1 to 0 and 0 to 1: avoid activation and deactivation of a minterm
0 to 0: no hazard in an AndOr circuit.

The covering problem may not have a solution if there are several MIC transitions. Therefore a delay insensitive implementation may no exist!

➢ Synthesis is different!
Asynchronous circuit classes

• Hazard free logic
• QDI / Speed Independent Circuits
  – Data path: a dual-rail OR Gate
  – Sequencing: the Q-Element
• Bounded delay circuits
  – Huffman circuits / Burst mode circuits
  – Sequencing: the Q-Element
• Micropipeline circuits

Asynchronous circuit classes

QDI Asynchronous circuits

• Functionally correct without any assumption on the wire and gate delays
  (unbounded delay model) except...
• "Isochronic fork" timing assumption

→ High level of robustness (with respect to delay variations)
Asynchronous circuit classes

Speed Independent Asynchronous circuits

- Functionally correct whatever the delays in the gates (unbounded delay model)
- The wires are assumed to be zero delay
- $\Rightarrow$ all wires are required to respect the isochronic fork property

$\Rightarrow$ Less accurate than the QDI model

QDI/SI asynchronous circuits

- Quasi Delay Insensitive & Speed Independent :
  $\Rightarrow$ hazard free control logic & hazard free data-paths

No timing assumption

- time is no longer discrete
  $\Rightarrow$ hazard free combinatorial logic
  $\Rightarrow$ handshake based communications
  $\Rightarrow$ mean time approach
QDI/SI asynchronous circuits

• Implementing delay insensitivity: examples
  – Choice of a communication protocol (request - acknowledge)
  - 1 bit Channel
    – Data encoding
      | Data | ai0 | ai1 |
      | 0    | 1   | 0   |
      | 1    | 0   | 1   |
      | Invalid | 0   | 0   |
    - 4 phase protocol

QDI/SI asynchronous circuits

• An example: dual-rail OR Gate
  \[ \begin{align*}
    & a_0, b_0, c_i \\
    & a_0, b_0, c_0 \\
    & a_1, b_1, c_0 \\
    & a_1, b_0, c_1 \\
    & a_0, b_1, c_0 \\
    & a_1, b_1, c_1 \\
  \end{align*} \]

Implement both the function and the protocol
QDI/SI Asynchronous circuits

• An example: the Q-Element

Asynchronous circuit classes

• Hazard free logic
• QDI / Speed Independent Circuits
  – Data path: a simple OR Gate
  – Sequencing: the Q-Element
• Bounded delay circuits
  – Huffman circuits / Burst mode circuits
  – Sequencing: the Q-Element
• Micropipeline circuits
Huffman/Burst-mode asynchronous circuits

- The correctness depends on the gate/wire delays
- Based on the "bounded delay" model
- "Fundamental mode" assumption for the environment

An example: The Q-Element

Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr

Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr

Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr

Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr+ → Ra+ → Lr → Ra → Lr → La+ → Lr
Timing assumptions

Asynchronous circuit classes

- Hazard free logic
- QDI / Speed Independent Circuits
  - Data path: a simple OR Gate
  - Sequencing: the Q-Element
- Bounded delay circuits
  - Huffman circuits / Burst mode circuits
  - Sequencing: the Q-Element
- Micropipeline circuits
Micropipeline asynchronous circuits

• Micropipeline

  - time is discrete
  - combinatorial logic is simple
  - communication channels (handshake based)
  - worst case approach locally

  \[ \text{Local timing assumptions} \]

Micropipeline asynchronous circuits

• An example: function \( F \)
  hazard free control logic + standard data path

  \[ \text{Areq} \quad \text{Breq} \quad \text{A} \quad \text{B} \quad \text{Aack} \quad \text{Back} \quad \text{F}(A,B) \quad \text{Reg} \quad \text{Delay} \quad \text{C} \quad \text{Creq} \]

  \[ \text{C} \quad \text{Delay} \quad \text{Cack} \]
Asynchronous circuit classes

• Conclusion
  – QDI / Speed Independent circuits are the most robust with respect to delays (isochronic fork for some/all wires)
  – Huffman & Burst-Mode circuits use the bounded delay model and require fundamental mode

→ QDI : Data-paths & Controllers
→ Speed Independent / Burst-Mode : Controllers
  (burst-mode controllers are difficult to compose)
→ Micropipeline : Standard data-path + QDI/SI Controllers

Asynchronous Circuits and Systems

• Introduction
• Asynchronous circuits design principles
• Asynchronous circuit classes
• Asynchronous circuits’ architecture design
  – Token game
  – Pipeline and ring optimization
  – Design example : ASPRO 16-bit RISC Asynchronous microprocessor
• Asynchronous circuits have very nice properties!
• TAST design flow
• Design experiments
• Conclusion and prospects
Asynchronous circuits’ architecture design

• Token game!

