PHILIPS

Performance/ Execution Architecture and its impact on the Product Family

SASG Meeting, Februari 2004

Vlatko Milosevski, Ton Kostelijk, PDSL-E

Overview

- Execution Architecture (EA) – focus and goals
  - Consumer Electronics domain specifics
  - EA benefits & focus
- HDD /DVD+RW Product Family
  - Architecture roadmap
  - Misleading trade-offs
- EA Findings & Extrapolation to the future
  - Data transfer Latency Model
  - Early bottleneck prediction & interpolation

CE – domain characteristics

- Large volumes (>million)
- “Squeeze” each CPU cycle
  - Reduce BOM and increase end-user features as much as possible
- Stand-alone, low filed call rate, and fast-responsive systems
- Strong HW-SW interaction
  - HW/SW Trade-off & alignment
  - HW – long lead times, unchangeable
  - SW – follows later, modifiable

Execution architecture - Focus

- Concurrent Timing Requirements
- Bandwidth Requirements
- Results:
  - Task /ISR priorities
  - Bandwidth analysis
  - CPU load analysis
  - Measurements, Worst Case Scenario Study
  - Feasibility study, etc...
  - HW / SW tradeoff

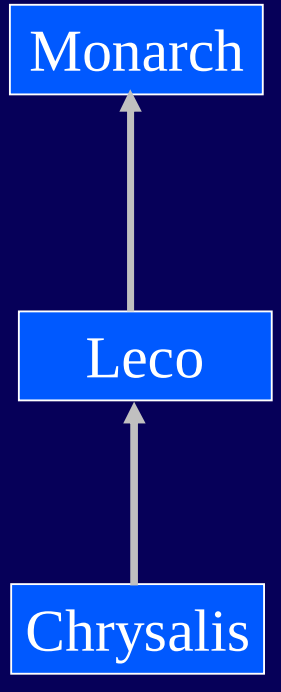
HDD / DVD+RW recorder product family

DXC - HDD / DVD+RW Combi Recorder

HDRW720 - box outlook

HDRW720 - screen outlook

HW Architecture Roadmap

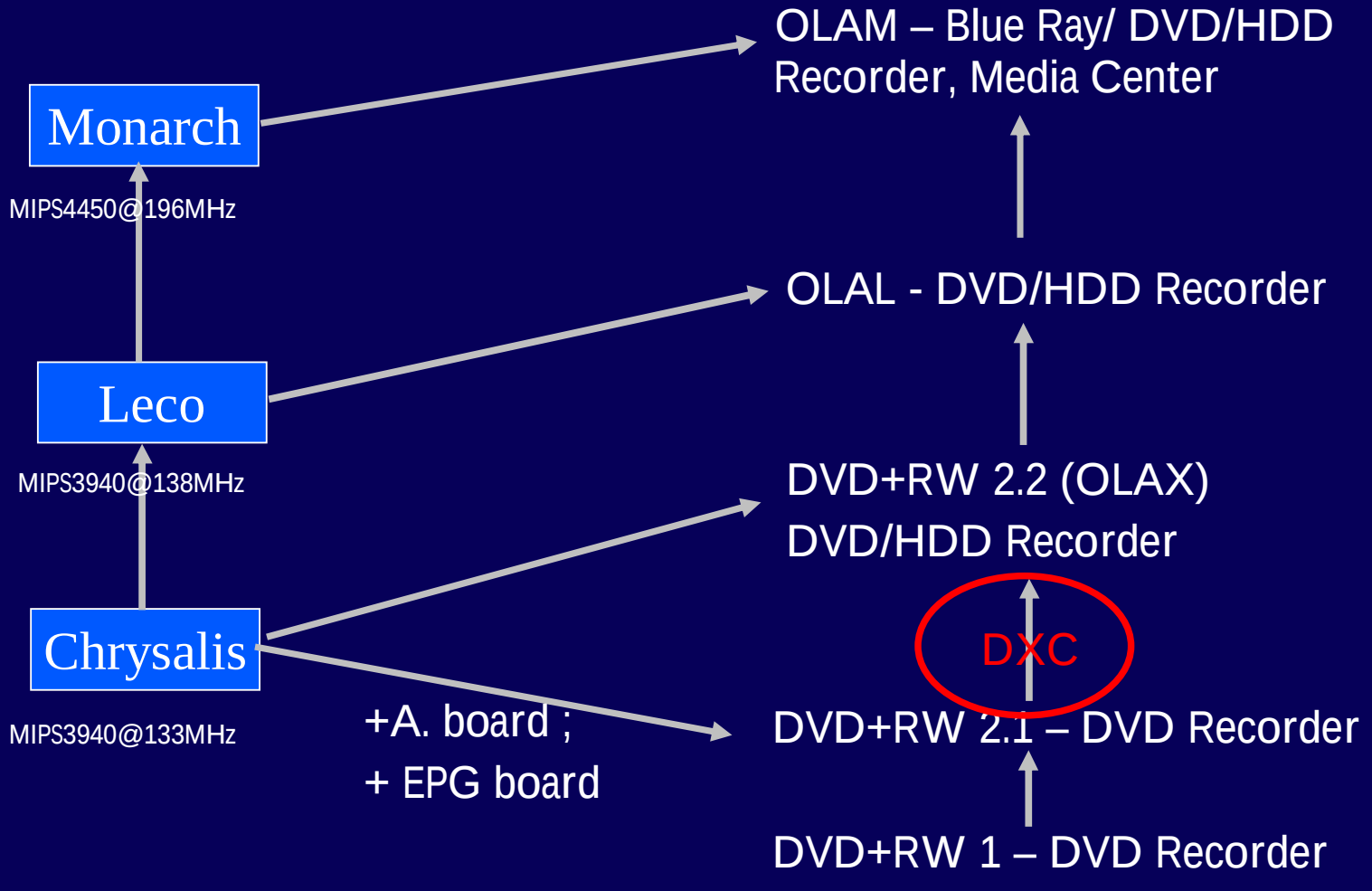


MIPS4450@196MHz
DDR 16bit bus @ 133/166/200MHz: 570MB/s
Cache access: I:32/64 bits; D:16 bits
8 stage pipeline

MIPS3940@138Mhz
16bit SDR bus @ 166MHz: 280MB/s

MIPS3940@133MHz
SDR 32bit bus @ 133MHz: 440MB/s
Cache access: I:32 bits; D:16 bits
6 stage pipeline

