INTRODUCTION TO OCTASIC ASYNCHRONOUS PROCESSOR TECHNOLOGY

Async 2012 , Copenhagen, May 7-9 th 2012

Michel Laurence, Founder & CEO

michel.laurence@octasic.com

octasic semiconductor

CONTENTS

- Background
- Asynchronous Circuits Description
- Processor Architecture and Operation
- Performance Analysis
- Conclusion

CONTENTS

- **Background**
- Asynchronous Circuits Description
- Processor Architecture and Operation
- Performance Analysis
- Conclusion

BACKGROUND ON OCTASIC

- Founded in 1998
- Headquartered in Montreal, Canada
- 85 employees
- Evolution:
  - 98/00 - Design ASICs for others
  - 2001 Convert to fabless model
    - **2001- 2003: VoIP** Support Products **(Synchronous)**:
      - **2001 - Voice Packetization Engine / OCT8304**
      - **2003 - Echo Cancellation Processor / OCT6100**
    - **2004 – DSPs (Asynchronous)** for Voice, Video, and Wireless Baseband
      - **2008 - First Generation / OCT1010**
      - **2011 - Second Generation / OCT2224**
      - **…2013 - Third Generation / OCTXXXX**

GENESIS OF MOVE INTO ASYNC DESIGN

- **First Processor Product**
  - **Specialized DSP for Echo Cancellation**
    - Entered the echo market 20 year late
    - Success because of unique algorithm
- **Next Product – Generic DSP?**
  - **How to succeed?**
    - Settle on **highest processing efficiency** – **Processing Power / Power Consumption**
    - 2+X improvement needed to be able to succeed and displace incumbents
  - **This led us forfuitously into the asynchronous world**
    - Started by removing the clock – the single greediest power culprit in synchronous designs
    - … then tried to figure our how to make our circuits work
    - … proceeded by trial and error until

      …we arrived at our current async design and methodology

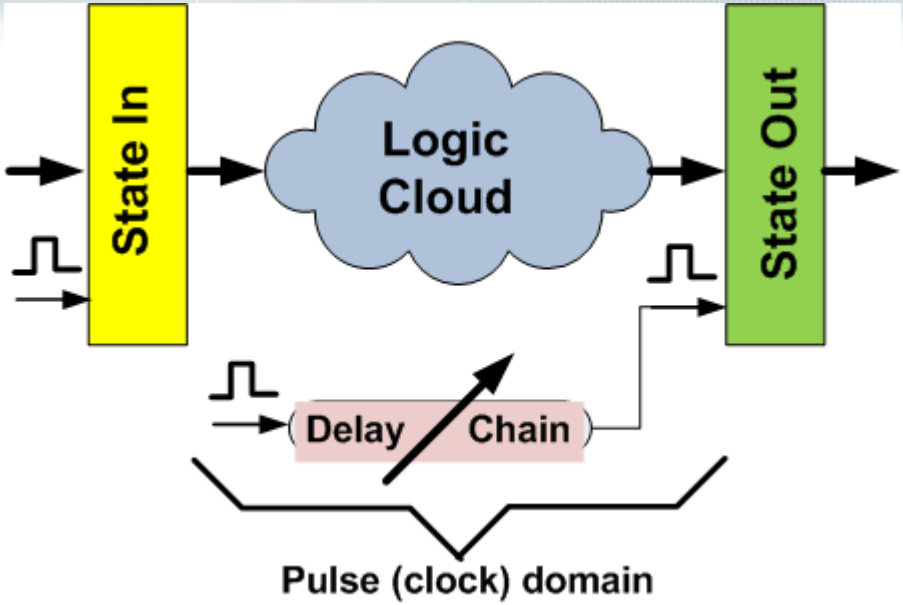
SET ADDITIONAL PRE-REQUIREMENTS

- **Use only standard ASIC library elements**
  - No custom cell
  - Ease of porting - from one silicon node to the next / from one vendor to another
- **Use (as much as possible) standard CAD tools and concepts**
  - To facilitate sign-off
  - To facilitate staff conversion training
- **Use an architecture presenting a traditional programming view**
  - S/W paradigm (same look and feel)
    - Avoid software programming model changes
      - Programming model change is an almost insurmountable barrier to product adoption
      - Allow re-use of existing S/W
      - Transparent to programmers
  - Similar single thread-performance
    - Avoid forcing to re-structure algorithms

CONTENTS

- Background
- **Asynchronous Circuits Description (Basic)**
- Processor Architecture and Operation
- Performance Analysis
- Conclusion

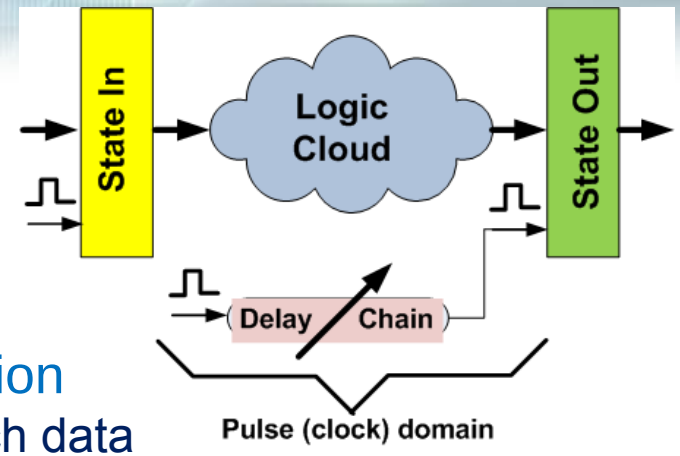
BASIC ASYNCHRONOUS CIRCUIT (1)



- **Logic Elements:** States In/Out, Logic Clouds, and Delay Chains
  - States are latches or flip-flops
  - Logic Clouds and delay chains use combinatorial logic
  - Delay chains are statically or dynamically controlled
- **Timing Elements**: Pulses
  - Pulses are asynchronous to each other and event (token) driven
  - Timing verification is performed via standard STA (Static Analysis Tools) Tools
    - on each pulse (clock) domain: Set-up and Hold-Time
    - each pulse (clock) domain is large (there are less than 20 domains in design)

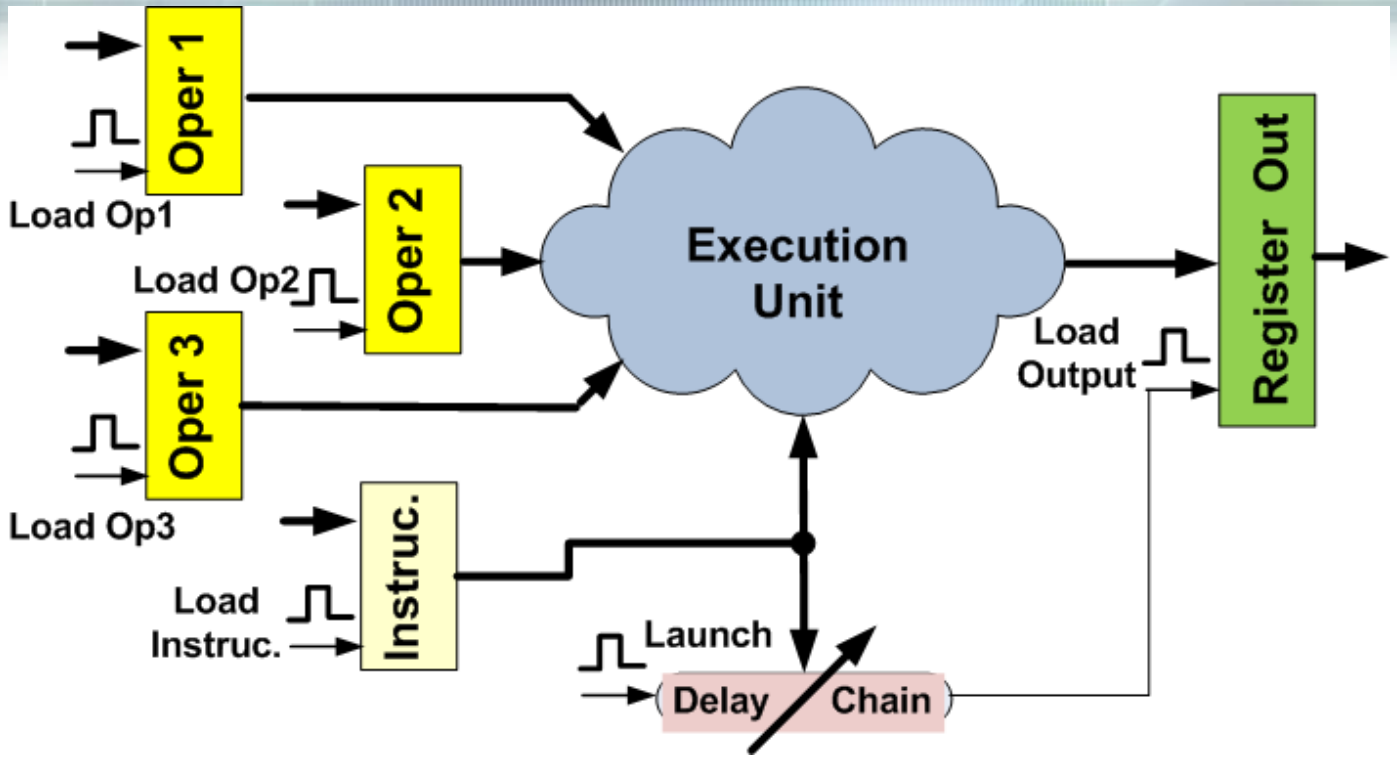
BASIC ASYNCHRONOUS CIRCUIT(2)

How does this maps into traditional classification of async circuits?



- Single-rail data bundled type for data transmission
  - With a worst-case delay "Bundling Signal" to latch data
- However no formal reverse ACK signal for flow control
  - Use a system of tokens to be described later
- Asynchronous Pipeline Structure: Static
  - Formal latches/FF to store data in between stages

SIMPLIFIED DSP EXECUTION UNIT

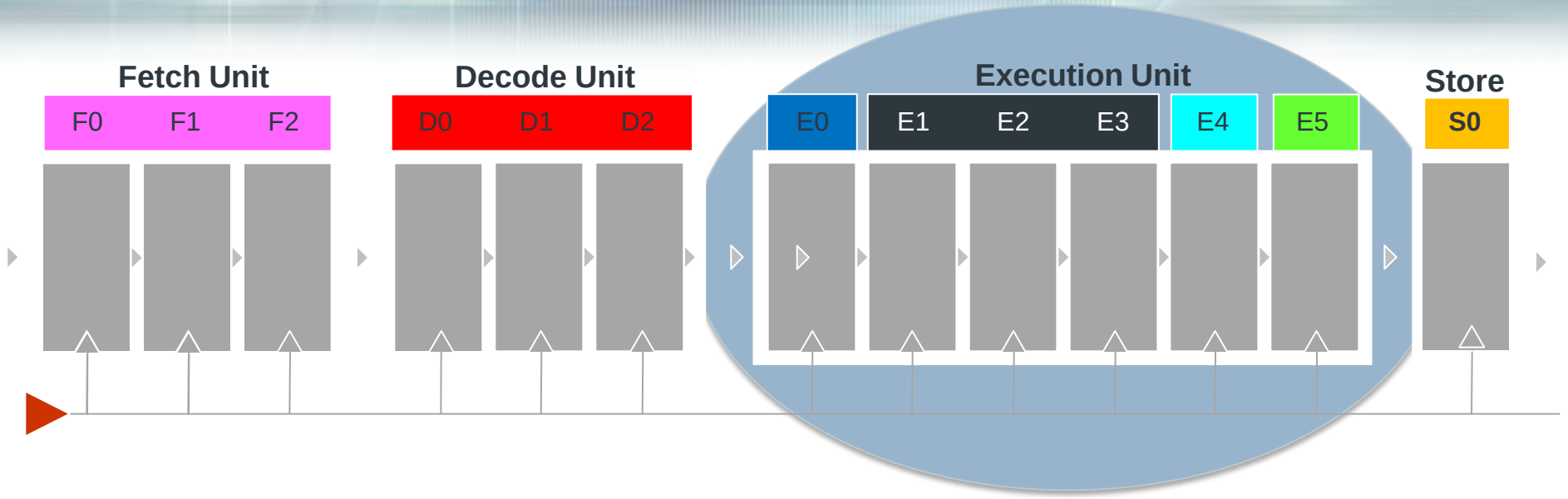


- The 3 operand state registers are asynchronously loaded
- The instruction state register is asynchronously loaded
- When ready (input registers loaded & output register released) a launch pulse is generated
- Delay chain timing is modulated according to instruction
- Output state register is asynchronously loaded with result of instruction

CONTENTS

- Background
- Asynchronous Circuits Description
- **Processor Architecture and Operation (Simplified)**
  - **Architecture**, Silicon, and ILP Implementation
  - Operation & Synchronization
- Performance Analysis
- Conclusion

SYNC VS ASYNC PROCESOR IMPLEMENTATION



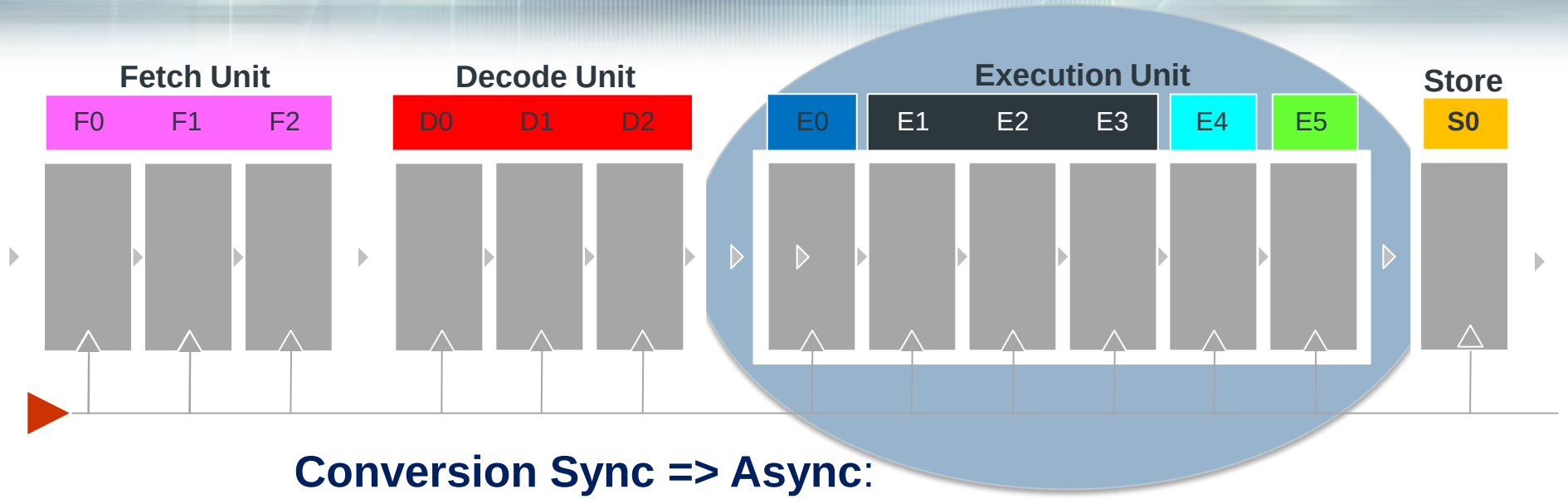
Fetch
Decode
Reg Reads
Execute
Branch
Output Write
Store

MEM load/store not show

In typical synchronous design, pipelining is used to boost performance and provide Instruction Level Parallelism (ILP)

How can we convert such synchronous design into an asynchronous one?