A token is carrying information
It is stored in a memory element.
It is represented by a filled circle next to the memory element it is stored in.

When using a four-phase protocol, the asynchronous data-path processes a stream of alternating valid and invalid (return to zero) token.
When a two-phase protocol is used, there are only valid tokens, but apart from that everything is the same.
Asynchronous circuits’ architecture design

• Token game!

A data flow of information is controlled by two rules:

• Token rule: a memory may receive and store a new token (valid or invalid) from its predecessor if and only if it has a bubble.

• Bubble rule: a memory becomes empty (bubble) if and only if its successor has received and stored the token that it was holding.

Asynchronous circuits’ architecture design

• Token game! (Join)
Asynchronous circuits’ architecture design

• Token game ! (Fork)

Asynchronous circuits’ architecture design

• Token game ! (FSM)
Asynchronous circuits’ architecture design

• Token game ! (FSM)

Asynchronous Circuits and Systems

• Introduction
• Asynchronous circuits design principles
• Asynchronous circuits’ architecture design
  – Token game
  – Pipeline and ring optimization
  – Design example : ASPRO 16-bit RISC Asynchronous microprocessor
• Asynchronous circuits have very nice properties !
• TAST design flow
• Design experiments
• Conclusion and prospects
Pipeline and rings

- Definitions
- Half-buffer / Full-buffer
- Pipeline behavior: throughput / latency
- Ring behavior and throughput optimization

Parameters definition (from Ted. Williams)

- pipeline and ring

\[ G : \text{total number of functional stages required to compute a given function} \]
\[ N : \text{number of stages in pipeline/ring} \]
\[ K : \text{number of tokens in pipeline/ring} \]
\[ S : \text{spread between statically packed tokens} \]
\[ L_f : \text{per-stage forward latency of tokens} \]
\[ L_r : \text{per-stage reverse latency of bubbles} \]
\[ P : \text{local minimum cycle time of stages} \]
\[ \lambda : \text{total latency of pipeline/ring} \]
\[ T : \text{throughput of pipeline/ring} \]
A half-buffer stage

Data | ai | aib
---|---|---
1   | 1  | 0  
0   | 0  | 1  
Invalid | 0 | 0

Lf = t (C2)

Lr = t (C2) + t (Nor2)

P = 2 (Lf + Lr)
A full-buffer stage

\[ L_f = t(C2) \]
\[ L_r = t(C2) + t(Nor2) \]
\[ P = 2(L_f + L_r) \]

Assuming \( t(C2) = 2\ tu \) and \( (Nor2) = 1\ tu \)
\[ => L_f = 2\ tu \]
\[ L_r = 3\ tu \]
\[ and \ P = 10\ tu \]

Throughput / Latency of a linear pipeline

\[ 5 \times L_f = 10\ t = P \]

\[ P <> L_f => T <> 1/L_f \]

Fold here to build a ring (with one token)
Rings

Optimized number of stages

Minimum number of stages

Ring behavior

- Total Latency = $\lambda = N \cdot Lf$
- Ring Throughput:
  $T = K / \lambda = K / N \cdot Lf$
  $T_{max} = 1 / P \iff N = K \cdot P / Lf$
- Behaves like a combinatorial structure when $N > K \cdot P / Lf$
Throughput optimization

If number of stages $N < 2K + 1$ => Deadlock
If number of stages $N > K \frac{P}{L_f}$ => Data limited
If number of stages $2K < N < K \frac{P}{L_f}$ => Bubble limited