SW Architecture Roadmap

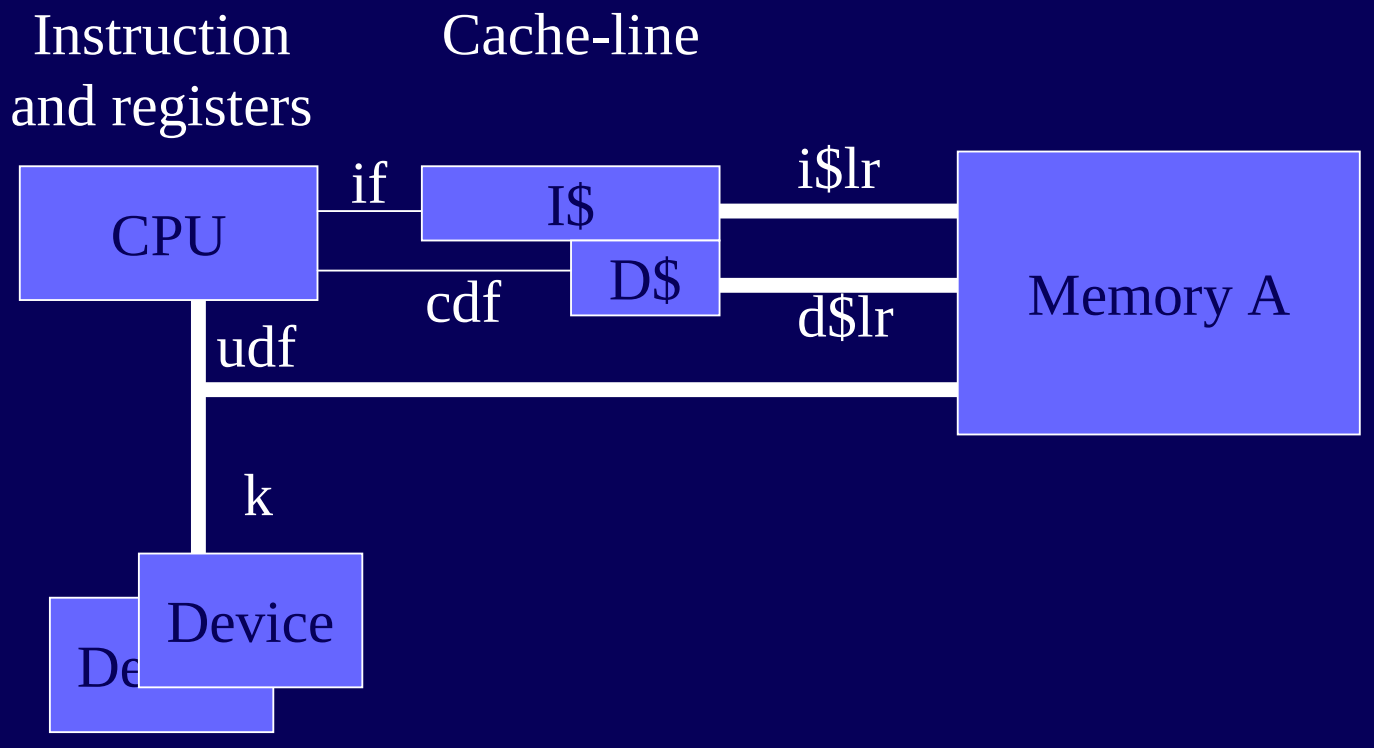


System extension
Challenges and analysis

Typical Problems in extending the functionality

- Responsiveness & feasibility (CPU load)
  - New use-case scenarios running **concurrently** with the previous scenarios
- Memory Bandwidth
- Architectural Extension
  - Synchronization and communication
  - User Interface / system consistency

