A Low-Latency, Energy-Efficient L1 Cache Based on a Self-Timed Pipeline

Louis-Charles Trudeau[1] Ghyslain Gagnon[1]
François Gagnon[1] Claude Thibeault[1] Thomas Awad[2]
Doug Morrissey[2]

[1]École de technologie supérieure, Montréal, Canada
[2]Octasic Inc., Montréal, Canada

21st IEEE International Symposium on Asynchronous Circuits and Systems, 2015

Plan

Research Program

Objective: Adapting Octasic's power-efficient asynchronous architecture in a general purpose processor (ARM v7-A).

Collaborators

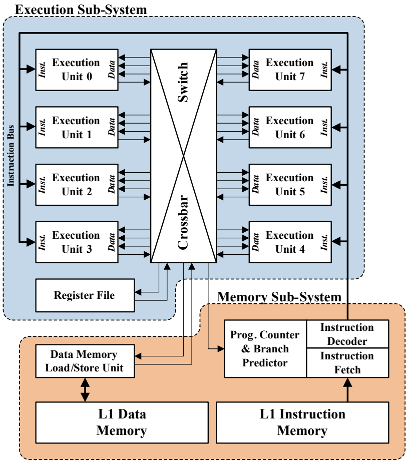
octasic

ÉTS
Le génie pour l'industrie

POLYTECHNIQUE
MONTRÉAL

Problematic

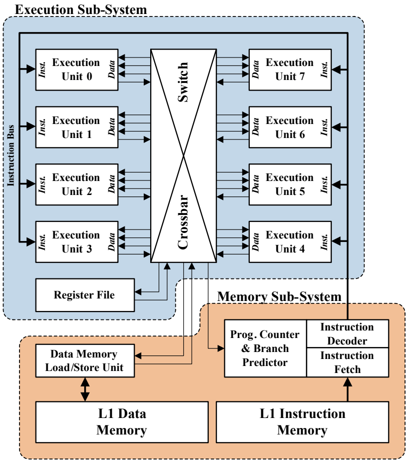
Current architecture separates the asynchronous CPU from the synchronous L1 memory.


Execution Sub-System
Instruction Bus
Inst. Execution Unit 0 Data
Inst. Execution Unit 1 Data
Inst. Execution Unit 2 Data
Inst. Execution Unit 3 Data
Data Execution Unit 7 Inst.
Data Execution Unit 6 Inst.
Data Execution Unit 5 Inst.
Data Execution Unit 4 Inst.
Switch
Crossbar
Register File
Memory Sub-System
Prog. Counter & Branch Predictor
Instruction Decoder
Instruction Fetch
Data Memory Load/Store Unit
L1 Data Memory
L1 Instruction Memory

Problematic

Current architecture separates the asynchronous CPU from the synchronous L1 memory.

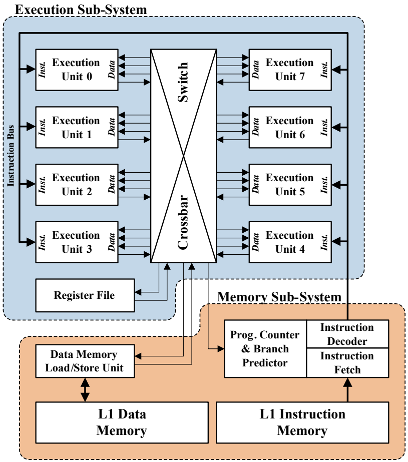
- **2-cycle synchronization penalty**.



Problematic

Current architecture separates the asynchronous CPU from the synchronous L1 memory.

- 2-cycle synchronization penalty.
- **Energy efficiency is suboptimal.**



Motivations

This work focuses on improving the L1 memory access.

Why Go Asynchronous ?

- No balanced clock trees.
  Clocks are point-to-point and skew insensitive.
- No major critical path due to frequency constraints.
  Less large/leaky gates.
- Less complex pipeline structure.
  Only neighboring stages are connected together.

Scope of Work

Design an asynchronous cache based on a self-timed pipeline.

Objectives

1. Mitigate CPU $\leftrightarrow$ L1 memory access latency.
2. Reduce the average memory access time.
3. Improve the cache energy efficiency.
4. Push the synchronization barrier at the L2 memory.