Asynchronous Circuits and Systems

- Introduction
- Asynchronous circuits design principles
- Asynchronous circuits’ architecture design
  - Token game
  - Pipeline and ring optimization
  - Design example: ASPRO 16-bit RISC Asynchronous microprocessor
- Asynchronous circuits have very nice properties!
- TAST design flow
- Design experiments
- Conclusion and prospects
ASPRO: a RISC Microprocessor

- A QDI Asynchronous 16 Bit RISC Microprocessor

=> Performance
=> Standard-Cells
=> Ease of design (4 m.y)
  - Architecture
    (out of order completion)
  - System Board

ASPRO

- QDI asynchronous logic
- Standard Cells
- 500 KTr for the core
- 6.3 MTr with memories
- Total area is 42 mm²
- CMOS 0.25µm 6 metal layers
  STMicroelectronics

- Use of standard tools except
  - CHP2VHDL translator
  - Synthesis
ASPRO

- Functional at first silicon: 0.65V and 3.0V
- 140 MIPS (max)
- ASPRO includes DSP capabilities (MACC unit, brr...),
  RIF routine runs at 115 MIPS, 500 mW (including memories)
- Serial links (2 phase): 50Mbit/s

ASPRO's instruction set

Alu instructions: std arithmetic, logic and shift/rotate
- min/max, bit reverse, slt/sltu (no status register)

Load/Store instructions: byte/word load and store
- basic addressing mode is indirect with displacement
- immediate load is provided (16 bit values)
- load relative address (relocatable code)
- program memory load/store through a dedicated register (boot)

Program flow instructions
- conditional relative branch (eq,ne,lt,ltu) => delayed or not
- djmp, dbsr, djsr => delayed

Custom instructions: 64 slots
- mpy, macc (integer, fixed point, rounding and saturation modes)
  (16 x 16) + 40 => 40 bit
ASPRO: a RISC Microprocessor

- ASPRO
- Flash memories
- Reset/Interrut logic
- Peripherals: 2Φ Serial Links & Parallel Ports

Daughter board

Switches & LEDs

Interrupt
Reset

Mother board

ASPRO

- Processor’s main features
- Memories
  - program: 16 kwords on chip
  - 48 kwords off chip
  - data: 64 kbytes on chip
- 16 general purpose registers
- On-chip peripherals
  - 2 parallel ports
  - 4 bidirectional serial links (2Φ)
- Custom units
  - added to the peripheral area
  - embedded in the data path
- Three supply voltages
- Interrupt mechanism

 Daughter board

Switches & LEDs

Interrupt
Reset

Mother board
ASPRO

• ASPRO’s architecture

Four main loops:

  . Fetch/Decode loop
  . Data-Path loop
  . Branch loop
  . Load/Store PM loop

The Fetch/Decode loop

- 3 tokens in the loop
  (2 delay slots)
- 21 half buffer stages
  \( P/Lf = 7 \)

Timing goals:
- 15 ns latency
- 5 ns cycle time

=> Memory architecture

(latency / throughput)
ASPRO

Memory architecture
- minimum latency
- high throughput

The Data-Path loop
- The latency depends on the unit and the data involved (3 to 14ns)
  (=> insertion of 0 to 2 instr. for ASPRO to run at full speed)
- 5 ns cycle time for Reg-File and Bus-Interface
  -> Can be more for the data-path units
  => Register-file Architecture
ASPRO

• In-order Issue / Out-of-order Completion
  - Two read ports
  - Four write ports (no arbitration, out-of-order write-back)
  - Locking mechanism
    - a register is locked as long as a write is pending
    - several registers may be locked, while other registers can still be accessed

=> Solves - read after write hazards
  - write after write hazards

ASPRO

• Register-File architecture

Op1
Op2
Dest
Unit
Fifos

Reg-File

From Data-Path Units

To Data-Path
The Branch loop

- Bcc, Dbcc regi, regj, Offset
- Djmp regi
- Djsr regi, regj
- Dbsr regi, Offset
- Lra regi, Offset

=> Optimize latency

The Load/Store Program Memory loop

Ldpg, Stpg instructions
No arbitration:
  inserts an extra token in the fetch/decode loop

- Latency balancing
- Following instructions may pass beyond in the Ld/St unit
ASPRO