SYNC VS ASYNC PROCESOR IMPLEMENTATION



Conversion Sync => Async:

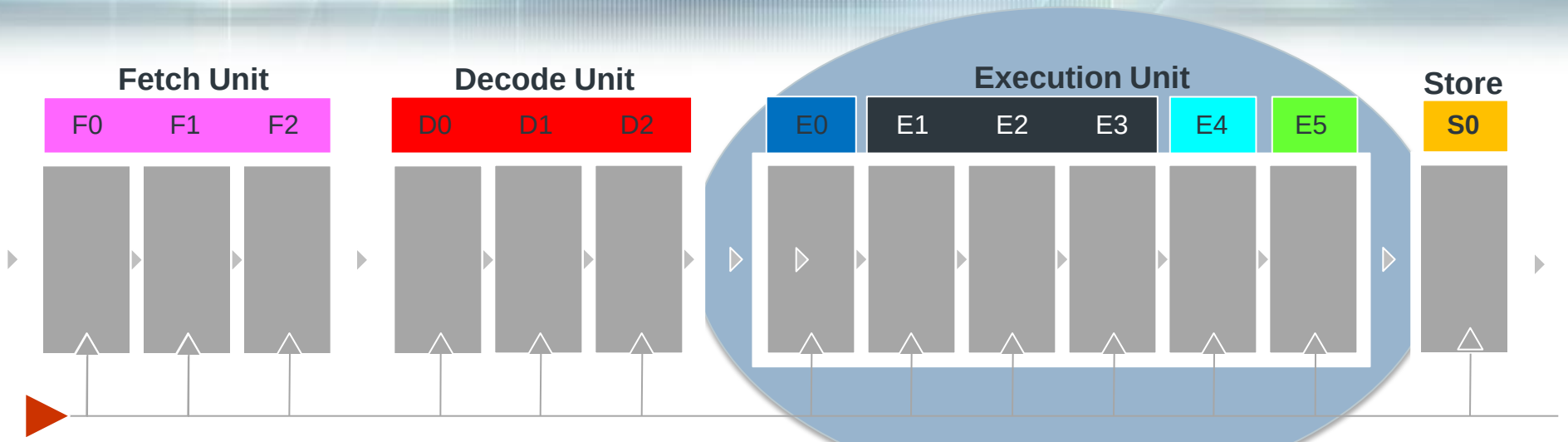
- One way is to map each **unit functionality** into an equivalent asynchronous unit

Fetch
Decode
Reg Reads
Execute
Branch
Output Write
Store

MEM load/store not show

octasic semiconductor

SYNC VS ASYNC PROCESOR IMPLEMENTATION

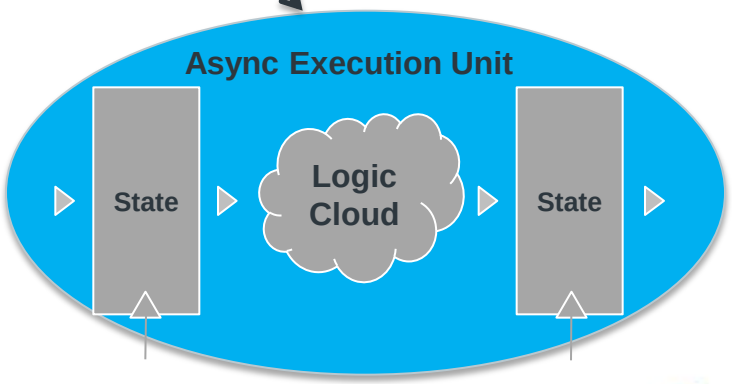


Fetch
Decode
Reg Reads
Execute
Branch
Output Write
Store

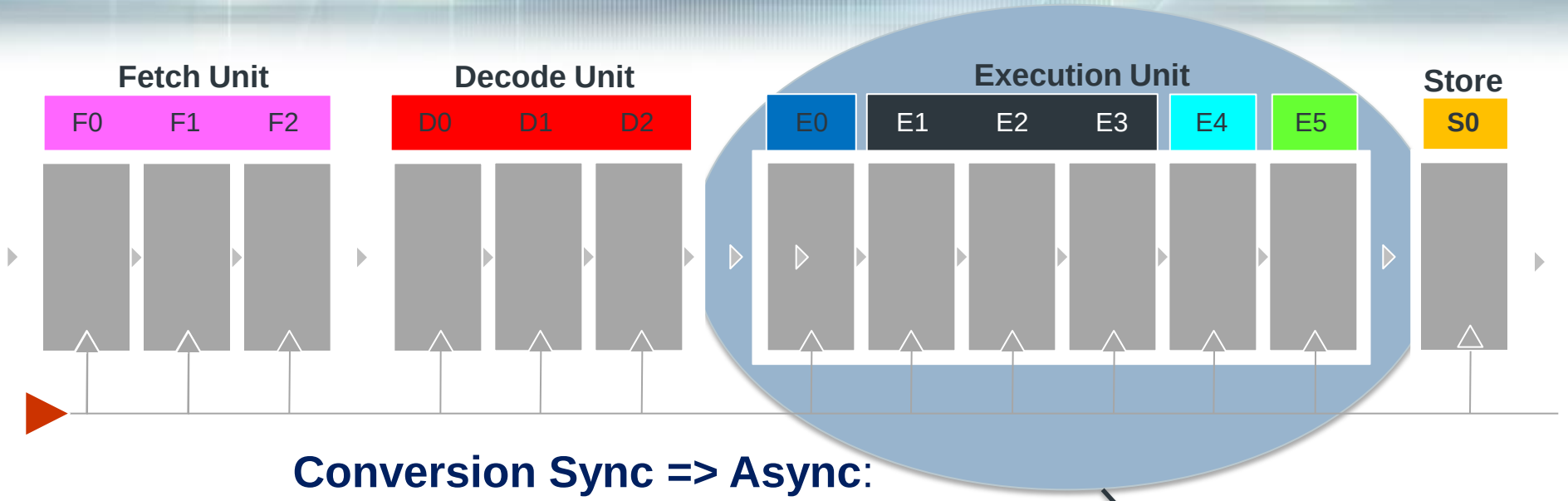
MEM load/store not show

Conversion Sync => Async:

- One way is to map each **unit functionality** into an equivalent asynchronous unit
- But using this methodology will slow down the unit!



SYNC VS ASYNC PROCESOR IMPLEMENTATION

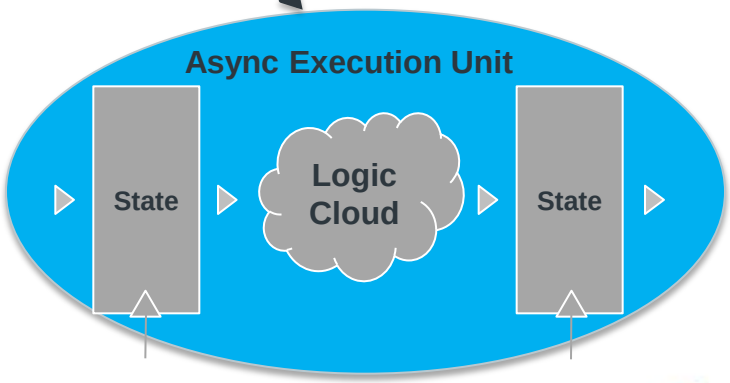


Fetch
Decode
Reg Reads
Execute
Branch
Output Write
Store

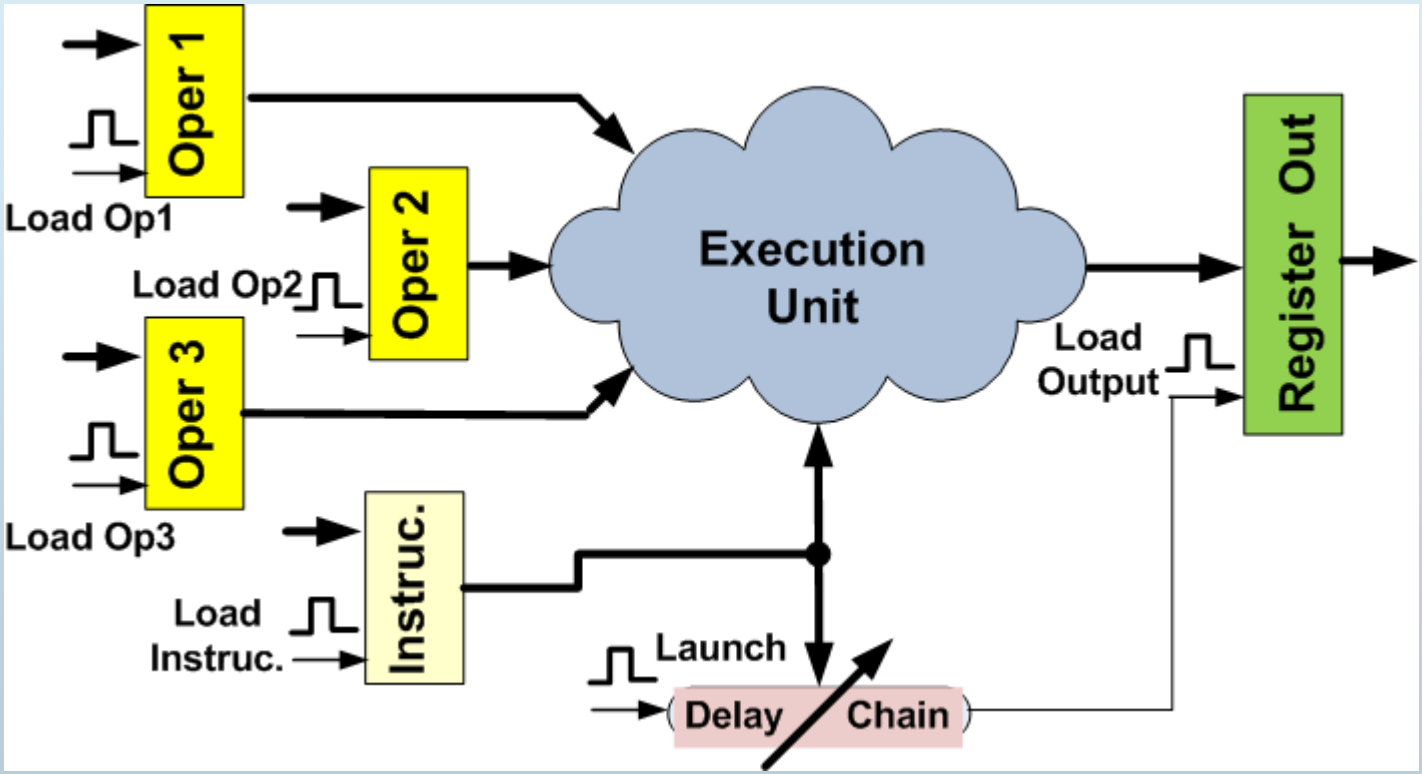
MEM load/store not show

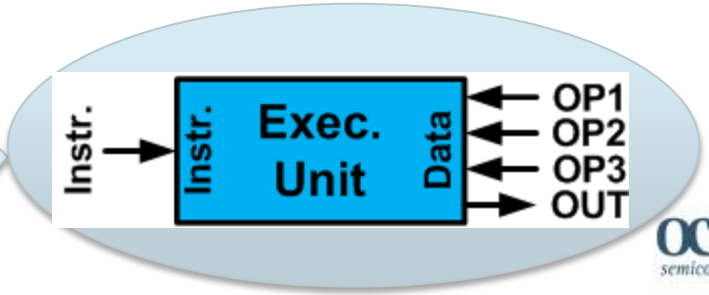
Conversion Sync => Async:

- One way is to map each **unit functionality** into an equivalent asynchronous unit
- But using this methodology will slow down the unit!
- **How can we get the performance back?**



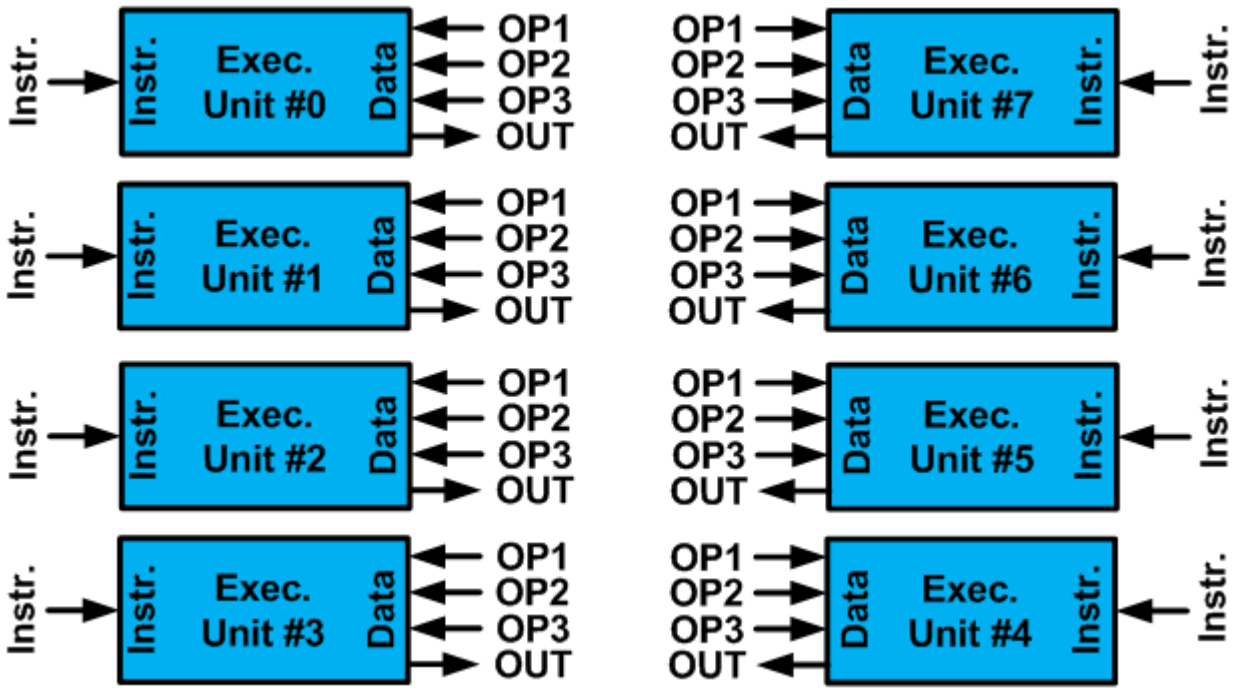
ASYNC ILP IMPLEMENTATION (1)





ASYNC ILP IMPLEMENTATION (2)

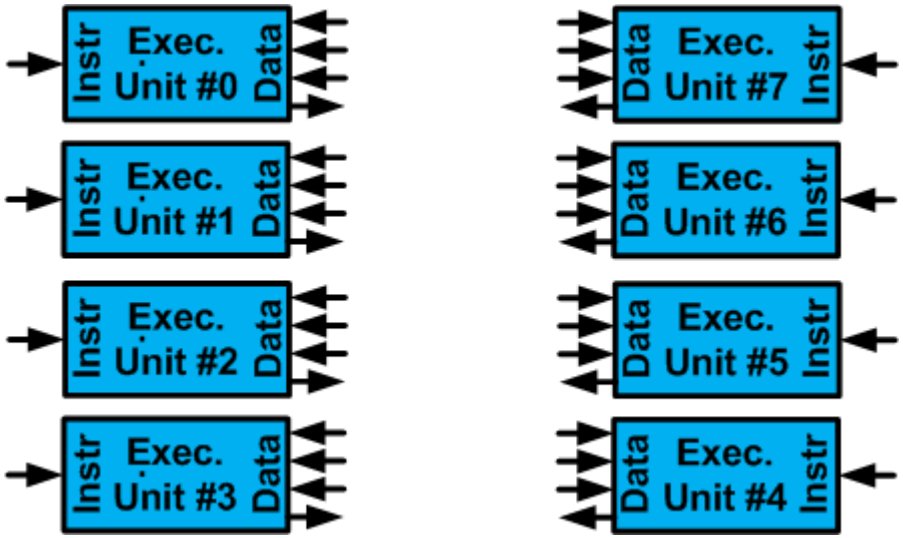
To multiply the processing power of our processor we could use multiple Exec Units (EUs) operating in parallel



Now how can we transparently weave together those EUs ...
....so they behave as one processor?