Plan

L1 Instruction Cache Design

Dual-fetch, 32kB, 4-way set-associative phased-cache.

Synchronous Cache

- 5-stage pipeline (hit).
- Pipeline stall on miss.
- 2-cycle pipeline reinjection following cache fill.

Asynchronous Cache

- 4-stage pipeline (hit)
- Single stage stall on miss.
- Resource arbitration for concurrent cache fill.

Integration in ARM-like processor

$\Rightarrow$ Dhrystone & Coremark (armcc compiled).

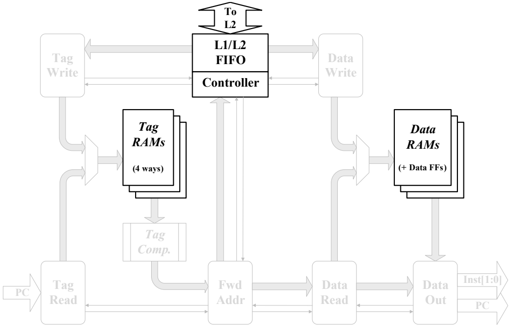
Cache Pipeline

Shared Resources

- **Tag Memory**
- **Data Memory**
- **(*L2 Memory*)**

Tasks Partitioning

- 6 pipeline stages.
- Two-phase handshake protocol.



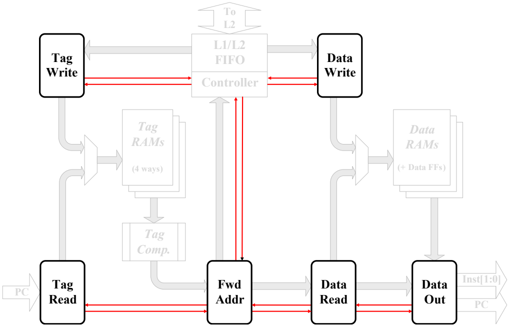
Cache Pipeline

Shared Resources

- Tag Memory
- Data Memory
- (*L2 Memory*)

Tasks Partitioning

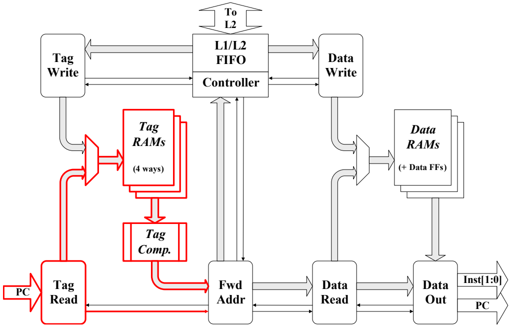
- **6 pipeline stages.**
- Two-phase handshake protocol.



Cache Operation

Pipeline Stages

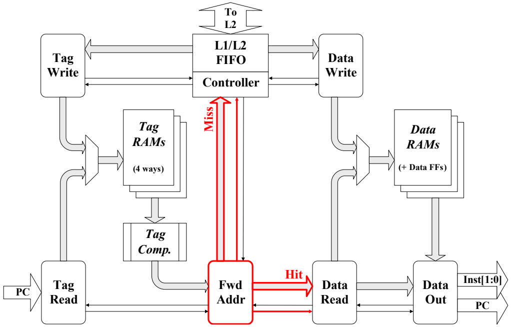
- Tag Read
- Forward Address
- Tag & Data Write
- Data Read
- Data Output



Cache Operation

Pipeline Stages

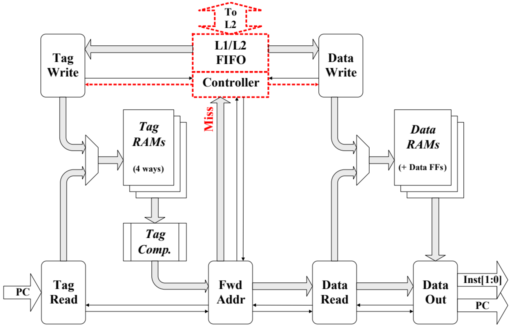
- Tag Read
- Forward Address
- Tag & Data Write
- Data Read
- Data Output