• Code optimization: an example

FIR:  MAc  ACC0, R0, R1
     Ldw  R0, (R10)
     Ldw  R1, (R11)
     Addi R10, R10, #1
     Dbne R7, #0, FIR:
     Addi R11, R11, #1
     Subi R7, R7, #1

Asynchronous circuits’ architecture design

• Conclusion
  – Token game
  – Pipeline optimization
  – Architectural features
    • Elastic pipeline
      – Variable number of data
      – This number can change dynamically
    • The latency is not 1/Throughput
      – Interleaving or overlapping is very efficient
Asynchronous Circuits and Systems

- Introduction
- Asynchronous circuits design principles
- Asynchronous circuit classes
- Asynchronous circuits’ architecture design
- Asynchronous circuits have very nice properties!
- TAST design flow
- Design experiments
- Conclusion and prospects

Systems design requirements

- Assemble / Reuse new and existing modules
- Modules with very different architectures
- Modules activity may be very different
- Modules with different speed/power trade-offs
- Flexible on-chip communication mechanisms
- Low power
- Mix digital and Analog modules

→ Modularity and locality => reusability
   → Get rid of global constraints (like a unique global clock)
   → Avoid inheritance of constraints from block to block (like clock, noise, communication and synchronization mechanisms…)
   → Do not consume when not in use
Modularity

- Methodology
  - functional dependencies specification
  - Separated design of
    - the function and the architecture
      - pipelining preserves functional correctness
      - performance optimization is performed independently of the functional correctness

\[ P_0 = \begin{array}{c}
\vdots \\
\text{A ? x} \\
\vdots \\
\end{array} \]
\[ P_1 = \begin{array}{c}
\vdots \\
\text{A ! y} \\
\vdots \\
\end{array} \]

Correct Transformations

Modularity

- Circuit

\[ \begin{array}{c}
D, \text{Ctrl} \\
\end{array} \]
\[ \begin{array}{c}
P \text{ stages} \\
\end{array} \]

=> Locality: global state knowledge is unnecessary
**Power Consumption and Speed**

Joint optimization of
- average case
- most probable case

Synthesis is using the probabilities
- => Finite State Machines
- => Data Paths

**Average throughput is increased without penalty on the latency**

=> Throughput-Latency are decoupled
Current consumption is very different

<table>
<thead>
<tr>
<th>Synchronous</th>
<th>Asynchronous</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Clock driven&quot;</td>
<td>Data driven</td>
<td>Mean power consumption is lower</td>
</tr>
<tr>
<td>&quot;Global clock&quot;</td>
<td>Distributed control</td>
<td>Smaller current peaks EM emission is lower</td>
</tr>
<tr>
<td>&quot;Clock timed&quot;</td>
<td>Self timed</td>
<td>Automatic performance regulation</td>
</tr>
</tbody>
</table>

Therefore:

- One can reduce noise
- One can design secure chips
- One can design system with an automatic performance regulation.
Noise

Measurements performed with "MICA" an asynchronous QDI 8-bit microcontroller
Current profile

- 10 mA average current
- 0.6 mA amplitude variations

Mean power consumption is lower
Smaller current peaks, EM emission is lower

Synchronous circuit

Noise

Measurements performed with "MICA" an asynchronous QDI 8-bit microcontroller
Current spectrum

Max = 0.06

0 to 12.5 GHz

0 to 0.5 GHz
Noise

- Data Encryption Standard (DES) crypto-processors

Asynchronous DES Chip architecture

Circuits’ Layout

Synchronous Asynchronous

CMOS 0.18 µm (HCMOS8) from
STMicroelectronics
6-LM, Power Supply 1.8 V

Results and Analysis

Current profile of circuits

Asynchronous

- Current profile is smoothed
- Peak variations of 1 mA

Pads activities

Current variations of asynchronous core

- High and fast current peaks caused by clock signal (3mA; 10 ns)
- Number of current peaks variations

Synchronous

Clock activities

- Number of current peaks variations

- Number of clock activities

Marc Renaudin
Ecole Archi’05 – 25 Mars 2005
Results and Analysis

Current Spectrum of circuits

Asynchronous
- Frequency of computing each DES iterations
- Band limited (about 45 MHz)
- Band-width of power signal is fixed.
  * Spectrum/noise characteristics can be fitted to the application requirements.
  * Filters can be designed to reduce emissions.