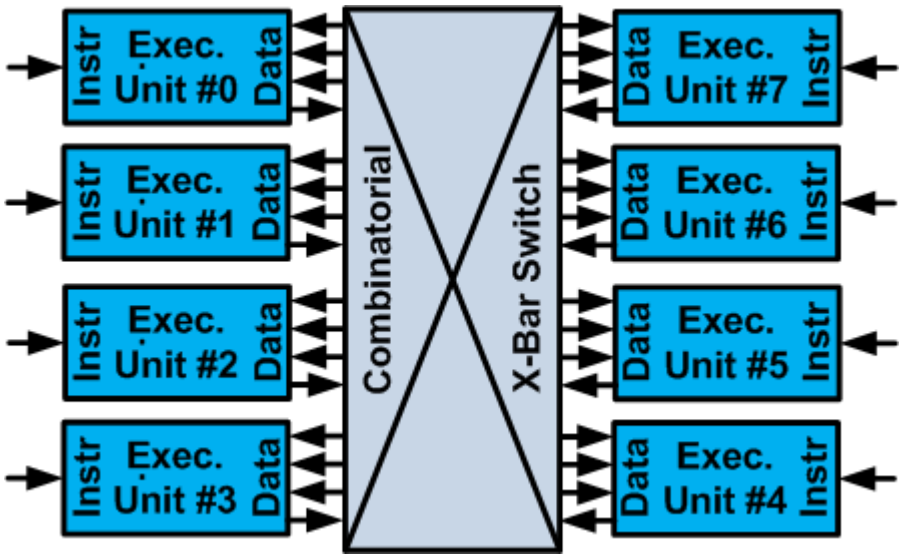
ASYNC PROCESSOR ARCHITECTURE (2)

- Starting with the 8 execution units …



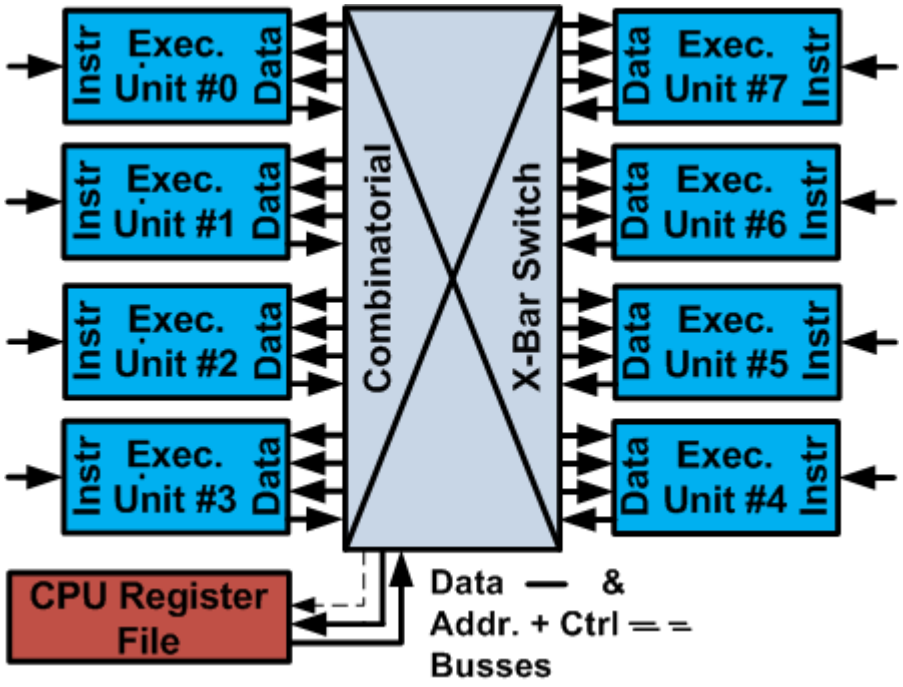
ASYNC PROCESSOR ARCHITECTURE (3)

- Adding a non-blocking **combinatorial X-Bar switch** to:
  - connect the execution units data paths among themselves, and
  - with external resources – register file, memory, etc.



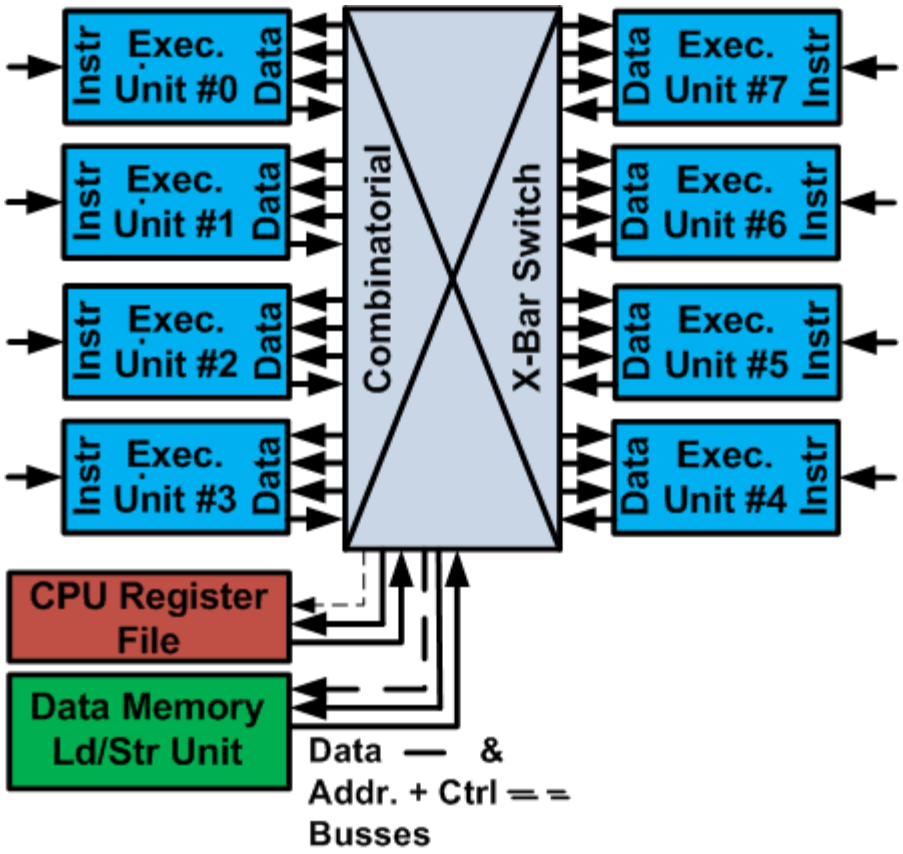
ASYNC PROCESSOR ARCHITECTURE (4)

- **Adding a CPU Register File to implement a load/store processor design:**



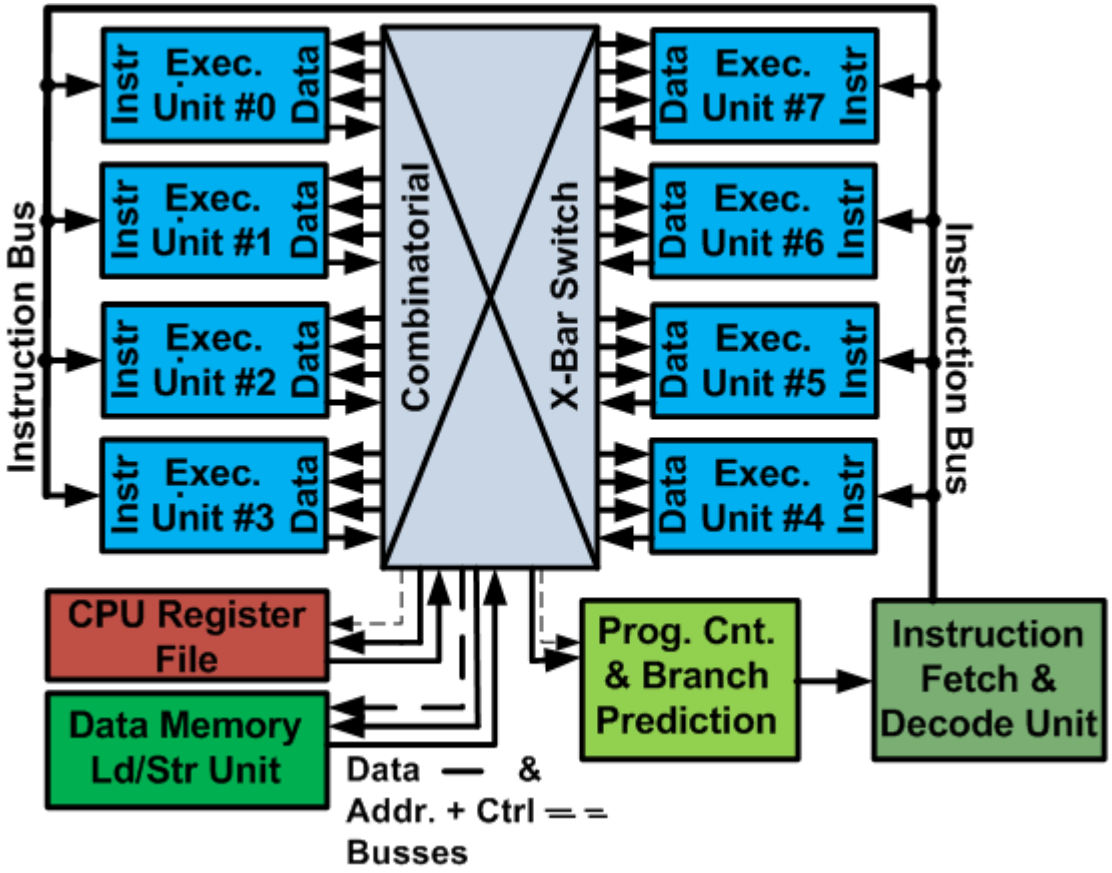
ASYNC PROCESSOR ARCHITECTURE (5)

- Adding a Data Memory Load/Store unit
  - to be able to load/store memory data into/from the CPU (registers)



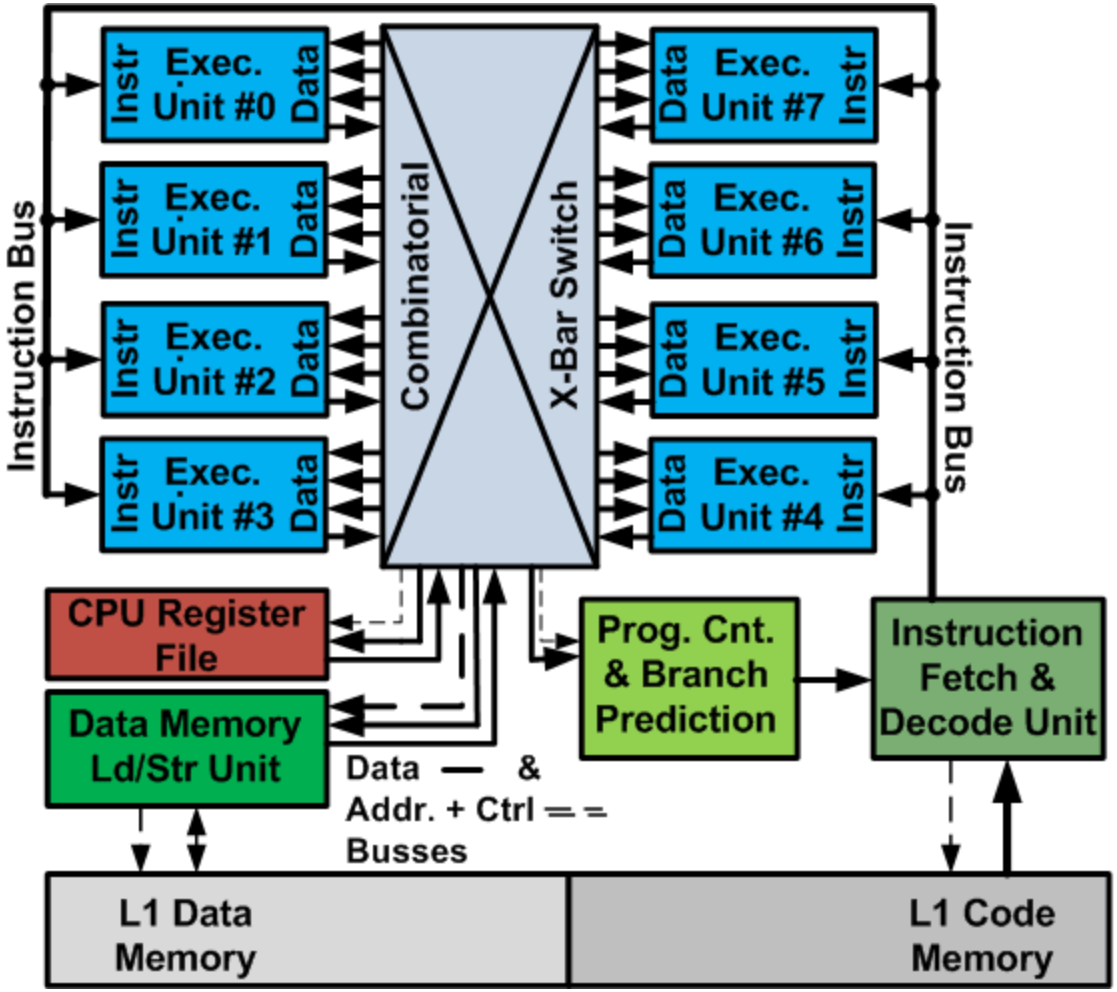
ASYNC PROCESSOR ARCHITECTURE (6)

- Adding a **Program Counter Control** unit including a branch predictor;
- Coupled with an **Instruction Fetch & Decode** Unit
  - to be able to load instructions into the execution units



ASYNC PROCESSOR ARCHITECTURE (7)