Cache Operation

Pipeline Stages

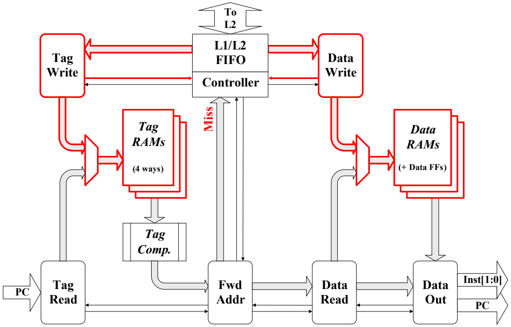
- Tag Read
- Forward Address
- Tag & Data Write
- Data Read
- Data Output



Cache Operation

Pipeline Stages

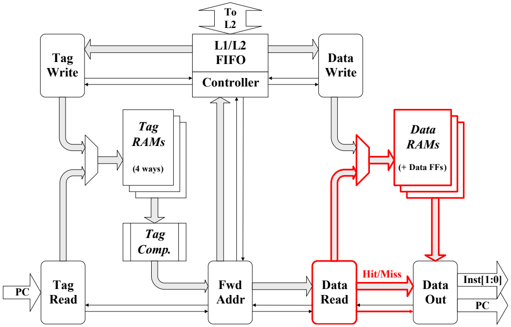
- Tag Read
- Forward Address
- Tag & Data Write
- Data Read
- Data Output



Cache Operation

Pipeline Stages

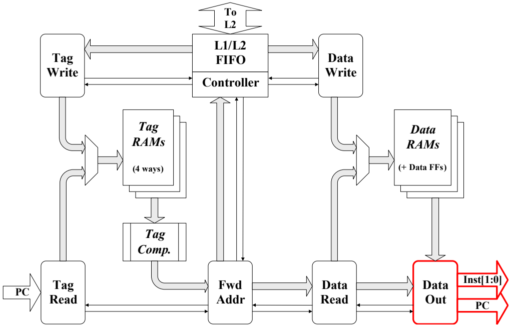
- Tag Read
- Forward Address
- Tag & Data Write
- Data Read
- Data Output



Cache Operation

Pipeline Stages

- Tag Read
- Forward Address
- Tag & Data Write
- Data Read
- Data Output



Plan

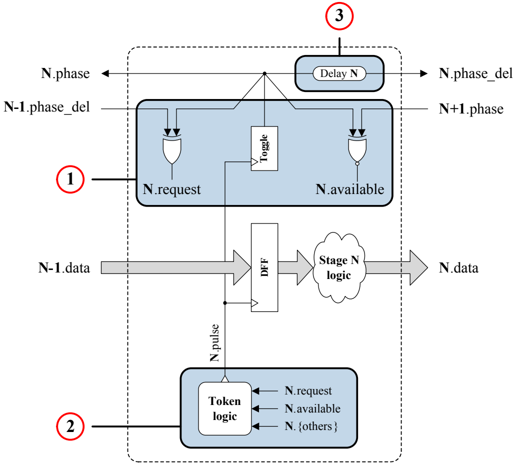
Design Guidelines

Self-timed pipeline design had to follow these guidelines:

- Standard cell libraries.
- Standard edge-triggered flip-flops.
- Prioritize High-Voltage Threshold (HVT) cells to limit leakage.

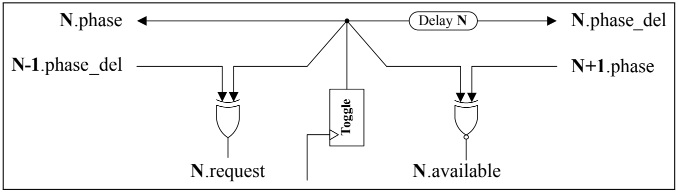
Overview Of A Single Pipeline Stage

1. Click controllers
   - Two-phase handshake protocol.
2. Token modules
   - Synchronization;
   - Pulse generation.
3. Adjustable delays



Click Controllers

- Based on a two-phase handshake protocol.
- Stores the stage phase, toggles it upon usage.