Synchronous
- Synchronous circuit exhibits significant peaks up to 300 MHz.

Security

- MEDEA+ ESP@SS-IS project (A302)
  Enhanced Smartcard Platform for Accessing Securely Services of the Information Society
  - Goals:
    - provide an innovative design technique and associated CAD tools to improve security and power consumption
    → uncorrelated data processing and electric-magnetic observations
  - Contributions:
    - study asynchronous logic to figure out the best asynchronous circuit style to improve security (signature)
    - develop the corresponding CAD tools to generate secure circuits against timing and power analysis attacks
Security

• Galois-field Multiplier

Architecture and cells are designed so that input data are processed using an equal number of electrical transitions.

Security

• Processor MICA

- Number of points : 100000
- 10000 computations
DPA : Design flow

- Specification
- Synthesis (DPA resistant option)
  - Gate Netlist
- Place and Route (constrained)
- Extraction
  - Formal Model of the Circuit
  - Balanced electrical paths?
    - Yes
    - No
- Balanced logical paths?
  - Yes
  - No
  - OK!

Tamper-resistant hardware

- Hardware cryptanalysis (measurements, post-processings)
- Counter measures design using ASL
  - PA
  - DFA
- Prototyping ASL
  - ASIC: DES, AES
  - FPGAs
- Methodologies and CAD tools for improving PA and DFA chip resistance using ASL
Automatic performance regulation

- Computation-power controlled systems: \( E = a.fCV^2 \)

\[ \begin{align*}
\text{Asynchronous System} & \quad \text{DC/DC Converter} \\
\text{Inputs} & \quad \text{Control} \\
\text{FIFO} & \quad \text{Processing} \\
\text{FIFO} & \quad \text{Outputs}
\end{align*} \]

- Dynamic regulation of the power supply with respect to the processing power required.
  
  (Philips Research, DCC)

\[ \rightarrow \text{Minimum energy computation} \]

Exploit:
- Processing and data have irregular nature
- A breakdown with respect to the synchronous approach

- Power supply controlled systems

- Processing power is limited by the power budget available

\[ \rightarrow \text{Maximum performance delivered with the available power received} \]

Applications:
- Remotely powered systems (RFIDs)

\[ \begin{align*}
\text{Voltage} \\
\text{Current}
\end{align*} \]

\[ \text{Activity} = \text{processing power} \]
Asynchronous Circuits and Systems

- Introduction
- Asynchronous circuits design principles
- Asynchronous circuit classes
- Asynchronous circuits’ architecture design
- Asynchronous circuits have very nice properties!
- TAST design flow
- Design experiments
- Conclusion and prospects

TAST*: TIMA ASYNCHRONOUS SYNTHESIS TOOLS

- Low Power
- Low Voltage
- Low Noise

Verification
- Collaboration with VDS

Simulation
- VHDL
- C

Synthesis
- QDI
- Micropipeline

Rapid Prototyping
- FPGAs (Xilinx, …)

Test

TAST Specification Language

TAST Compiler

TAST Intermediate Format

ASPRO

Current Profiles (MICA)

GALS, GSLA SoCs

Secure Smart-Cards

Smart Devices

*http://tima.imag.fr/cis
Motivations and objectives

- High Level of Abstraction
  - CHP
  - Petri Net
- High Level of Automation
  - Multi-target Compiler
    - Simulation
    - Synthesis
    - Verification
  - Interface with Standard Tools
  - Ease of Use
- Co-Design Platform
  - Hierarchy
  - Micropipeline, QDI, Synchronous and, Behavioral Modules
  - Design Reuse

Compiler => Intermediate Format

- Contextual Tree
- Petri Net

Transformations

Back-end
- Simulation
- Synthesis
- Verification
C Simulator

Support non determinism!

TAST : Status

- TAST is running on
  - Solaris
  - Linux

- TAST is compatible with VHDL env. From Synopsys & Mentor

- TAST includes
  - Data-base management
  - Graphical User Interface

- Used within the CIS group and by some partners
TAST : Conclusion

• What to keep in mind?
  – Weakness
    • Testing
  – In progress
    • Logic Optimization
    • Technology mapping
  – Strengths
    • High Level of Abstraction
    • High Level of Automation
    • 1 out of N Arithmetic
    • Co-Design Framework
    • Prototyping using FPGAs
    • Address VDSM SoCs

• Conclusion
  We are very close to offer a set of CAD Tools to design efficient
  Asynchronous VLSI Circuits and allow designers to make the Best Trade-
  offs for their Applications.