- Adding L1 Memory accessible for:
  - Data, or
  - Code

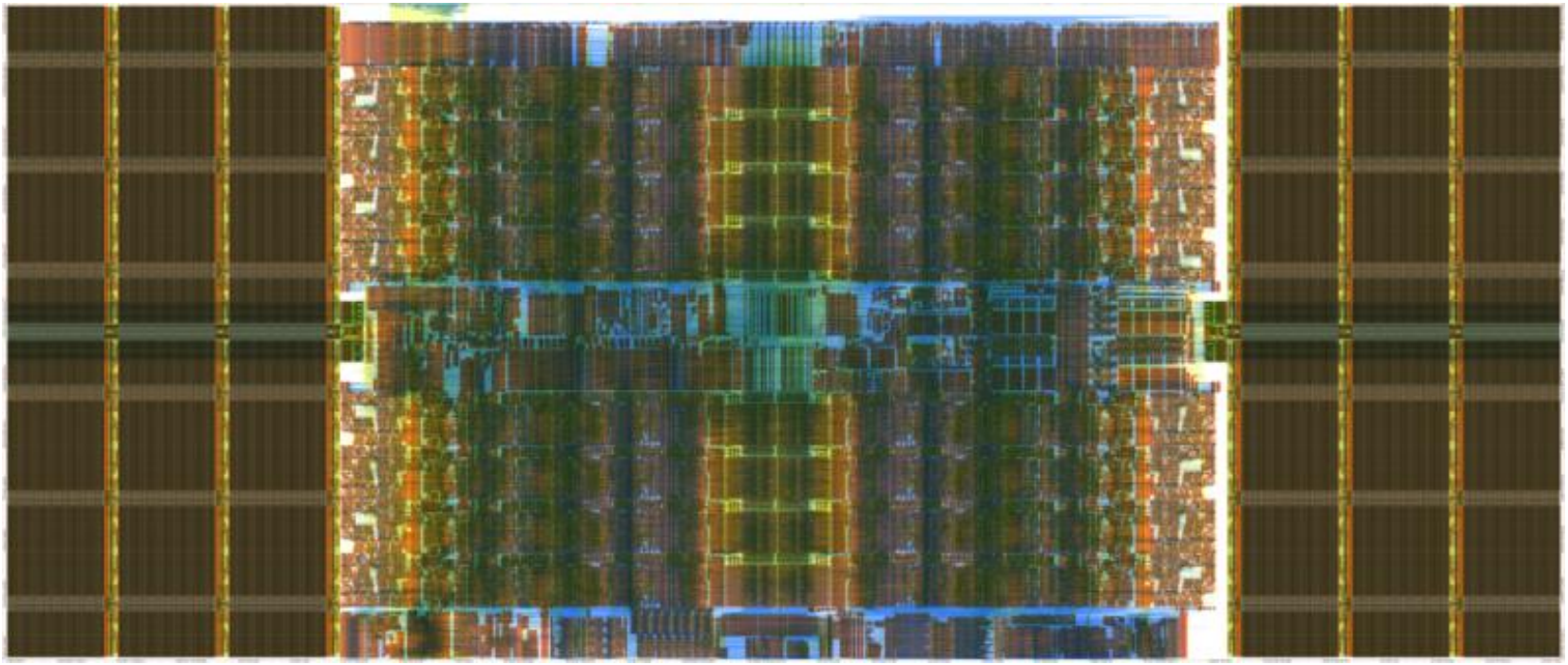


CONTENTS

- Background
- Asynchronous Circuits Description
- **Processor Architecture and Operation (Simplified)**
  - Architecture, **Silicon**, and ILP Implementation
  - Operation & Synchronization
- Performance Analysis
- Conclusion

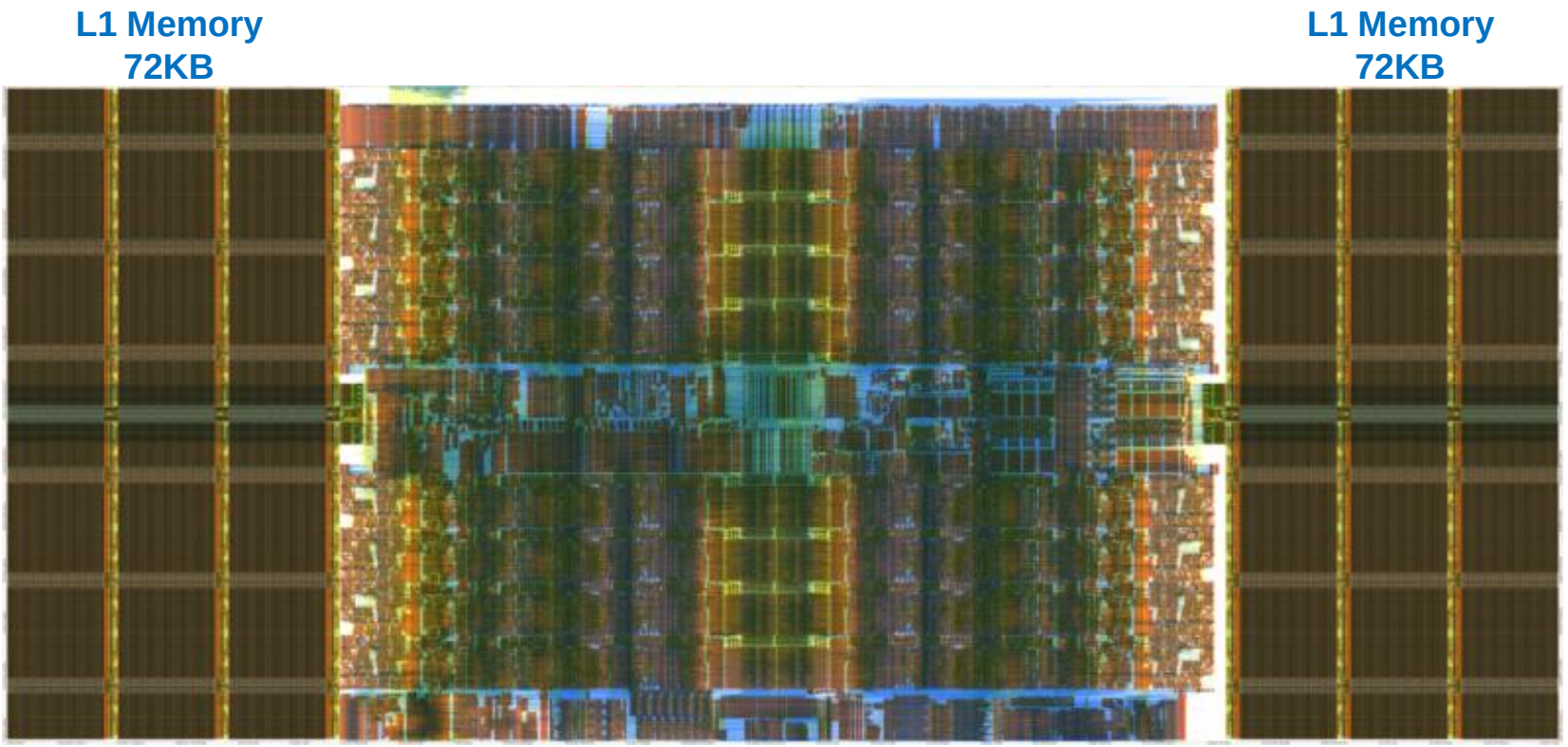
ASYNC PROCESSOR ARCHITECTURE (8)

- **How does this map on silicon?**



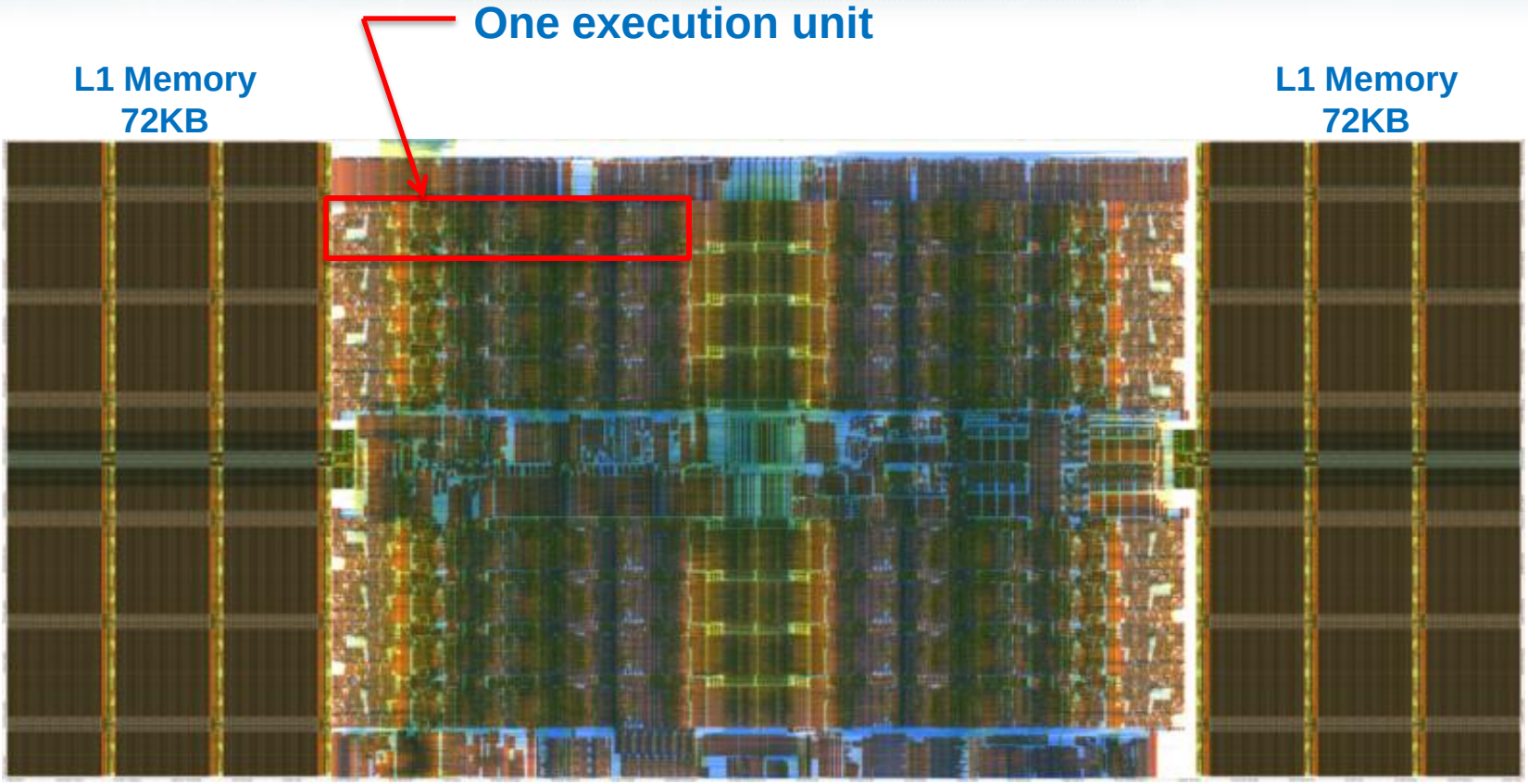
ASYNC PROCESSOR ARCHITECTURE (8)

- How does it map on silicon?



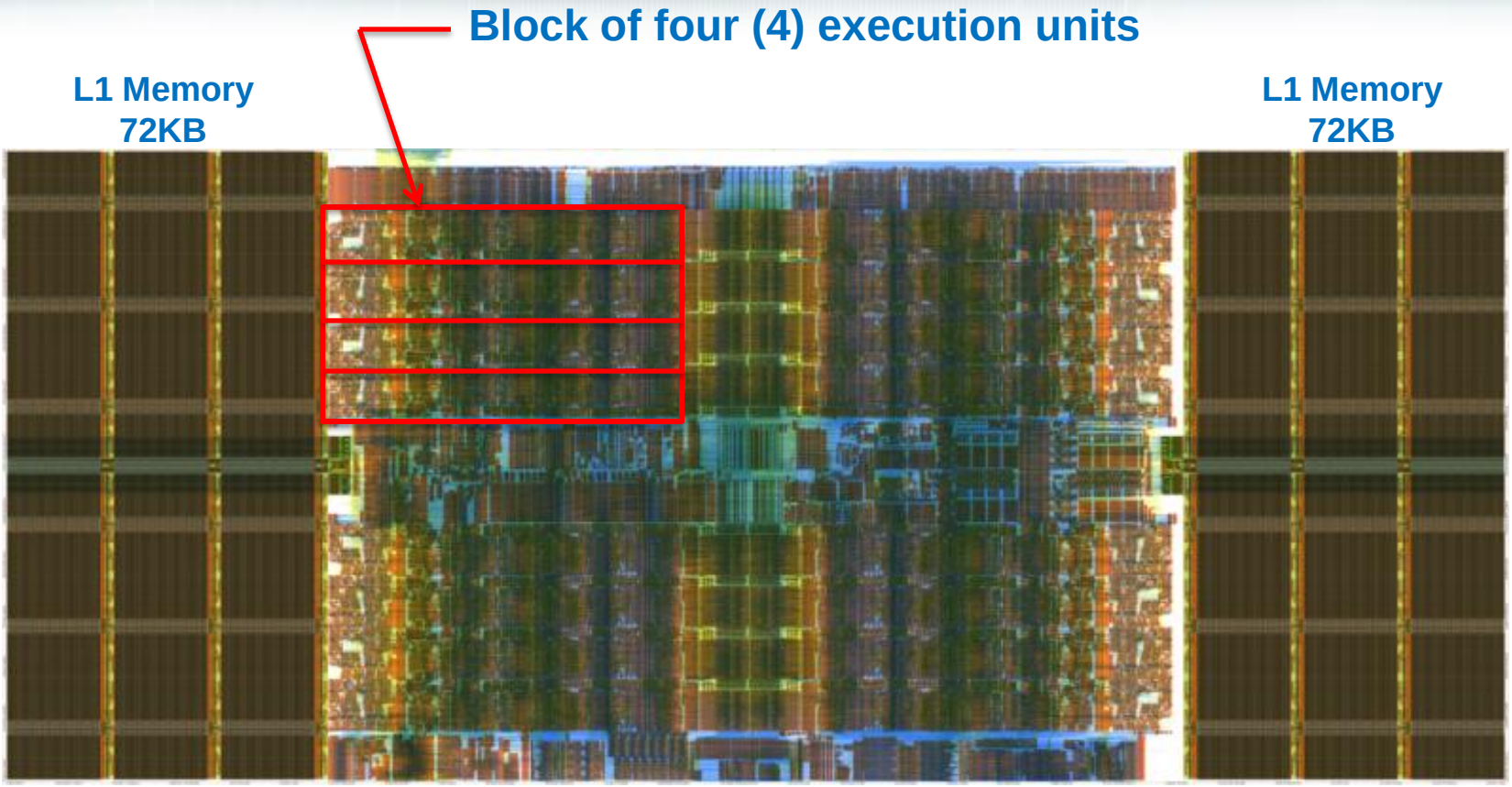
ASYNC PROCESSOR ARCHITECTURE (8)

- How does it map on silicon?