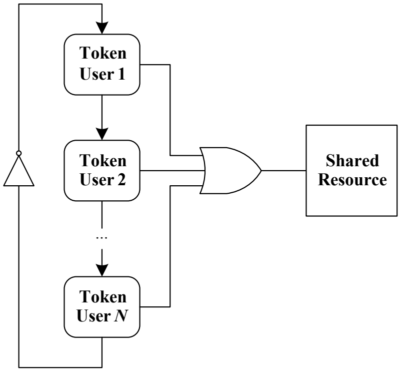
Control Signals

- Request: $N - 1.\mathrm{phase}_{\mathrm{del}} \oplus N.\mathrm{phase}$
- Available: $\overline{N.\mathrm{phase} \oplus N + 1.\mathrm{phase}}$

Token Modules

General Idea

To enable a transaction with a specific resource, “users” must possess the resource’s token.



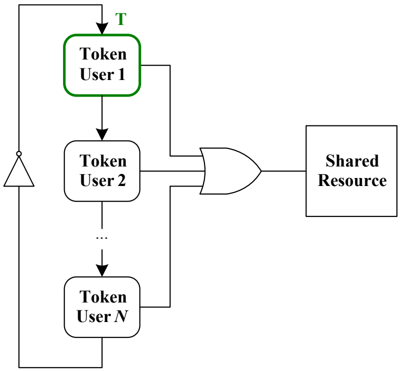
Token Modules

General Idea

To enable a transaction with a specific resource, “users” must possess the resource’s token.

Operation

1. Hold the token



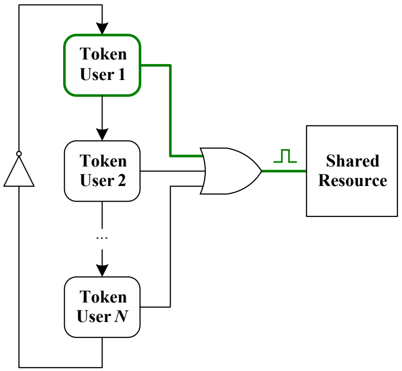
Token Modules

General Idea

To enable a transaction with a specific resource, “users” must possess the resource’s token.

Operation

1. Hold the token
2. Consume the token and access resource



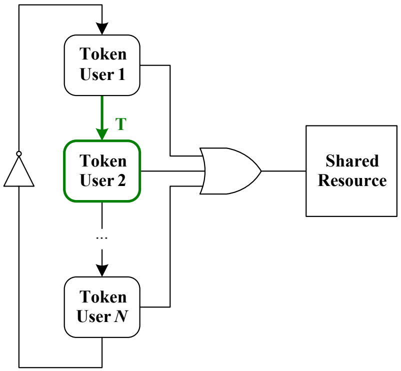
Token Modules

General Idea

To enable a transaction with a specific resource, “users” must possess the resource’s token.

Operation

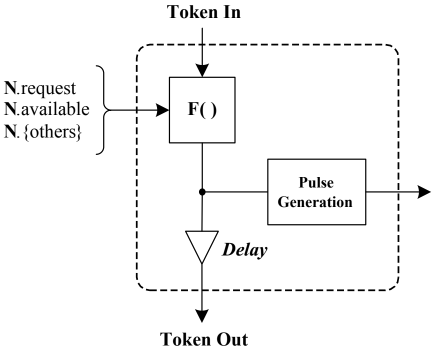
1. Hold the token
2. Consume the token and access resource
3. Pass the token



Token Modules

Internal Structure

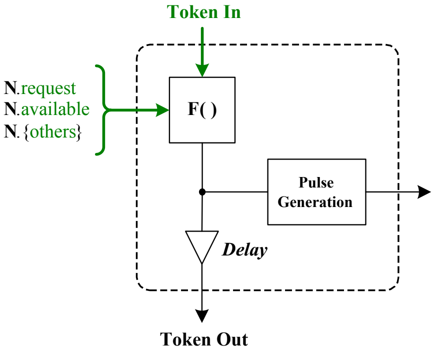
- Token control function
- Pulse generation
- Token ring delay



Token Modules

Operation

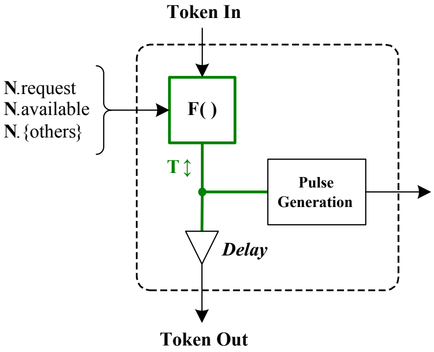
1. Token conditions are met.