‘Data’ transfer latency model

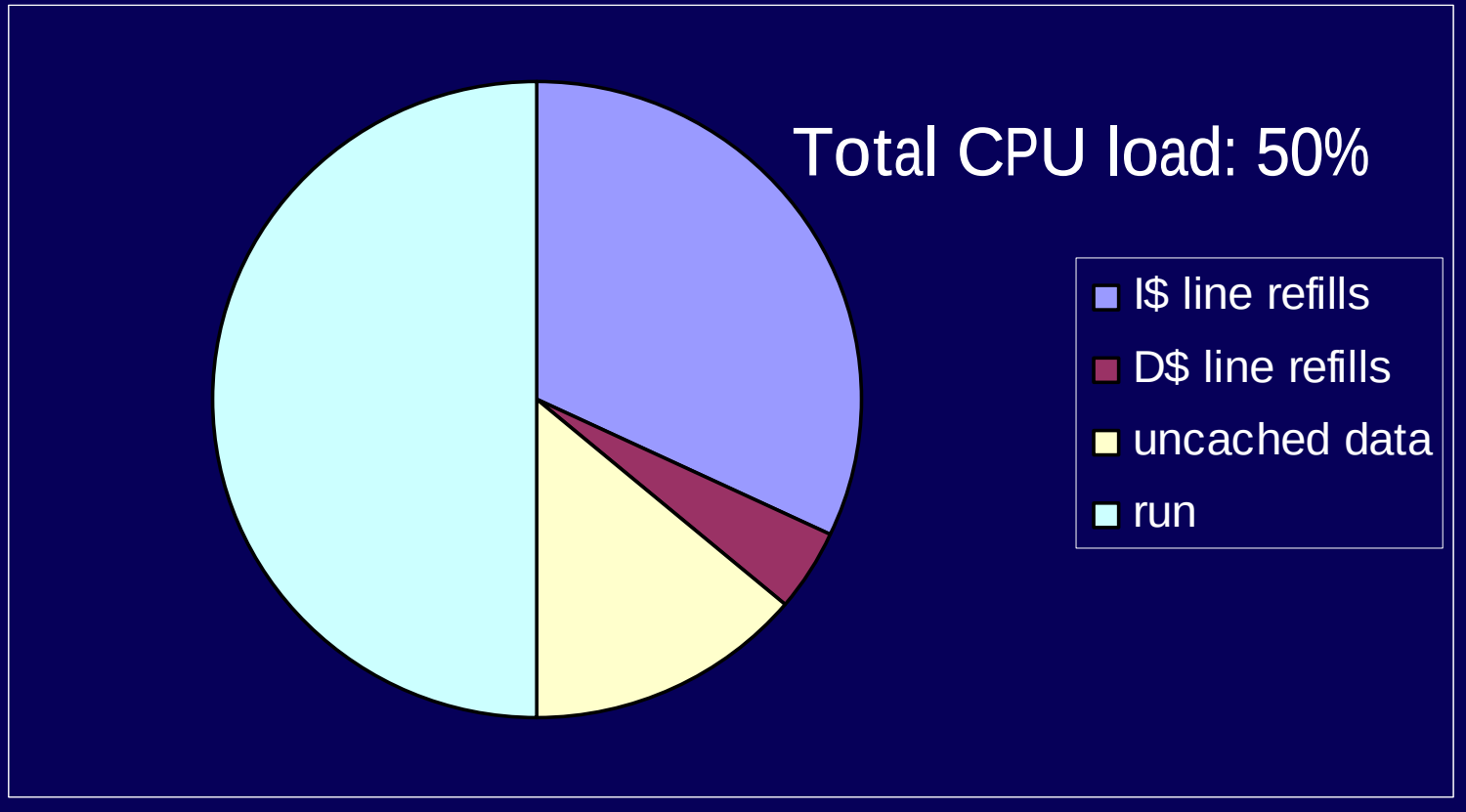


Data fetch characteristics

instruction-**f**etch	**c**ached **d**ata **f**etch **U**n-cached **d**ata **f**etch	Determined by program only. Cache independent.
I$ line **r**efill	**D$ l**ine **r**efill	Determined by program + cache-characteristics.