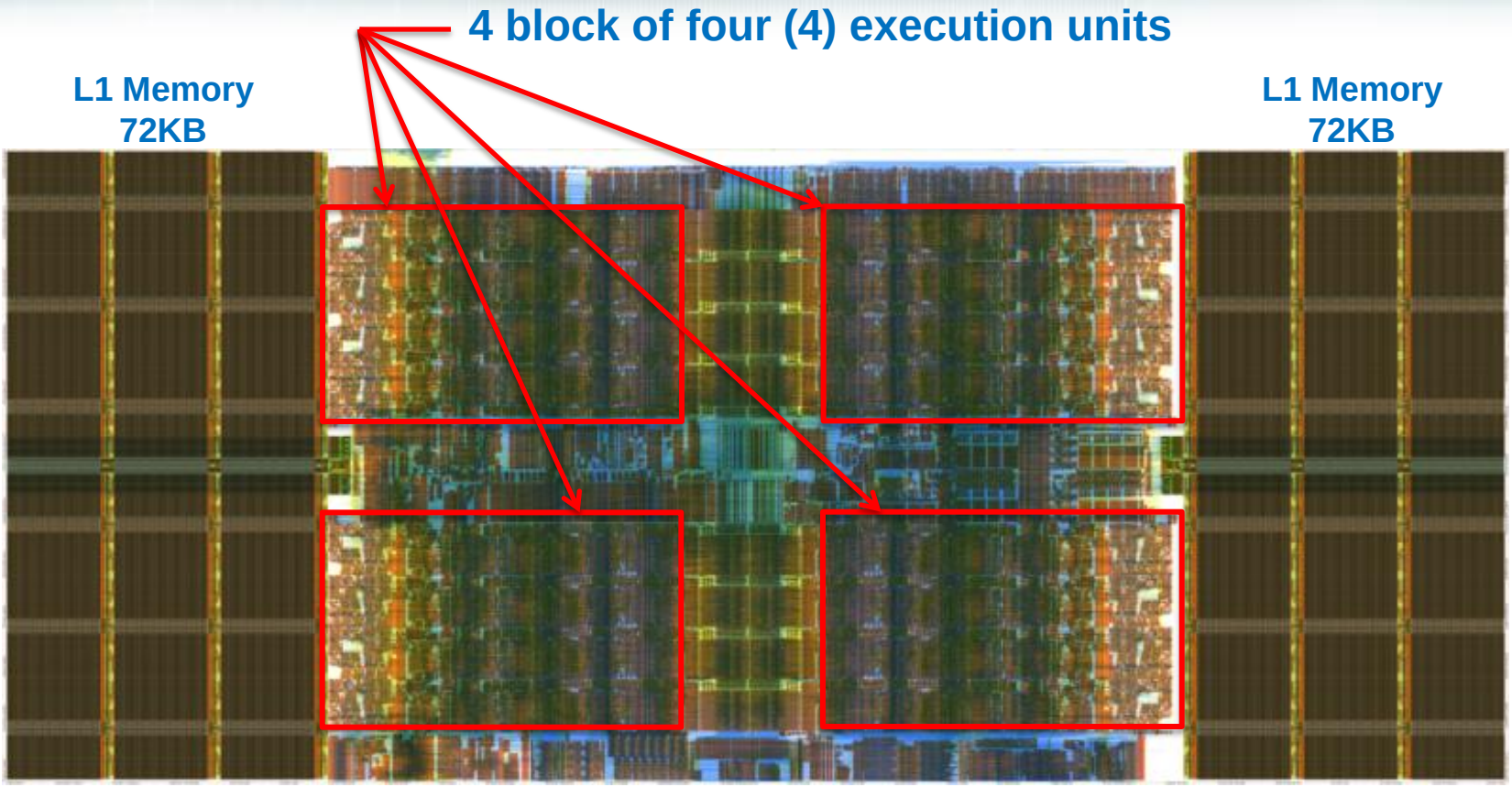
ASYNC PROCESSOR ARCHITECTURE (8)

- How does it map on silicon?



ASYNC PROCESSOR ARCHITECTURE (8)

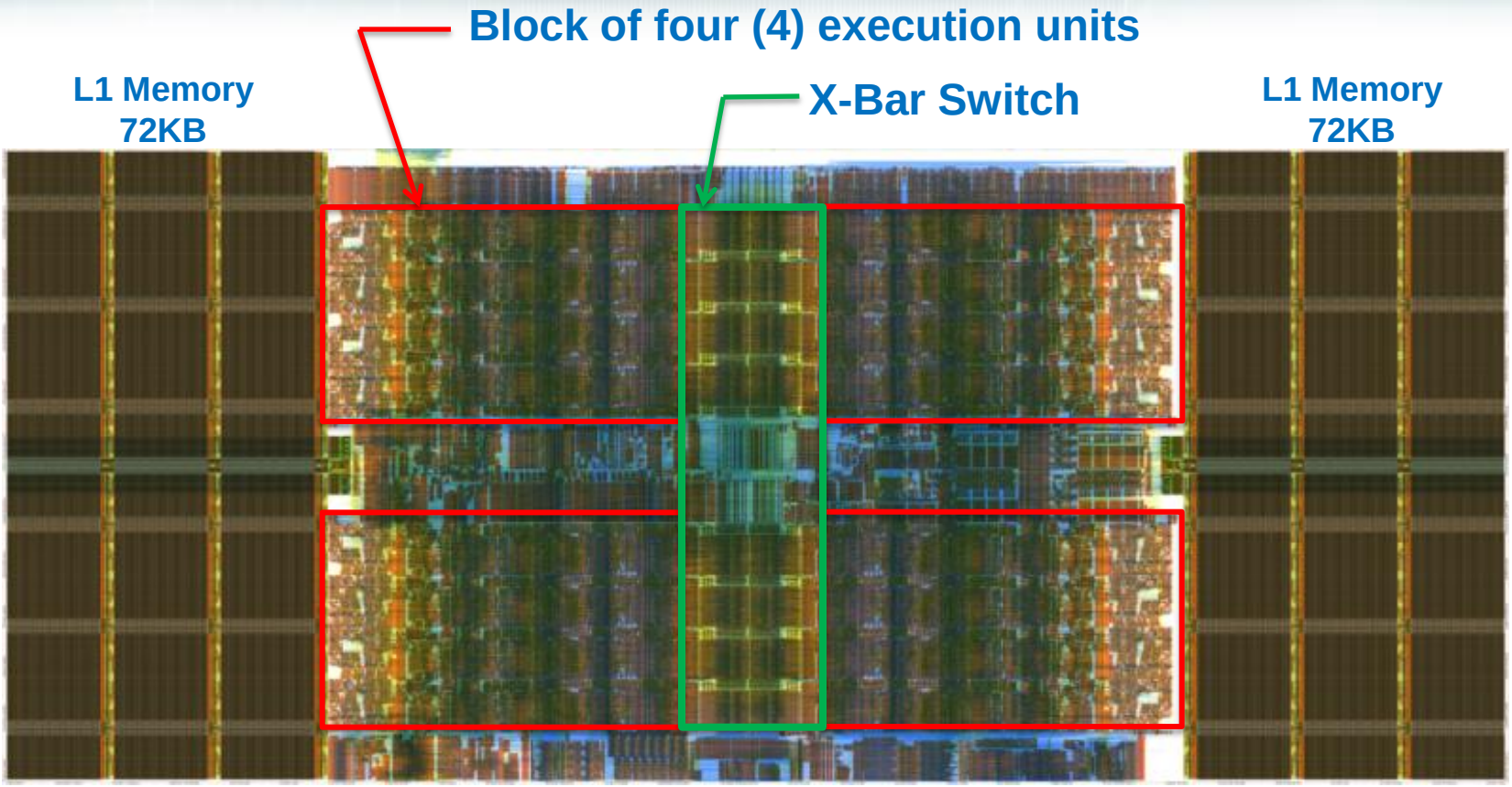
- How does it map on silicon?



There are indeed 16 Execution Units, not 8 EUs in this DSP core!

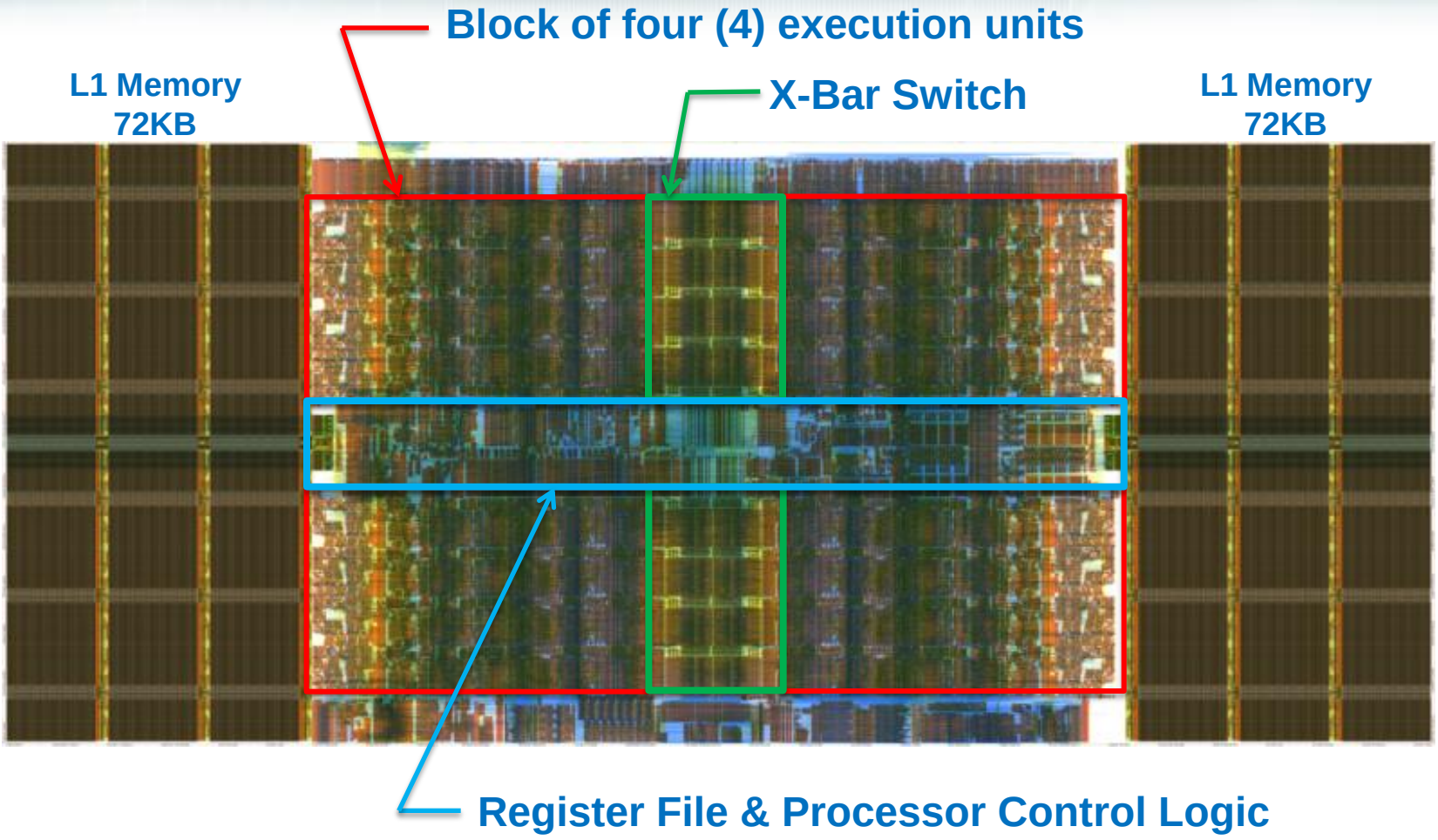
ASYNC PROCESSOR ARCHITECTURE (8)

- How does it map on silicon?



ASYNC PROCESSOR ARCHITECTURE (8)

- How does it map on silicon?

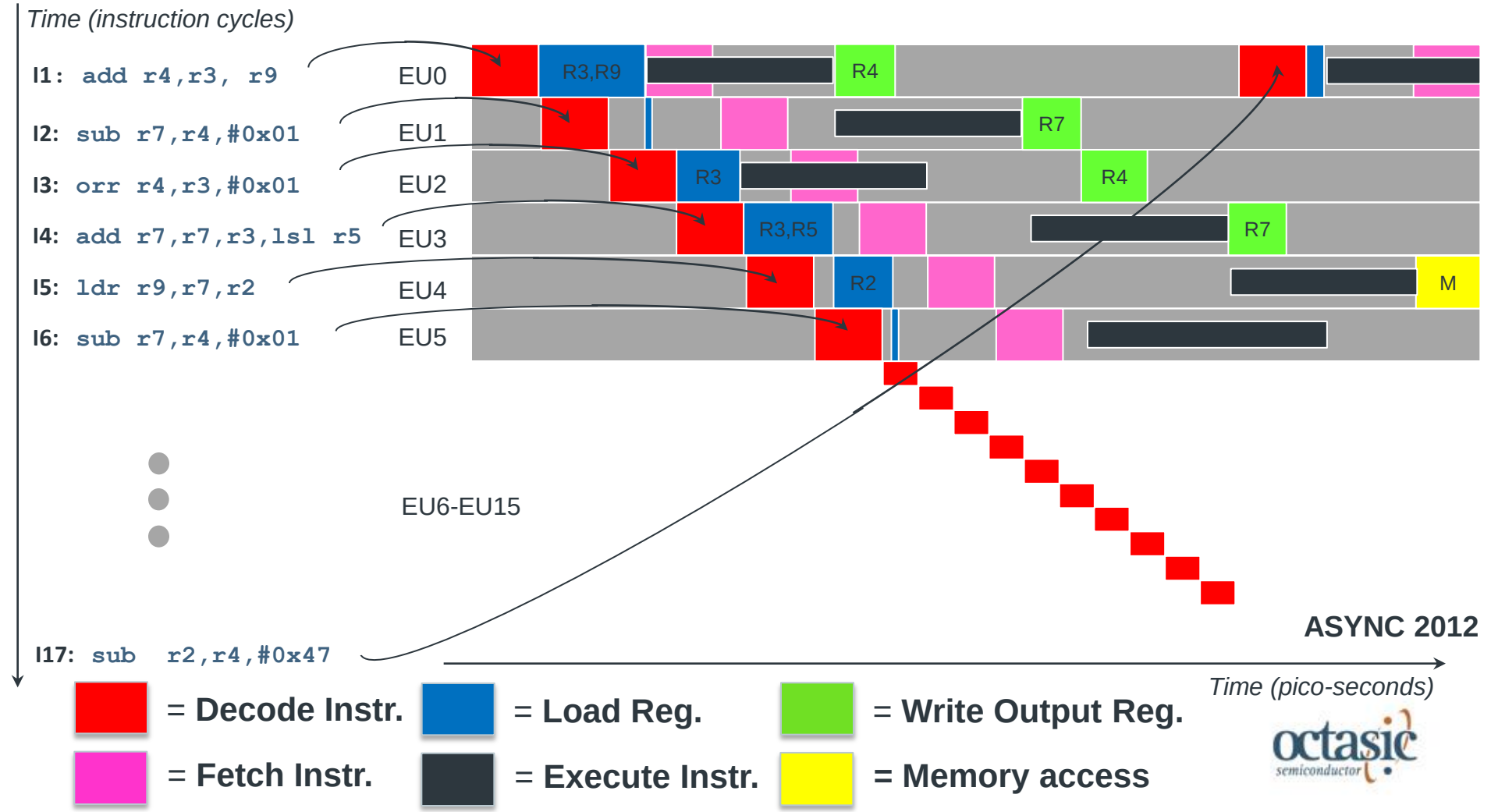


CONTENTS

- Background
- Asynchronous Circuits Description
- **Processor Architecture and Operation (Simplified)**
  - Architecture, Silicon, and **ILP Implementation**
  - Operation & Synchronization
- Performance Analysis
- Conclusion