Token Modules

Operation

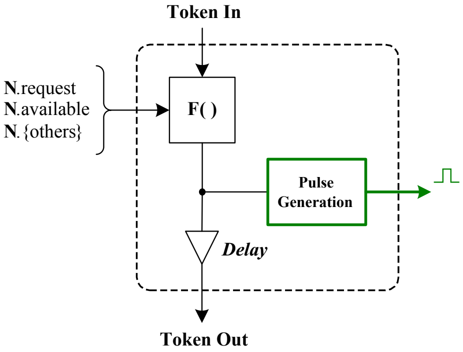
1. Token conditions are met.
2. Token passes through F() thus causing a transition.



Token Modules

Operation

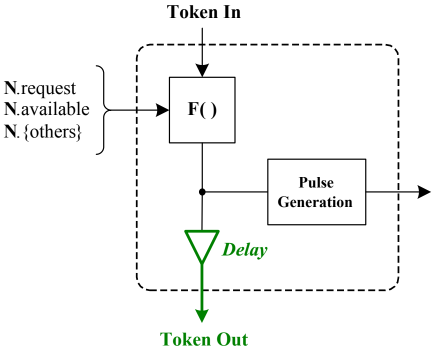
1. Token conditions are met.
2. Token passes through F() thus causing a transition.
3. Transition (edge) generates a pulse signal.



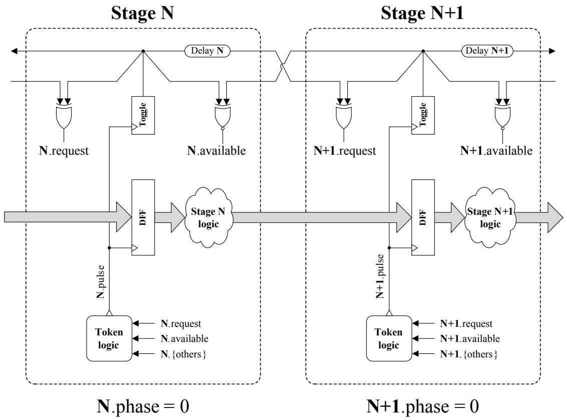
Token Modules

Operation

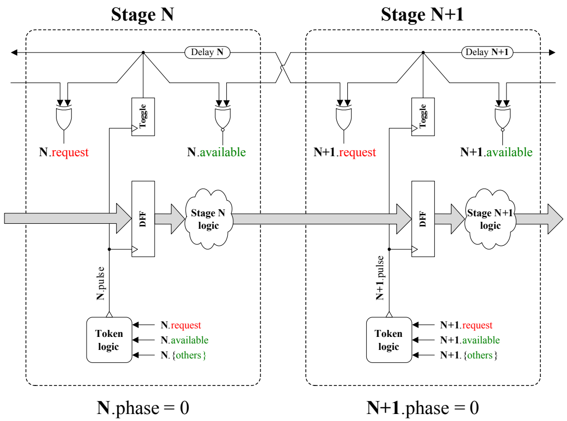
1. Token conditions are met.
2. Token passes through F() thus causing a transition.
3. Transition (edge) generates a pulse signal.
4. Token is delayed, then passed to next user.