A lot of C code !…

Asynchronous Circuits and Systems

• Introduction
• Asynchronous circuits design principles
• Asynchronous circuit classes
• Asynchronous circuits’ architecture design
• Asynchronous circuits have very nice properties !
• TAST design flow
  – DES crypto processor
  – Contact-less Smart Card
  – Asynchronous A-to-D converter and signal processing chain
  – Towards fully asynchronous systems
• Conclusion and prospects
DES crypto processor

- Technology: CMOS 0.18µm STMicro
- Total area: 0.426 mm² / 34650 gates
- Consumption: 6 mA
- Performances:
  - DES (1.8 V): 780 ns
  - TDES (1.8 V): 2240 ns
  - DES (1.25 V): 1560 ns
  - TDES (1.25 V): 4370 ns

PA and FA improved resistance

AES crypto processor

AES Chip
- Technology: CMOS 0.13 µm (HCMOS9) from STMicroelectronics
- Core area (mm²): 0.473
- Total area (mm²): 1.24

Current profile of AES Asynchronous core

Time versus Power supply
- Core area (mm²): 0.473
- Total area (mm²): 1.24
- AES Chip
  - Technology: CMOS 0.13 µm (HCMOS9) from STMicroelectronics
  - 6-LM, Power Supply 1.2 V

Current variations on the core
- Ciphering time

0.6 0.7 0.8 0.9 1 1.1 1.2

0 1 2 3 4 5

2 mA 8 mA 12 mA

Ciphering time

0 1 2 3 4

256 bits 192 bits 128 bits
Asynchronous Circuits and Systems

- Introduction
- Asynchronous circuits design principles
- Asynchronous circuit classes
- Asynchronous circuits’ architecture design
- Asynchronous circuits have very nice properties!
- TAST design flow
- Design experiments
  - DES crypto processor
  - Contact-less Smart Card
  - Asynchronous A-to-D converter and signal processing chain
  - Towards fully asynchronous systems
- Conclusion and prospects

Contactless Smart-Card Chip

- "SoC" for Contactless Smart-Card
  - Power reception system (on-chip coil)
  - ISO 14443-B std compliant
  - 8-bit CISC Asynchronous Microcontroller designed with standard cells (Mica)
  - Rom
  - Rams

Collaboration with France Telecom/R&D
Cmos 0.25 µm STMicroelectronics
[IEEE-JSSC July 2001]
Mica : an 8-bit CISC QDI Asynchronous µC

- 1-of-4 DI codes for arith. and reg.
- 1-of-n DI codes for the control
- Complexity
  - 145 000 transistors (0.25 µm)
  - 1 M transistors with memories
  - 13 mm² with pads (prototype)
  - PGA120 package for the prototype
- Test
  - BIST (approx. 300 instr)
  - functional at 1st silicon between 3V et 0.65 v
- 24 Mips / 28 mW@ 2.5V
- 4.3 Mips / 800 µW@ 1V

<table>
<thead>
<tr>
<th>Supply (V)</th>
<th>Mips</th>
<th>Core Current (mA)</th>
<th>Power(mW)</th>
<th>Mips/Watt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.3</td>
<td>0.8</td>
<td>0.8</td>
<td>5603.6</td>
</tr>
<tr>
<td>1.5</td>
<td>11.9</td>
<td>3.1</td>
<td>4.7</td>
<td>2560.2</td>
</tr>
<tr>
<td>2</td>
<td>18.6</td>
<td>6.7</td>
<td>13.3</td>
<td>1398.0</td>
</tr>
<tr>
<td>2.5</td>
<td>23.8</td>
<td>11.2</td>
<td>28.0</td>
<td>850.3</td>
</tr>
<tr>
<td>3</td>
<td>27.8</td>
<td>16.3</td>
<td>48.9</td>
<td>568.1</td>
</tr>
<tr>
<td>3.5</td>
<td>31.3</td>
<td>22.0</td>
<td>77.0</td>
<td>405.8</td>
</tr>
</tbody>
</table>