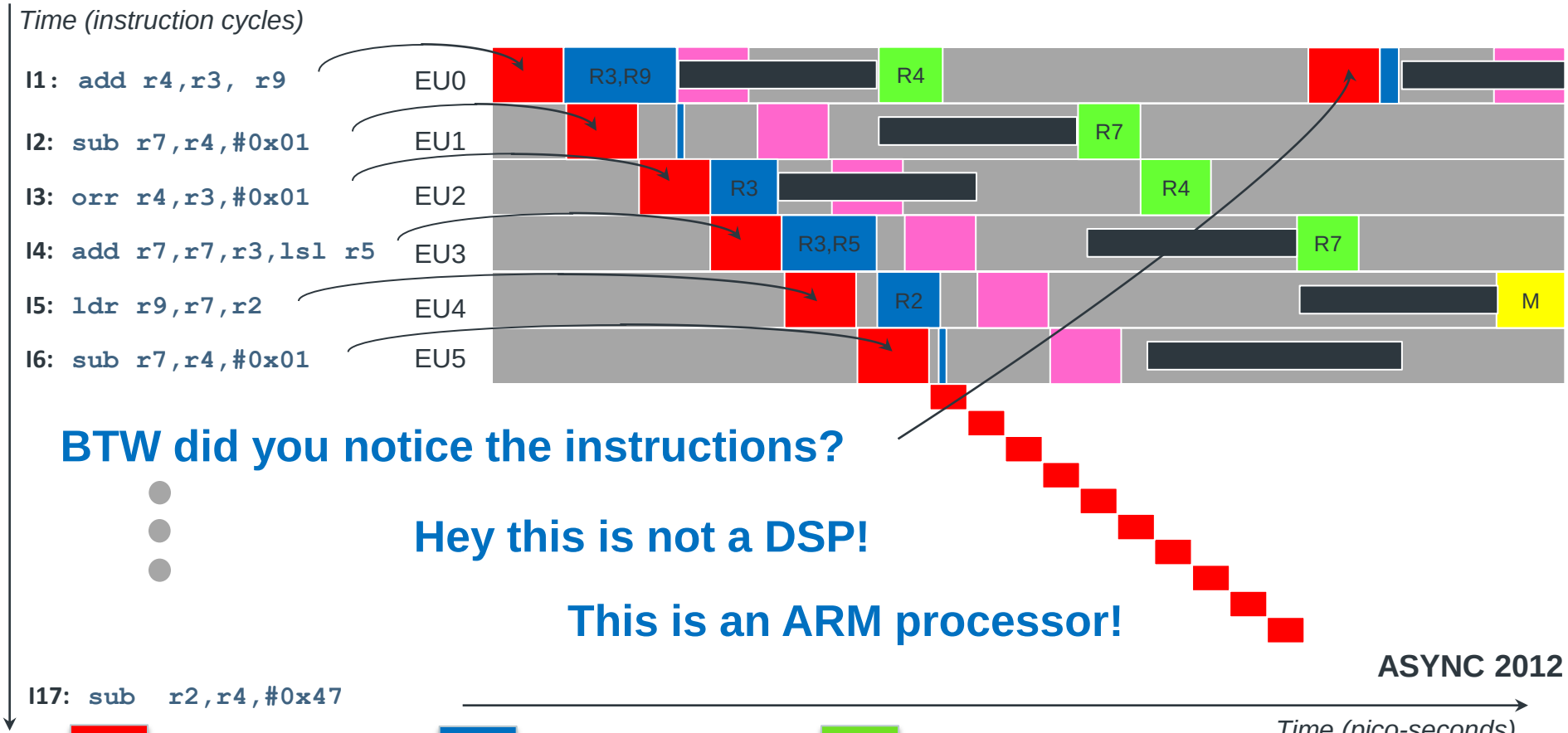
PROCESSOR OPERATION – SIMPLIFIED ILP (1)

Assuming the operation of the Execution Units and resources (registers, memory, …) are somehow synchronized, here is the flow of instructions overlap that would result in the processor; hence realizing the Instruction Level Parallelism (ILP) mechanism to boost performance

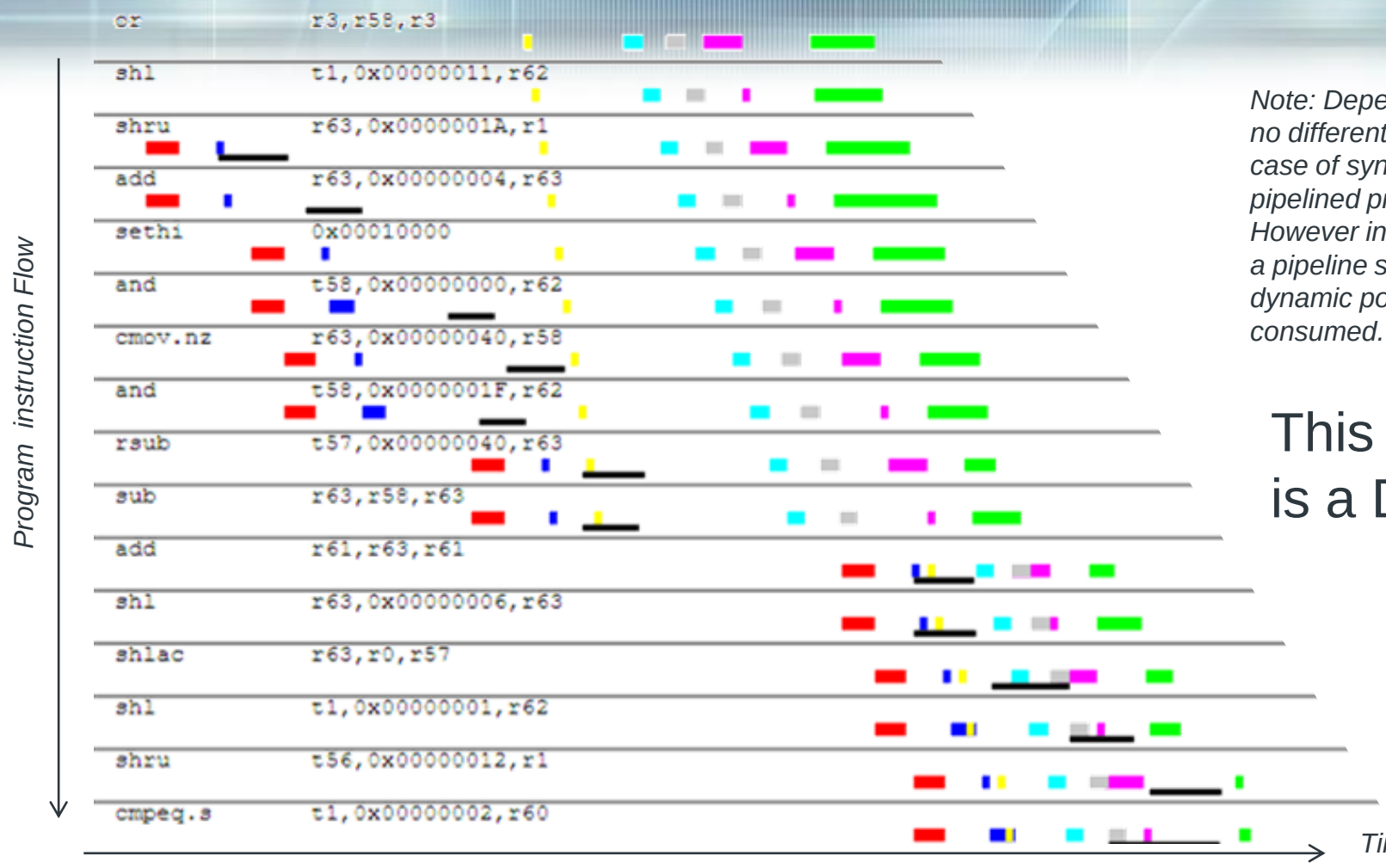


PROCESSOR OPERATION – SIMPLIFIED ILP (1)

Assuming the operation of the Execution Units and resources (registers, memory, …) are somehow synchronized, here is the flow of instructions overlap that would result in the processor; hence realizing the Instruction Level Parallelism (ILP) mechanism to boost performance



PROCESSOR OPERATION ILP: REAL-WORLD EXAMPLE (2)



Note: Dependencies are no different than in the case of synchronous pipelined processors. However in the event of a pipeline stall, no dynamic power is consumed.

This time it is a DSP!

ASYNC 2012

= Decode Instr. **= Load Reg.** **= Write Output Reg.**

= Fetch Instr. **= Execute Instr.** **= Memory access**

octasic semiconductor

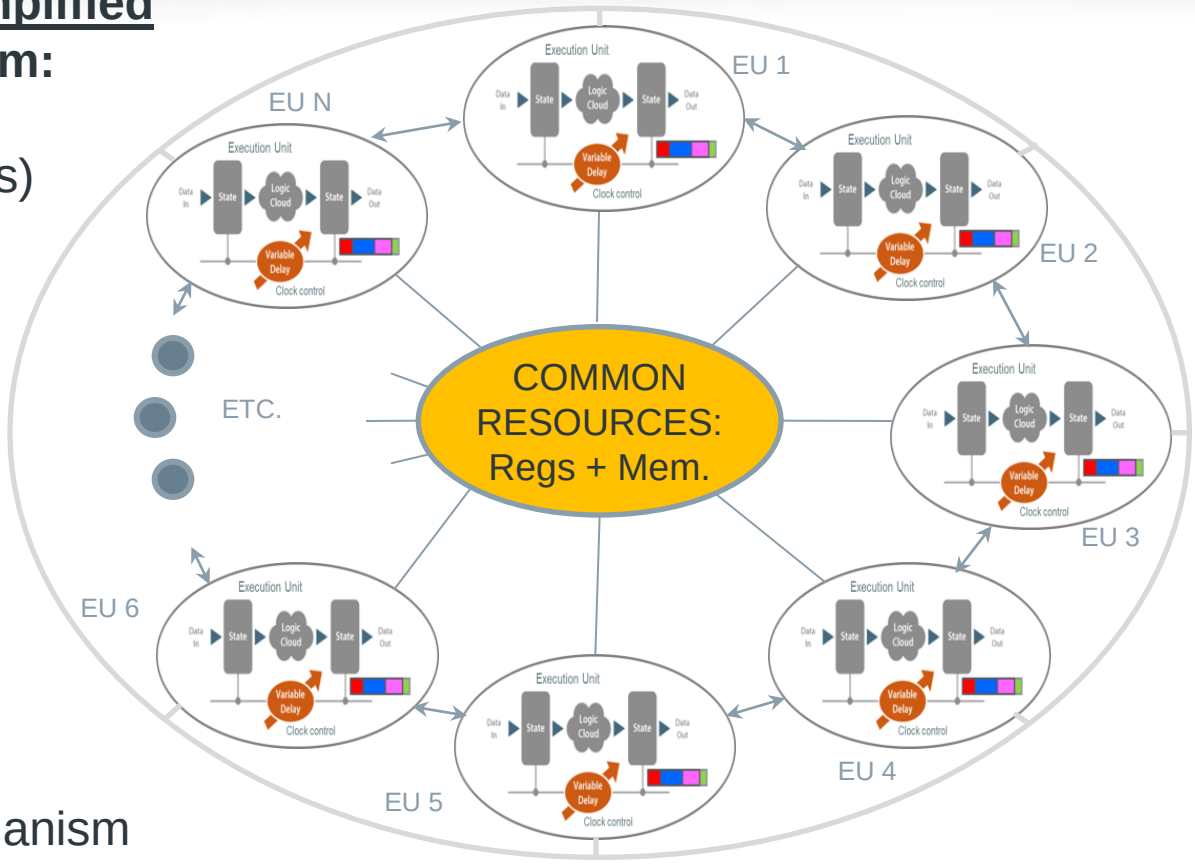
CONTENTS

- Background
- Asynchronous Circuits Description
- **Processor Architecture and Operation (Simplified)**
  - Architecture, Silicon, and ILP Implementation
  - **Operation & Synchronization**
- Performance Analysis
- Conclusion

OPERATION AND SYNCHRONIZATION (1)

This is an alternate <u>simplified</u> processor block diagram:

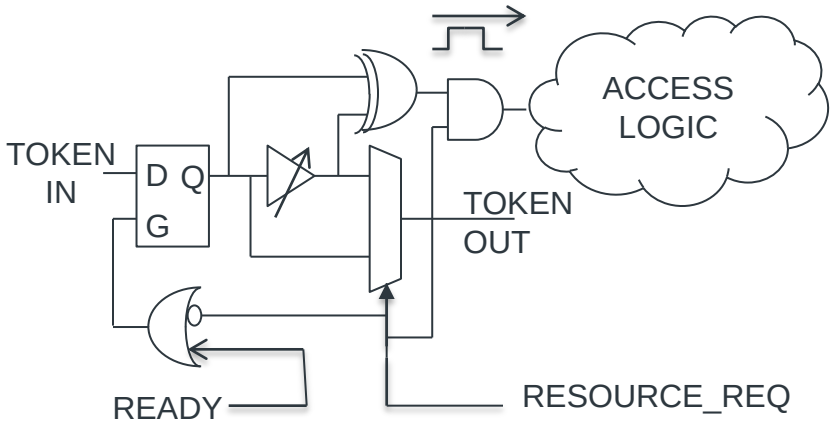
- the execution units (EUs) are mapped in a ring like fashion
- the EUs have access to common resources:
  - Register File
  - Data Memory
  - Code Memory
  - X-Bar
  - PC Control Logic
- a synchronization mechanism is needed to arbitrate and avoid conflicts in the access of the EUs to the common resources



OPERATION AND SYNCHRONIZATION (2)

In contrast with a synchronous processor which is generally centrally controlled, this asynchronous processor has a fully distributed control system:

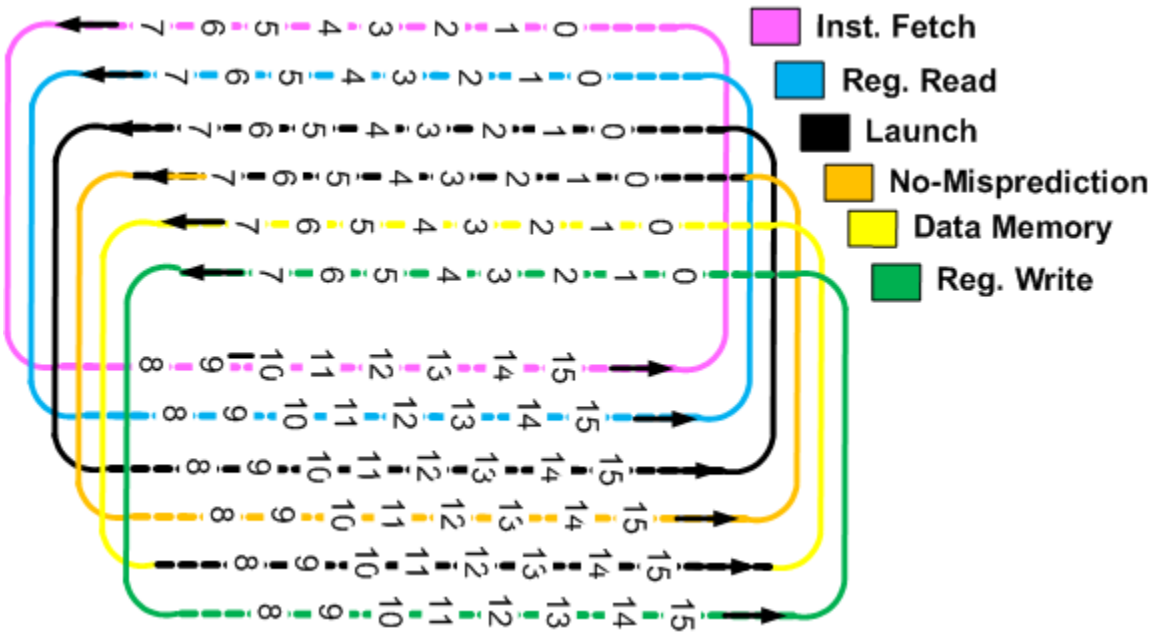
- Control is exercised individually by each Execution Unit (EU)
- Control tokens are passed asynchronously among the EUs in a ring fashion to synchronize accesses to common resources and avoid conflicts
- In the simplified model discussed herein, six (6) tokens are used:
  - Instruction Fetch Token
  - Register Read Token
  - Launch Execution Token (X-Bar, Reg Ready)
  - No Mis-Prediction Token (PC & Write Commit)
  - Data Memory Token (Rd or Wr)
  - Register Write Token