Self-Timed Pipeline Operation

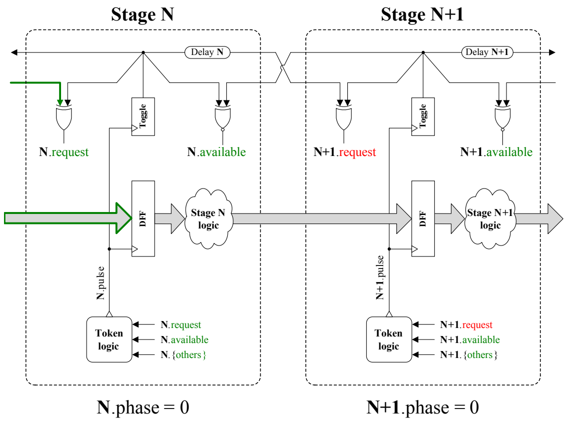


Self-Timed Pipeline Operation



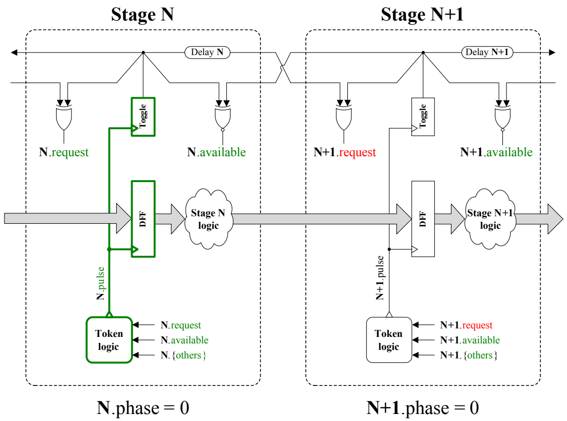
1. Initialization: all stages are available.

Self-Timed Pipeline Operation



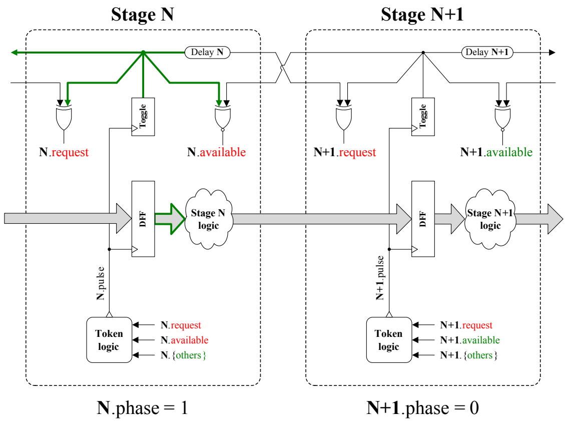
2. Request in: stage N conditions are met.

Self-Timed Pipeline Operation



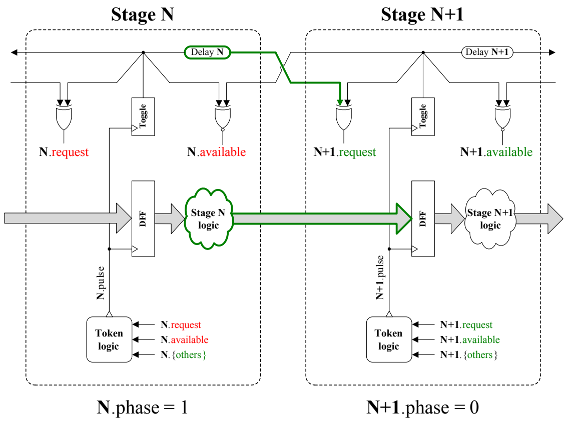
3. Pulse generation: N flops input data & toggles phase.

Self-Timed Pipeline Operation



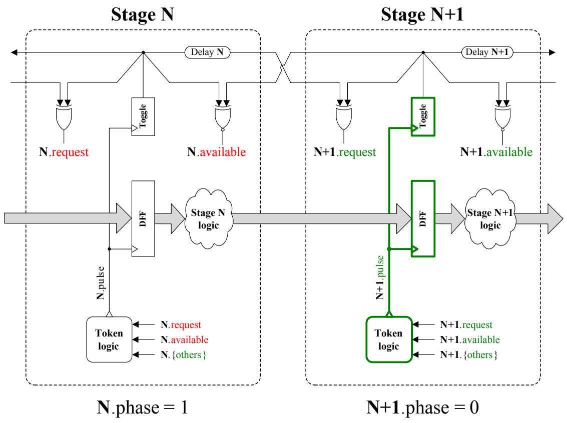
4. Processing: stage N is used, therefore unavailable.