MICA : Architecture

- Memory
  - 16 Kbytes RAM
  - 2 Kbytes ROM

- Index registers
- Data registers
- Status register
- ALU 16-bits

- Write data
- Read data
- Smart card interface
- Flash memories
- Serial links
### MICA: Instruction Set

<table>
<thead>
<tr>
<th>Arithmetic/logic</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add, Adde, Sub, Subb, Inc, Dec, Neg</td>
<td>Add, Adde, Sub, Subb, Inc, Dec, Neg</td>
</tr>
<tr>
<td>And, Or, Xor, Not, Cho (clear bit i), Shn (set bit i), Thn (test bit i)</td>
<td>And, Or, Xor, Not, Cho (clear bit i), Shn (set bit i), Thn (test bit i)</td>
</tr>
<tr>
<td>Rol, Ror, Rel, Rcr (rotate without and with carry)</td>
<td>Rol, Ror, Rel, Rcr (rotate without and with carry)</td>
</tr>
<tr>
<td>Shl, Shr, Shr (shift left, shift right unsigned and signed)</td>
<td>Shl, Shr, Shr (shift left, shift right unsigned and signed)</td>
</tr>
<tr>
<td>Index</td>
<td>Index</td>
</tr>
<tr>
<td>Addx, Subx</td>
<td>Addx, Subx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load/Store</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ld, Ldp, Stp (load, store peripheral), St</td>
<td>Ld, Ldp, Stp (load, store peripheral), St</td>
</tr>
<tr>
<td>Cp (copy from memory to memory is available)</td>
<td>Cp (copy from memory to memory is available)</td>
</tr>
<tr>
<td>Pl (push &amp; load)</td>
<td>Pl (push &amp; load)</td>
</tr>
<tr>
<td>Psh, Pshr (push, push status register), Pop</td>
<td>Psh, Pshr (push, push status register), Pop</td>
</tr>
<tr>
<td>Index</td>
<td>Index</td>
</tr>
<tr>
<td>Pshx, Popx</td>
<td>Pshx, Popx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control flow</th>
<th>Rti, Rts, Jmp, JSr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bee, Burec (ce = a, eq, ne, ce, es, l, le, lt, ge, gt, le, ge, gt, ec, ec)</td>
<td>Bee, Burec (ce = a, eq, ne, ce, es, l, le, lt, ge, gt, le, ge, gt, ec, ec)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Misc</th>
<th>Nop, Wfi, Eint, Diint (enable and disable interrupt)</th>
</tr>
</thead>
</table>

### Contactless Smart-Card Chip

- Asynchronous Logic relaxed Design constraints
  - Not sensitive to supply voltage variations
    - Power reception system (capacitances area, voltage regulation)
  - Lower current peaks
    - The Micro-controller can be running during the communications without disturbing the load modulation.
  - Maximum processing power delivered according to the power received

Collaboration with France Telecom/R&D
Asynchronous Circuits and Systems

- Introduction
- Asynchronous circuits design principles
- Asynchronous circuit classes
- Asynchronous circuits’ architecture design
- Asynchronous circuits have very nice properties!
- TAST design flow
- Design experiments
  - DES crypto processor
  - Contact-less Smart Card
  - Asynchronous A-to-D converter and signal processing chain
  - Towards fully asynchronous systems
- Conclusion and prospects

Scope of this work

- Context: Integrated smart Devices & Communicating Objects
  - Power consumption reduction by more than one order of magnitude
- Solution: Re-think the whole processing chain
  - Systems only driven by the signal information
    - Asynchronous design (without any global clock)
    - Irregular sampling
Irregular sampling

Regular sampling

Irregular sampling

- Respect the Shannon theorem
- Instants exactly known
- Information: $T_{sample}$, $\{i_k\}$
- In an ADC: Amplitude quantization
- Many useless samples

- “Level-crossing sampling”
- Amplitudes exactly known
- Information: quantum, $\{dt_k\}$
- In an A-ADC: Time quantization
- Only useful samples