Roughly: #cycles = if + cdf + (udf + I$lr + D$lr) * $r_latency

Chrysalis CPU Load Analysis

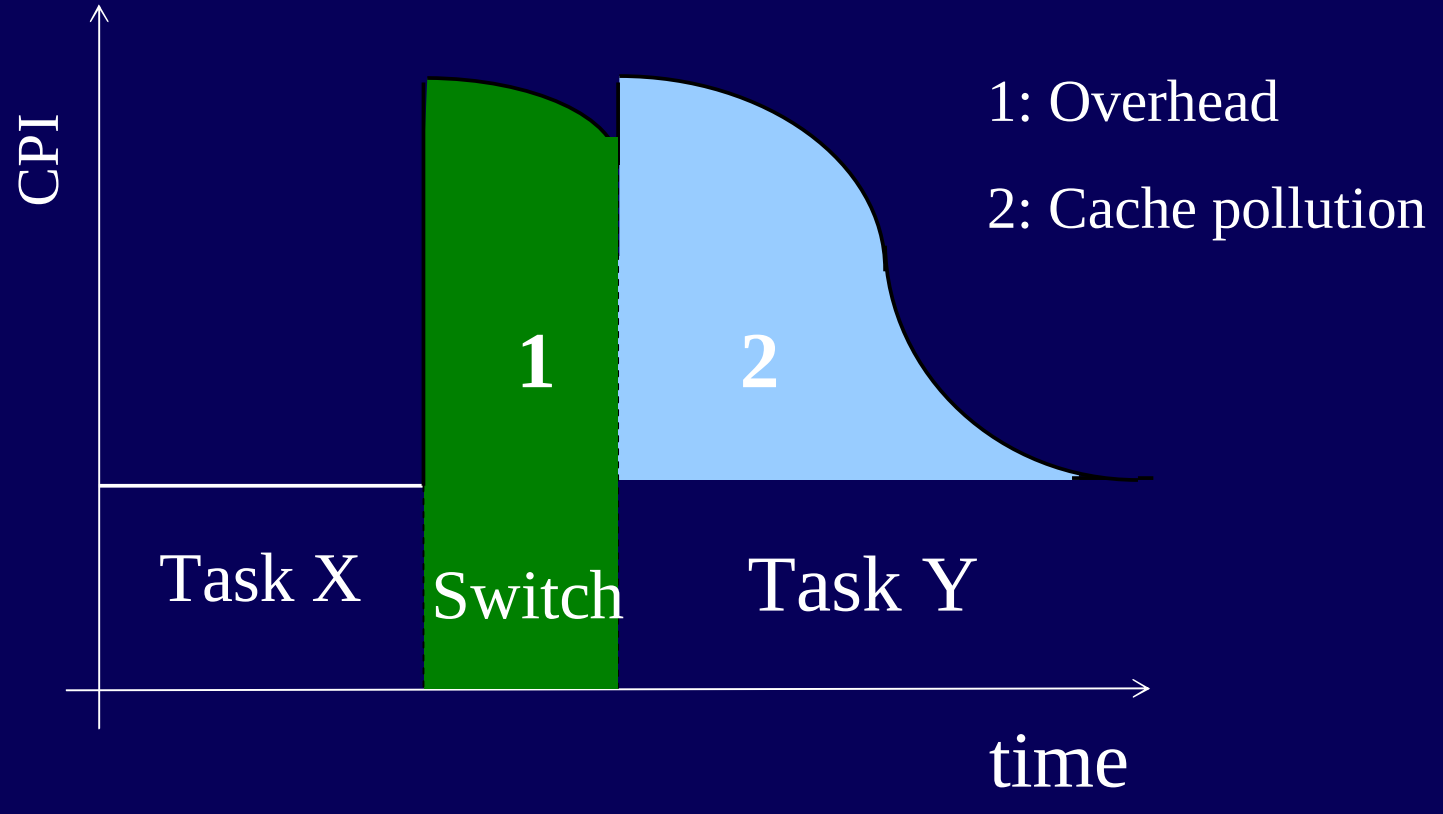


After code non-intrusive optimization

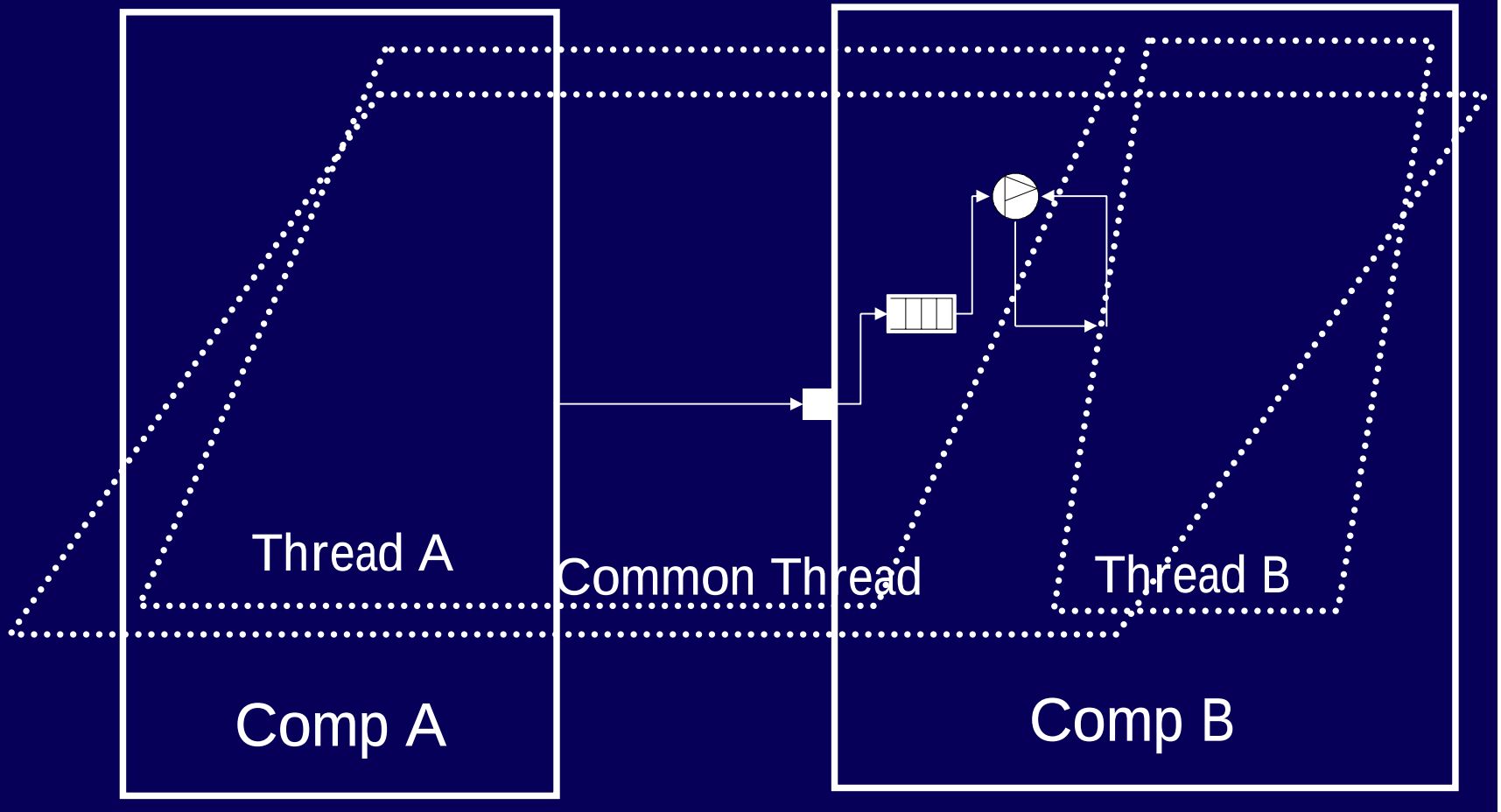
Reducing Memory Fetch Penalty

- Increasing Cache Sizes?
  - Not if there are a lot of threat switching.
- Turn un-cached to cached data?
  - Depends on the application
- Reducing Memory Access Time?
- Memory bus Architecture (64/32/16 bits)?
  - Limited benefit, depends on the latency.
- Reducing code size?
  - Mips16 I.s.o. Mips2, Compiler switches
- Reducing Thread Switches?
  - (example follows)