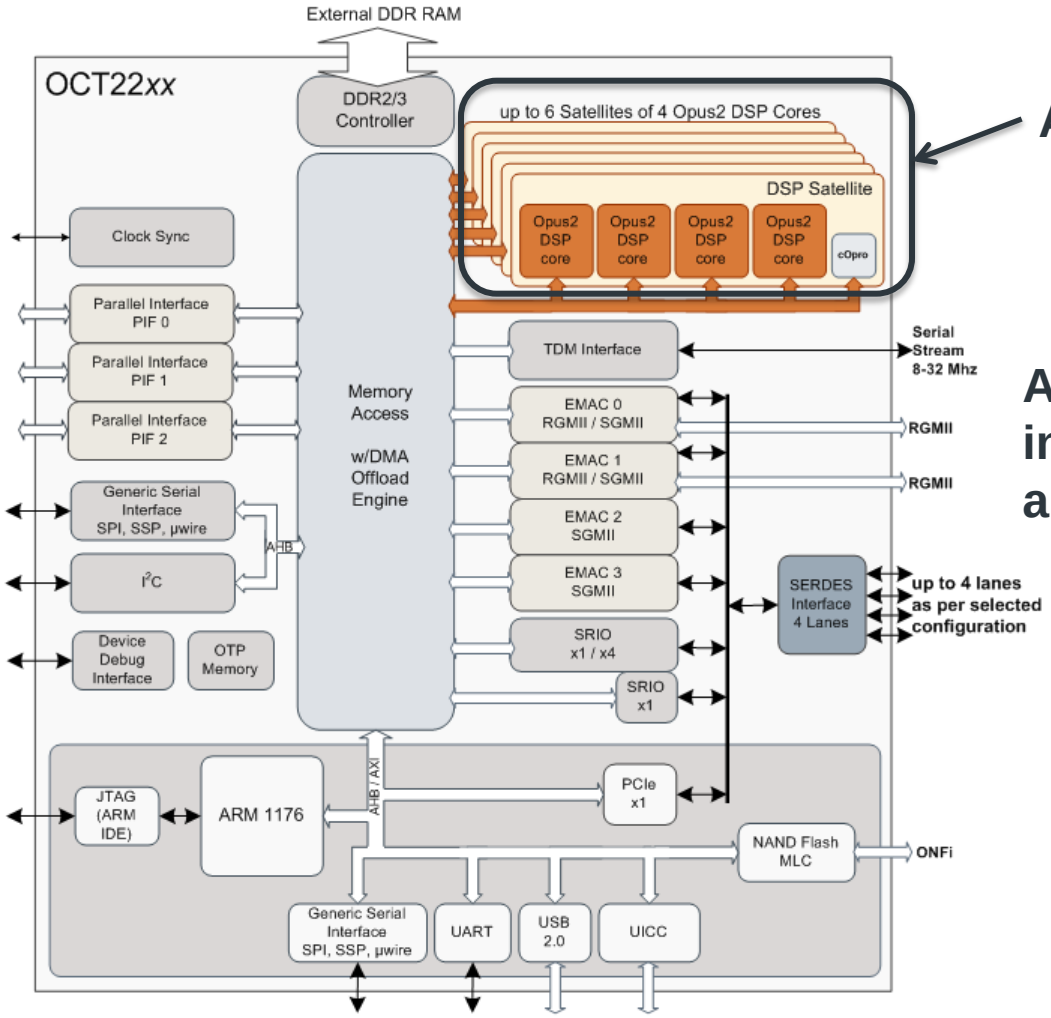
OPERATION AND SYNCHRONIZATION (3)

Asynchronous control tokens are used to control and synchronize the overall operation of the processor.

- Control tokens are passed from one EU to the next in a ring fashion.
- When a token is owned by an EU it can use it to request services (via Req pulses)
- When a service request is sent and a certain time has elapsed and certain conditions are met, or when the EU does not need the token (resource) the token is passed to the next EU.
- On start up or after a flush (wrongly predicted branch), all tokens are assigned to the same EU.



OCT2224 SOC ARCHITECTURE (1)



Asynchronous SoC Portion:

- 24 async DSP Cores

All other modules in the SoC including the external interfaces are all synchronous:

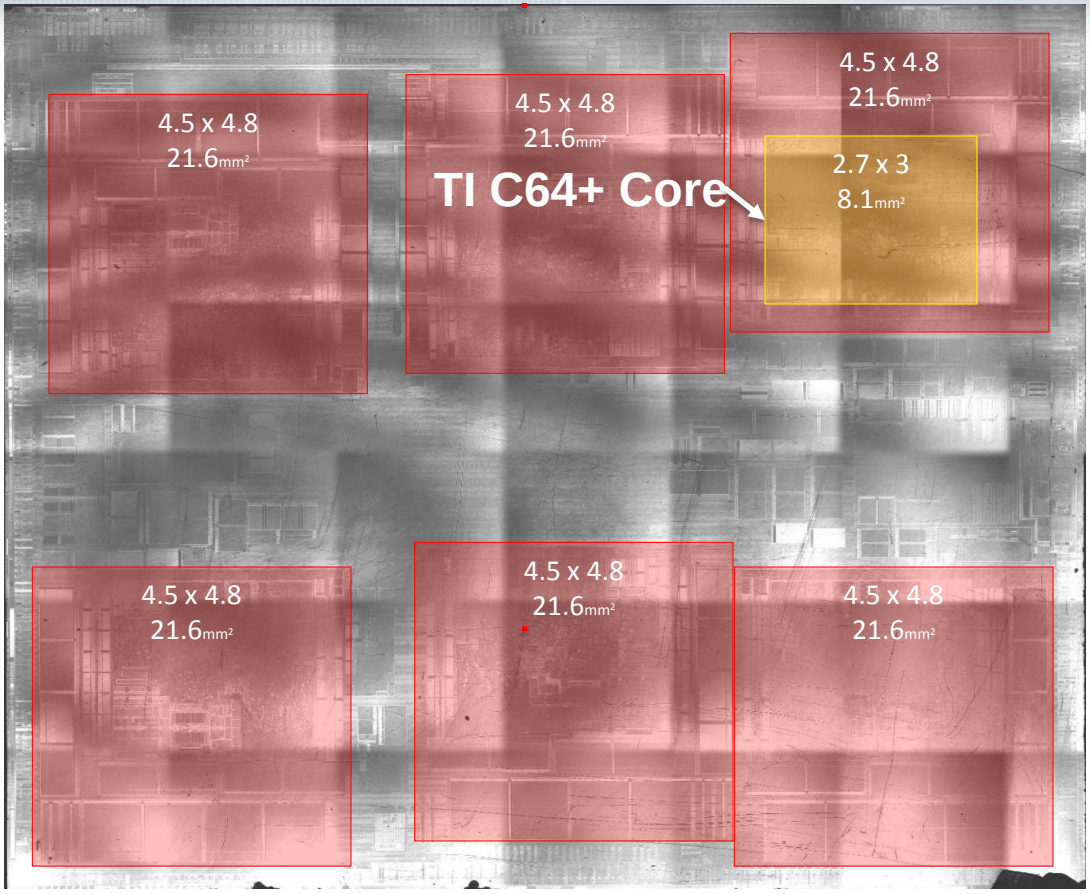
- not power critical
- bought IP blocks
- ease of interface

CONTENTS

- Background
- Asynchronous Circuits Description
- Processor Architecture and Operation
- **Performance Analysis**
- Conclusion

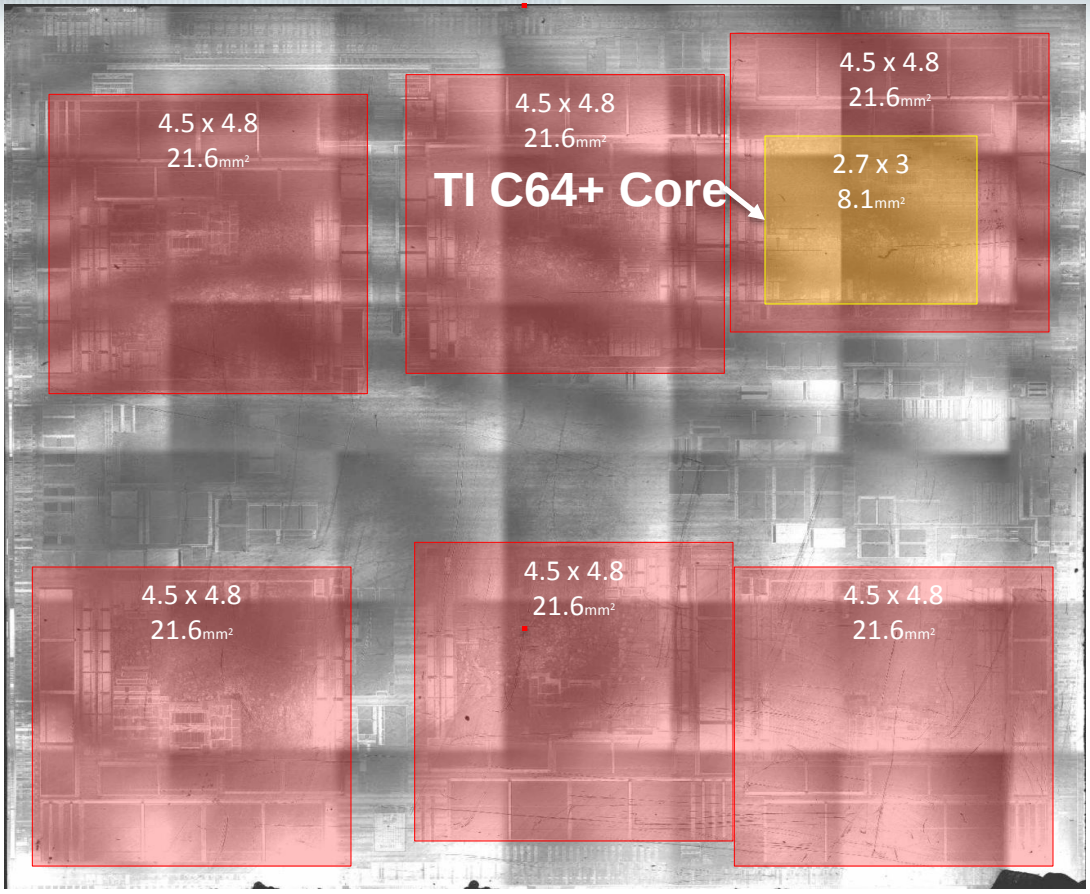
COMPARISON – DIE AREA

- Texas Instruments (TI) is the leading DSP vendor in the industry;
- TI literature claims the C6472® is the most power efficient high-performance DSP in the market. It features 6 ea C64+® cores;



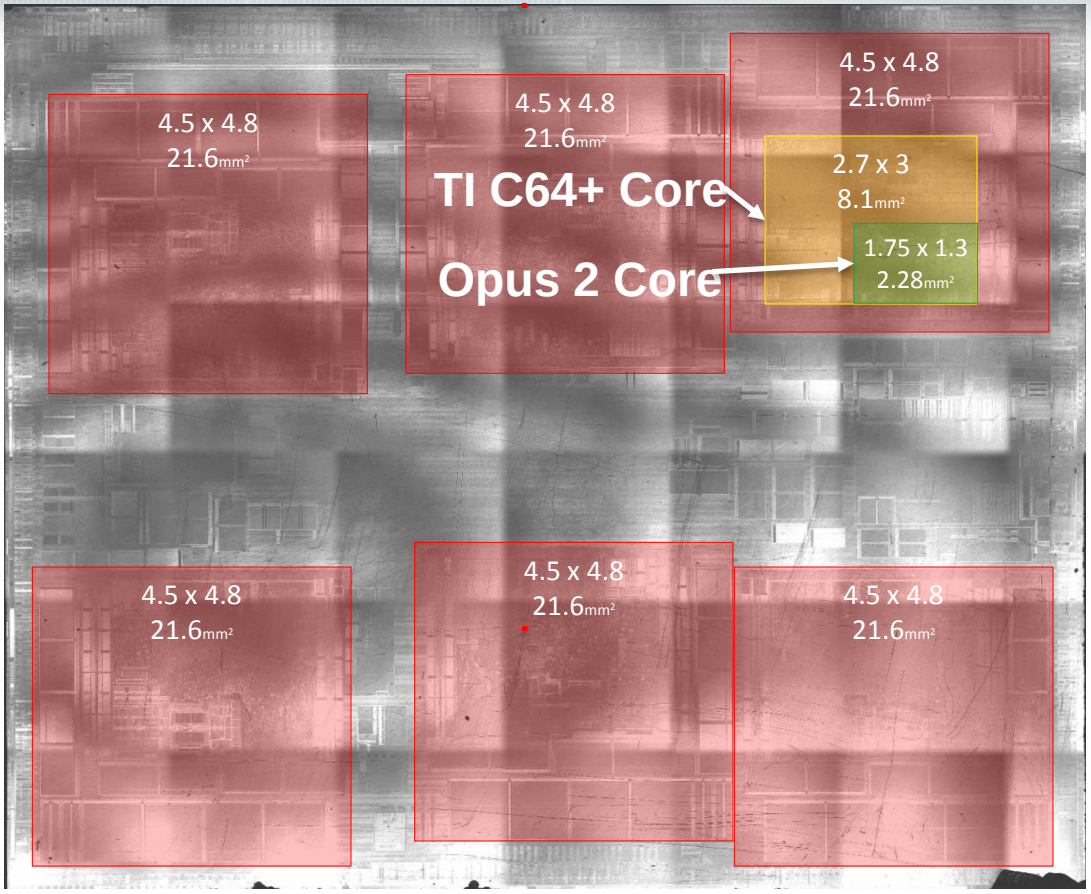
COMPARISON – DIE AREA

- Texas Instruments (TI) is the leading DSP vendor in the industry;
- TI literature claims the C6472® is the most power efficient high-performance DSP in the market. It features 6 ea C64+® cores;
- **The C6472® is implemented in the same silicon technology as one of our DSP so it provides a reasonably fair benchmark*;**
- **The C6472® is a mature device so fairly accurate data is available for area, power consumption, and processing capability*;**
- **The C64+® core area is ~8.1mm² (estimate);**