Asynchronous A-to-D converter chip

| Technology | CMOS 0.12µm 1.2V |
| Application | Speech signals |
| ENOB | Up to 10-bit |
| Hardware resolution | M=4-8 |  |
| Timer resolution | $M_{max}=12$-bit |
| $T_0$ | Up to 30kHz |
| Power supply | $V_T=2.5V$ |
| Voltage dynamic | $\delta=100nV$ |
| Voltage quantum | $q=40nV$ |
| Loop delay | $\tau=60ns$ |
| Input signal bandwidth | $f_{max}=1MHz$ |
| Power consumption of the A-ADC analog part | $P=158.4µW$ |
| Power consumption of the A-ADC digital part | $P=2.27µW$ |
| Total Power consumption of the A-ADC | $P_{total}=161.67µW$ |
| Analog area | $S_{analog}=350µm^2 \times 350µm^2$ |
| Digital area | $S_{digital}=296µm^2 \times 375µm^2$ |
Asynchronous Circuits and Systems

- Introduction
- Asynchronous circuits design principles
- Asynchronous circuit classes
- Asynchronous circuits’ architecture design
- Asynchronous circuits have very nice properties!
- TAST design flow
- Design experiments
  - DES crypto processor
  - Contact-less Smart Card
  - Asynchronous A-to-D converter and signal processing chain
  - Towards fully asynchronous systems
- Conclusion and prospects

Towards Fully Asynchronous Smart Devices

- Camera
- Image processing
- RF communication

→ Integration on a single chip
  - Modular
  - Event driven
  - Low power
  - Low noise

- Software?
Dynamic Voltage Scaling/Scheduling

Principle: control the processing power using the voltage (in the OS)

\[ P \propto C V^2 f \]

- Simulation and results

- Without Dynamic Voltage Scheduling:

- With Dynamic Voltage Scheduling:

Results: 60% power saving for an image processing application running on ASPRO

Comparison with synchronous processors

- StrongARM-1100
  - 59MHz at 0.79V and 251MHz at 1.65V
  - Frequency changing: 140\(\mu\)s
  - Voltage changing: 40\(\mu\)s
  - Standby mode: 90\(\mu\)s
  - Wake up: 160 ms

- lparm (ARM8 Berkeley)
  - 6Mips, 2.8mW at 5MHz - 1.2V and 85Mips, 460mW at 80MHz - 3.8V
  - Frequency is changing from 5MHz to 80MHz in about 70\(\mu\)s
  - Standby mode (idle): 0.5mW

- Transmeta Crusoe
  - 300MHz at 1.2V and 600 MHZ at 1.6V, 33 MHz step, 25 mV
  - 0.7 W at 300MHz and 6.8 W at 600MHz
  - Frequency is changing from 200 MHz to 700 MHZ in 300 \(\mu\)s
**Dynamic Voltage Scaling/Scheduling**

- Benefits of an asynchronous processor:
  - No software to manage the processor or the peripherals activity (run, idle, sleep)
  - Consumption in sleep mode = consumption in idle mode (ratio higher than 100 for the synchronous processors)
  - Wake up/stop at very high frequency for the processor and the peripherals (thousands of cycles for a synchronous processor)
  - Speed and energy controlled by the voltage only (switching time much smaller: factor of 2 to 5)
  - Large voltage range (more and more critical with the supply voltage decreasing)

**Asynchronous Circuits and Systems**

- Introduction
- Asynchronous circuits design principles
- Asynchronous circuit classes
- Asynchronous circuits’ architecture design
- Asynchronous circuits have very nice properties!
- TAST design flow
- Design experiments
- Conclusion and prospects
Conclusion and Prospects

- Technology trends
  - Delay variations due to:
    process, cross-talk, peak current, temperature gradient

- System integration trends
  - Low noise, low power required (mixed signal chips, RF)
  - High complexity on chip synchronization schemes
  - On chip networks design (NoC)
  - Heterogeneous architectures: modularity and reusability

Conclusion

- Channel-based / Data flow system design instead of clock-based
- Asynchronous circuits = event-driven instead of timing-driven
  → low power, low noise
  → modularity, locality, scalability => reusability
  → design time => time to market
- Fully asynchronous system is the future…?
  → Reduce the clock to a common resource of the system,
    only used to measure time
  → Software is simplified
  → Synergy between sensors – actuators - interfaces and processing