Self-Timed Pipeline Operation



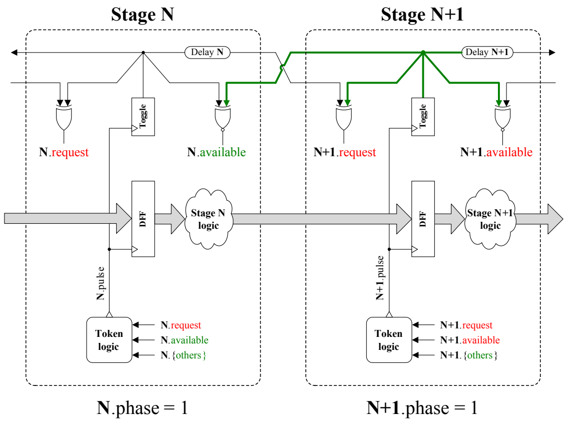
5. Request in: delayed N phase triggers stage N+1 request.

Self-Timed Pipeline Operation



6. Pulse generation: N+1 flops input data & toggles phase.

Self-Timed Pipeline Operation



7. Stage N is now available, N+1 processes data.

Plan

Extracting Results

Objective

Use synchronous instruction cache for baseline results.

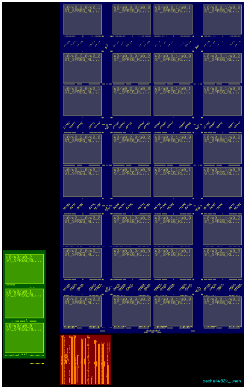
$\Rightarrow$ Keep very similar silicon layout for proper comparison.

Performance Metrics

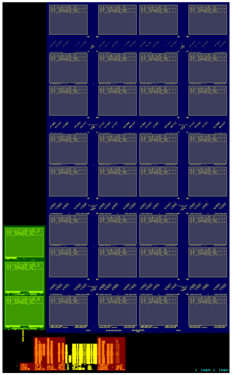
- Average memory access time
- Energy consumption
- Area

Results: Area

Synchronous



Asynchronous



Results: Area

	Synchronous		**Asynchronous**	
Pipeline	Area (μm^2)	(%)	Area (μm^2)	(%)
Total	11185	100	9900	100
(w/o L2 FIFO)	8455	75,6	7170	72,4
(control)	375	3,35	915	9,24

Table: Pipeline Size Comparison

$\Rightarrow$ Pipeline size reduced by : 10-15%.
$\Rightarrow$ Pipeline control is $\sim 2.5\times$ larger.

Results : Energy

Power Analysis

- Power estimations based on capacitive switching.
- Routing estimated from Manhattan distance.
- L1-L2 interface clock frequency matched.

L1 Cache behavior tests (32kB program)

1. Miss : 20k random instructions fetch;
2. Hit : 200k, 2M, 20M random instructions fetch.

Results : Energy

	Synchronous	**Asynchronous**	
Sequence	Energy (nJ)	Energy (nJ)	Δ E $(\%)$
20k inst.	150.6	118.0	21.6
200k inst.	1373.5	1048.3	23.7
2M inst.	13609.1	10357.9	23.9
20M inst.	135917.3	103424.7	24.0

Table: Energy Consumption

$\Rightarrow$ Energy efficiency improved by : $>$ 22%.

Results : Performance

	Synchronous	Asynchronous	
Sequence	Exec. Time (ms)	Exec. Time (ms)	Δ T $(\%)$
20k inst.	43.1	29.5	31.6
200k inst.	358.1	265.8	25.8
2M inst.	3508.1	2628.3	25.1
20M inst.	34360.0	25769.9	25.0

Table: Average Memory Access Time

$\Rightarrow$ Access time reduced by : $> 25\%$.
$\Rightarrow$ Throughput at L1-L2 interface : $> 40\%$.

Results : What needs to be addressed

Future Work

- Further pipeline cache to reach > 1 GHz equivalent.
- Design L1 data cache based on self-timed pipeline.
- Integrate asynchronous L1 caches in Octasic's next-generation processors.

Plan

Summary

Problematic

$\Rightarrow$ Asynchronous CPU accesses synchronous L1 memory.

Goals & Results

Design and implement an asynchronous L1 cache:

1. Mitigate CPU $\leftrightarrow$ L1 memory access latency: ✓
2. Improve the cache energy efficiency: $> 22\%$
3. Reduce the average memory access time: $> 25\%$
4. Push the synchronization barrier at the L2 memory: ✓