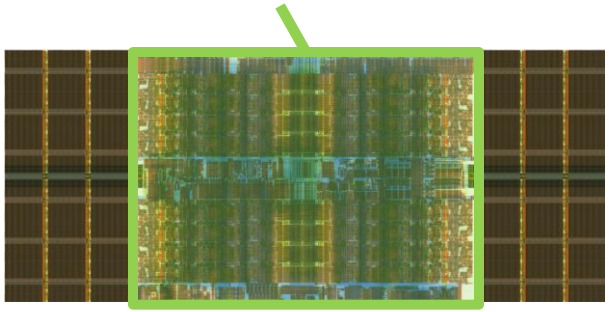


COMPARISON – DIE AREA

- Texas Instruments (TI) is the leading DSP vendor in the industry;
- TI literature claims the C6472® is the most power efficient high-performance DSP in the market. It features 6 ea C64+® cores;
- The C6472® is implemented in the same silicon technology as one of our DSP so it provides a reasonably fair benchmark*;
- The C6472® is a mature device so fairly accurate data is available for area, power consumption, and processing capability*;
- The C64+® core area is ~8.1mm^2 (estimate)
- **Octasic's Opus2 core is 2.28mm^2**
- **Ratio of area: ~3.5**



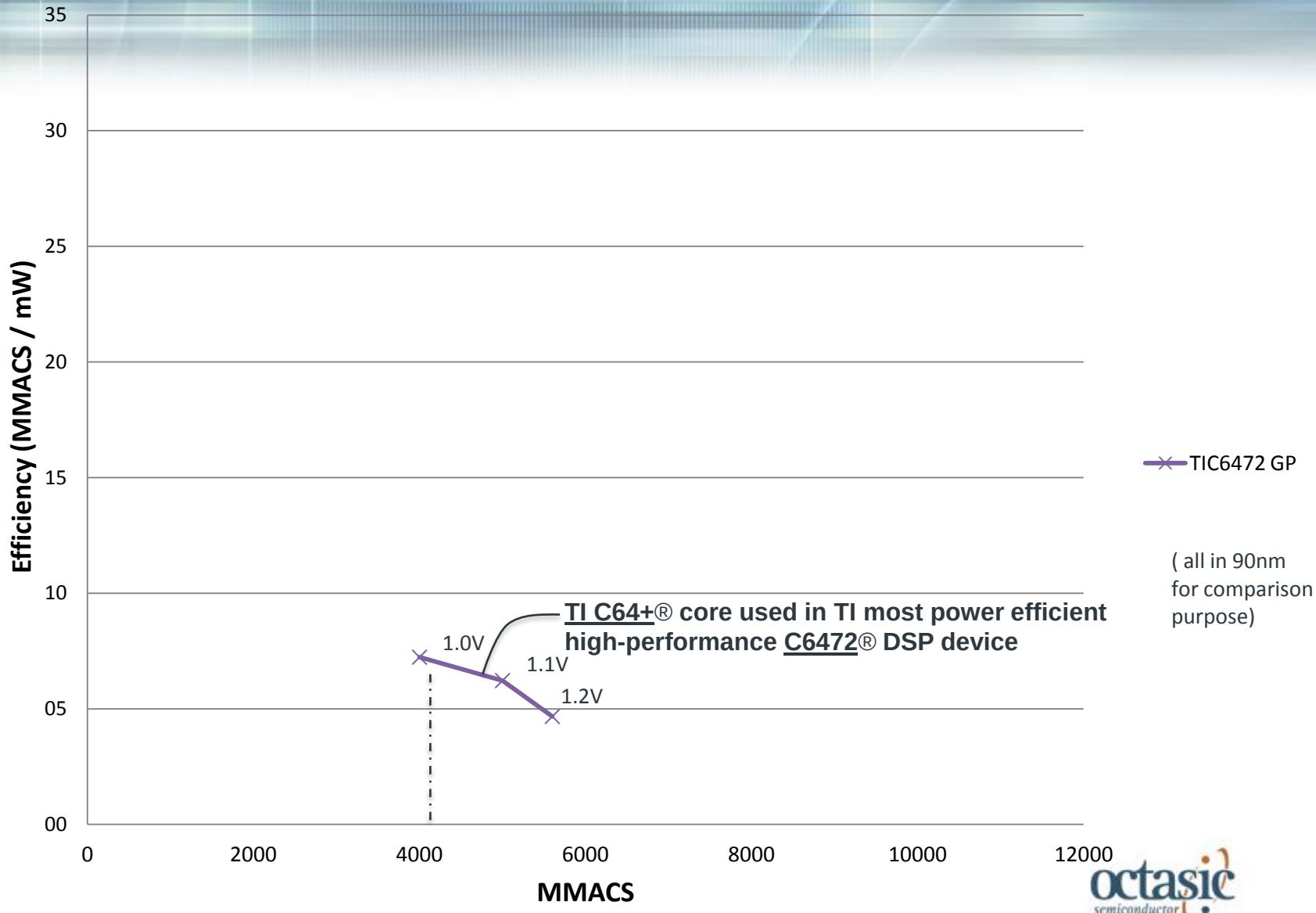
Octasic Opus 2 Core



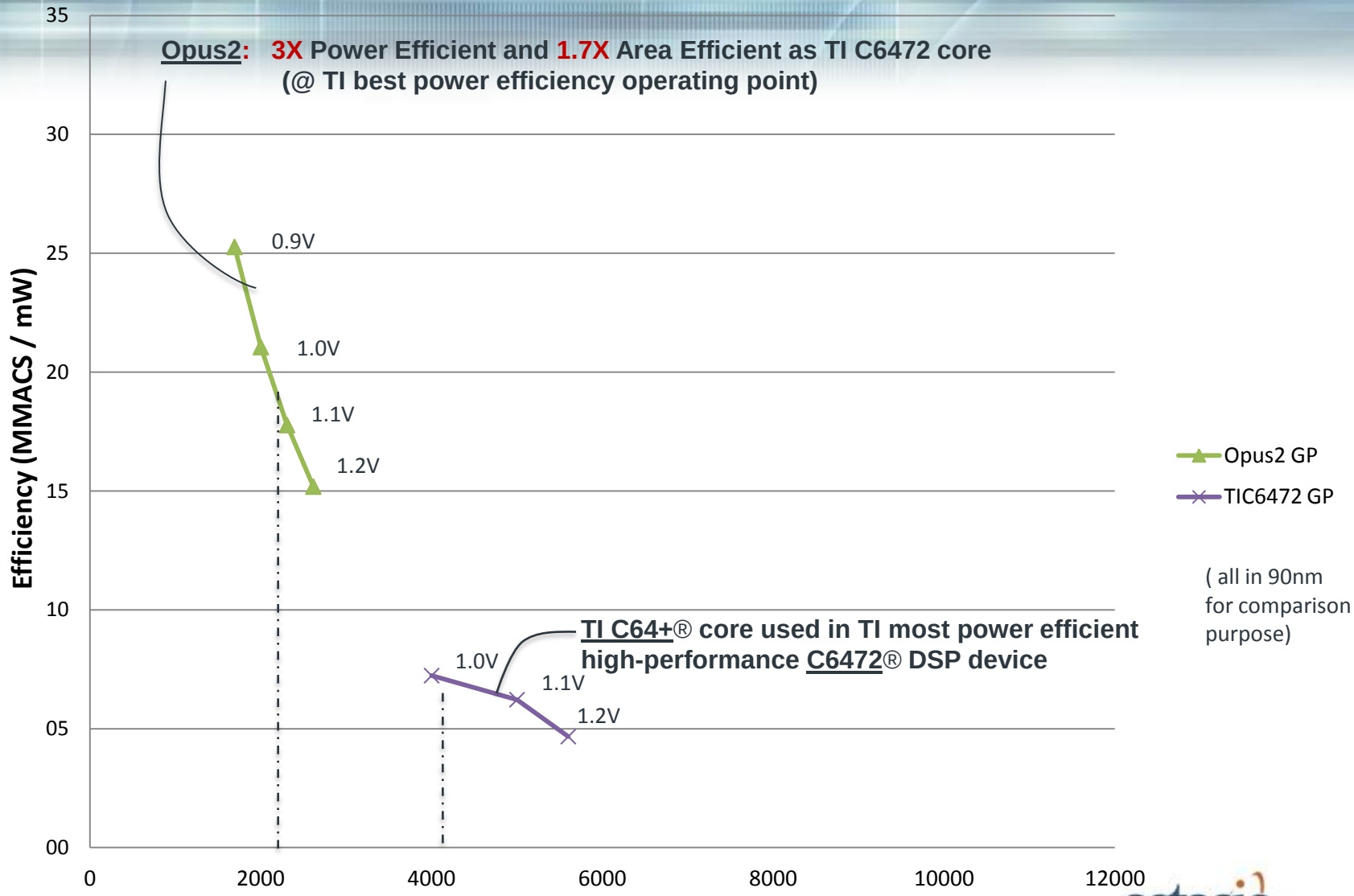
2.28 mm^2

*It is understood that any such data and comparison is never totally accurate and can be subject to many interpretations. The data is therefore provided for discussion only.

COMPARISON – POWER EFFICIENCY

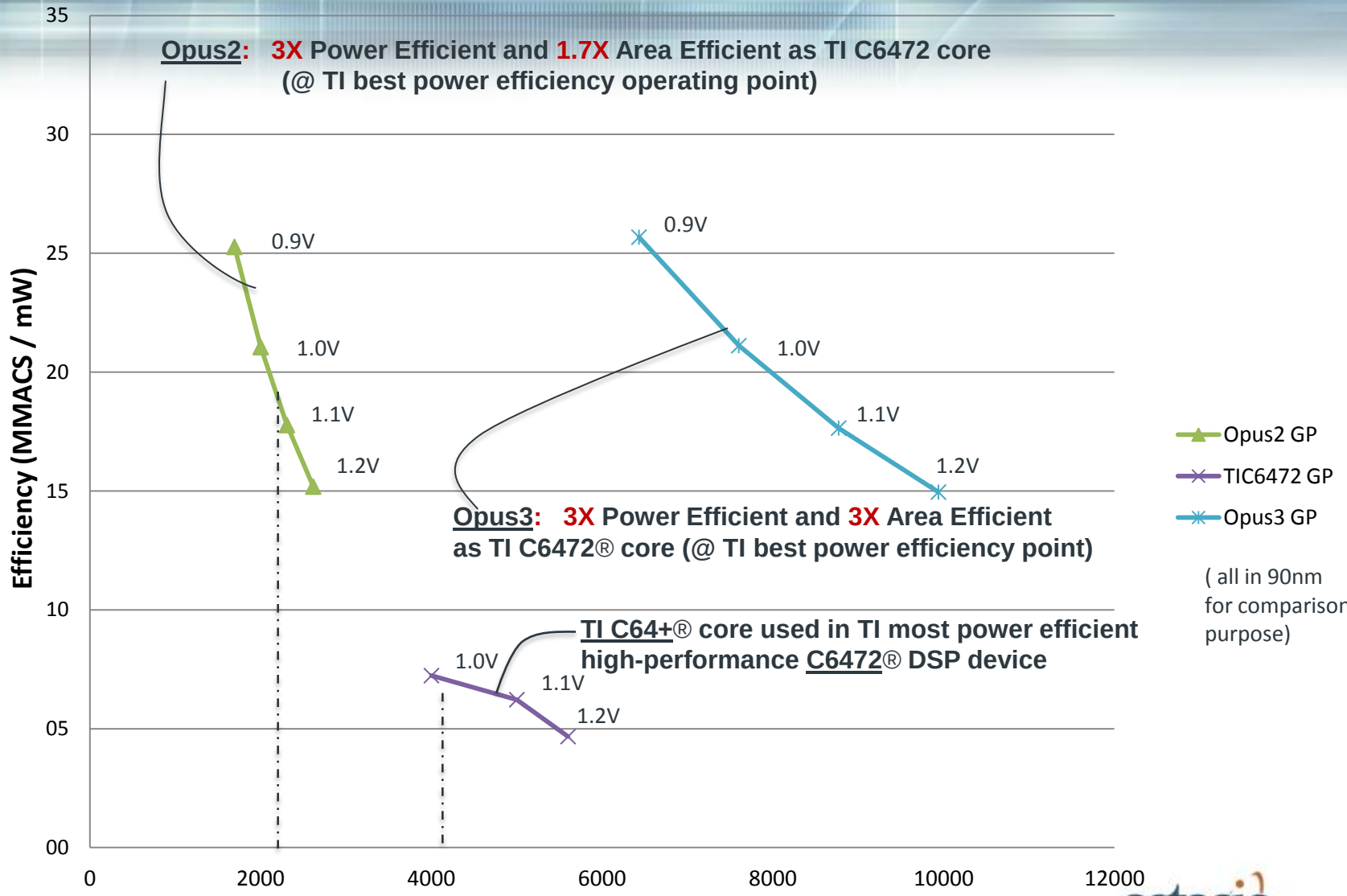


COMPARISON – POWER EFFICIENCY



*It is understood that any such data and comparison is never totally accurate and can be subject to many interpretations. The data is therefore provided for discussion only.

COMPARISON – POWER EFFICIENCY



*It is understood that any such data and comparison is never totally accurate and can be subject to many interpretations. The data is therefore provided for discussion only.

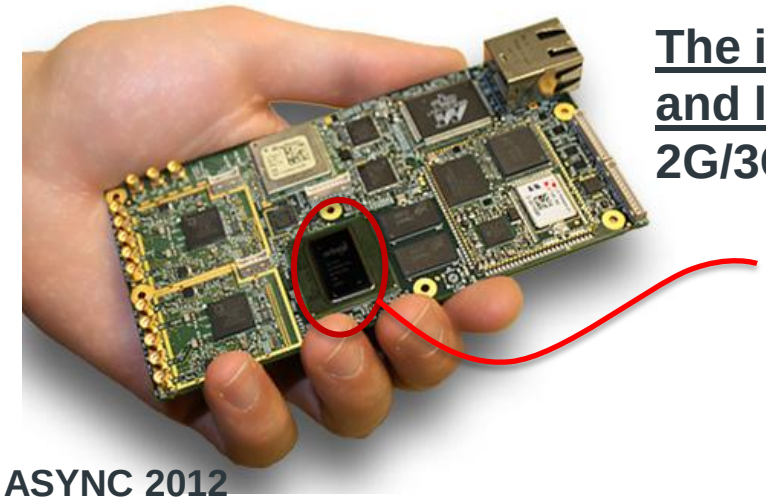
CONTENTS

- Background
- Asynchronous Circuits Description
- Processor Architecture and Operation
- Performance Analysis
- **Conclusion**

CONCLUSION

- **Asynchronous technology does works!**
  - not only in the universities and labs, but
  - in real-life commercial products used by people worldwide

- **Asynchronous technology can be quite advantageous!**
  - **area** efficiency wise,
    ....but more importantly...
  - **power efficiency** wise
    - in the DSP processor market: ~3X more than equivalent synchronous products
    - **same for other processors and datapath engines**



The industry smallest and lowest power 2G/3G/4G basestation

...powered by an OCT2224 Async DSP

Thank you!

Michel Laurence
michel.laurence@octasic.com