Thread Switch Effect



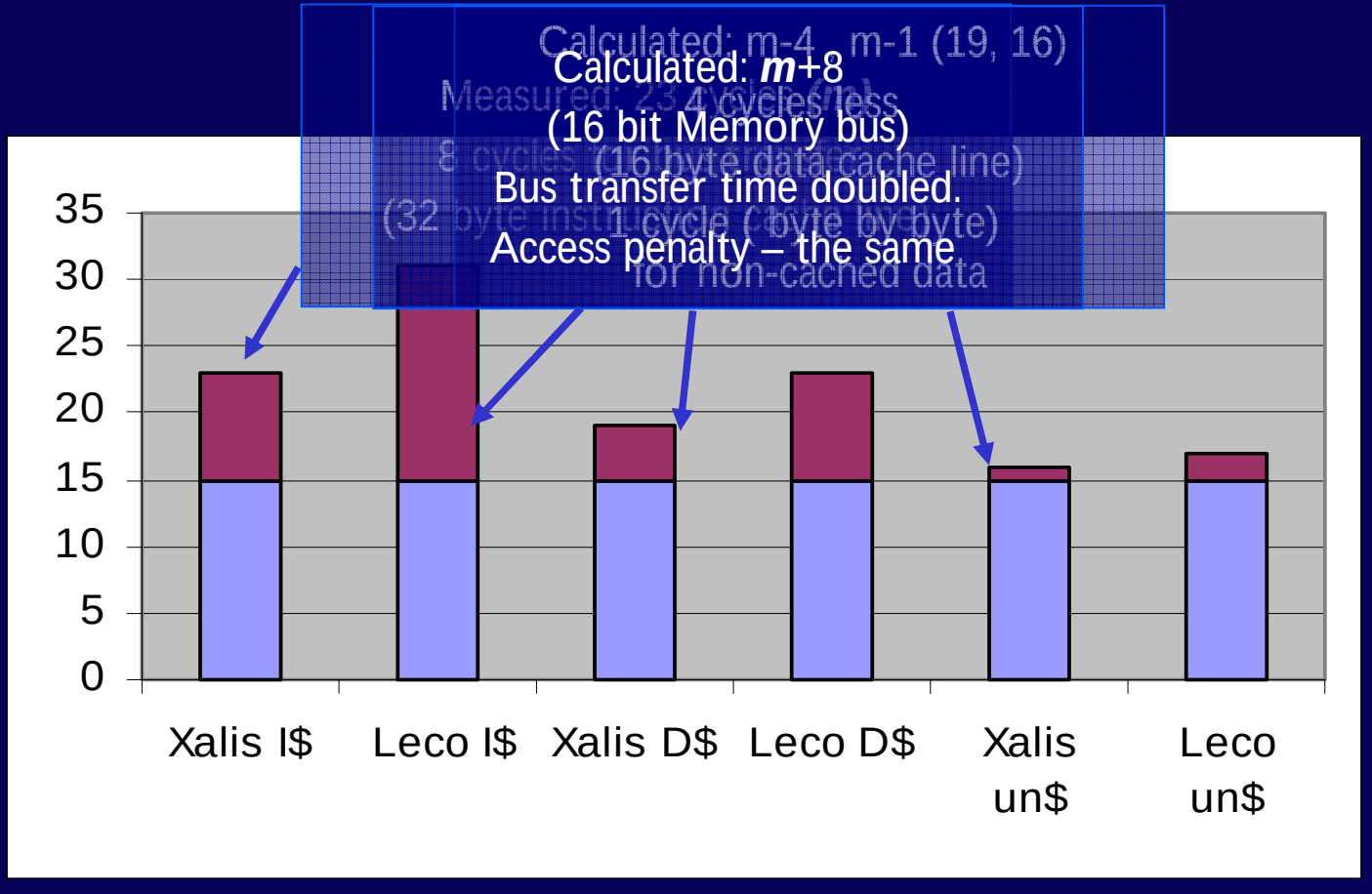
SW Architecture Change



Extrapolation to the future architecture

Extrapolation: LECO v.s. Chrysalis

16 bits SDRAM v.s. 32 bits SDRAM



Extrapolation - LECO

- 16 bit I.s.o. 32 bit memory bus

(31/23 =35%) * (30% or 50%) * (50%)

(I$ cycle increase) (CPU load share) (use-case CPU load)

= 5% to 8.7% use-case load increase

- 166MHz I.s.o. 133MHz memory bus freq, 138Mhz iso 133 MHz CPU frequency

(31/23) * (133 /166) * 50% + (133 /138) * 50%

= (54% + 48%) * 50% = **51 %**

ONLY 1% performance degradation on this use-case scenario!!

LECO Risks

Not Everything is predictable!

- Functionality changes
- SIF arbiter settings – Memory Access penalty
- Bus Load

MONARCH Extrapolation

- 16 bit DDR memory v.s. 32 bit SDRAM

  Net Effect – NO LOAD INCREASE!

- 166 MHz Memory bus + 196MHz CPU

(133 /166) * 50% + (133/196) * 50% =

(40% + 34%) * 50 % = **37%**

50% CPU load scenario on Chrysalis will be 37% CPU load scenario on Monarch

Running two Chrysalis 50% CPU load scenarios concurrently on Monarch is still not feasible!!!

Further improvements are needed

Thank You!