

DMA

- Lab 3
 - DMA
 - The task
 - (Bursts)
- MMU, soft CPUs

Guest lecture

Time: Friday 5/12 1515-1600.

Place: Nollstället

Image Processing on FPGAs.

Johan Pettersson, Sick IVP

Packed array

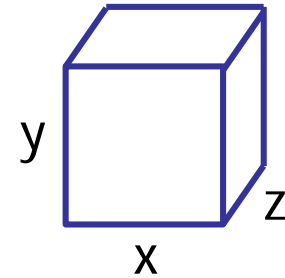
left to right, right first



```
logic [11:0] tm1[0:7][0:7];
```




```
logic [0:7][0:7][11:0] tm2;
```



```
tm1[0][0] // DC component
```

```
tm2[0][0] // "-"
```

```
tm2[0] // 
```

```
tm2[0:7][0] // 
```

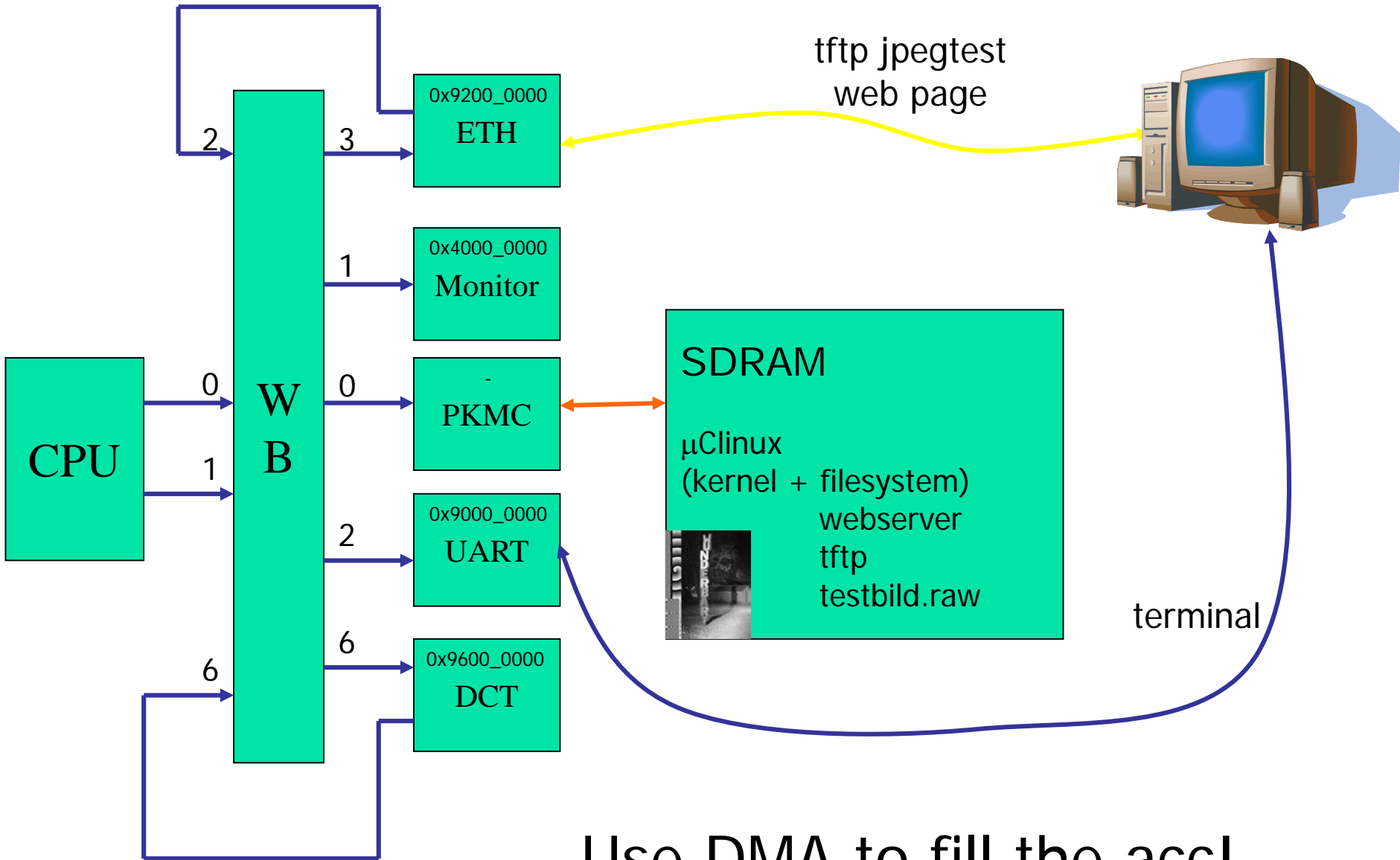
Comment: Array slicing

The size of the part select or slice must be constant, but the position can be variable.

```
logic [31:0] b;  
logic [7:0] a1, a2;
```

```
a1 = b[x -: 8];           // OK fixed width  
a2 = b[y +: 8];         // OK fixed width  
d = b[x:y];             // not OK
```

Lab 3 - DMA



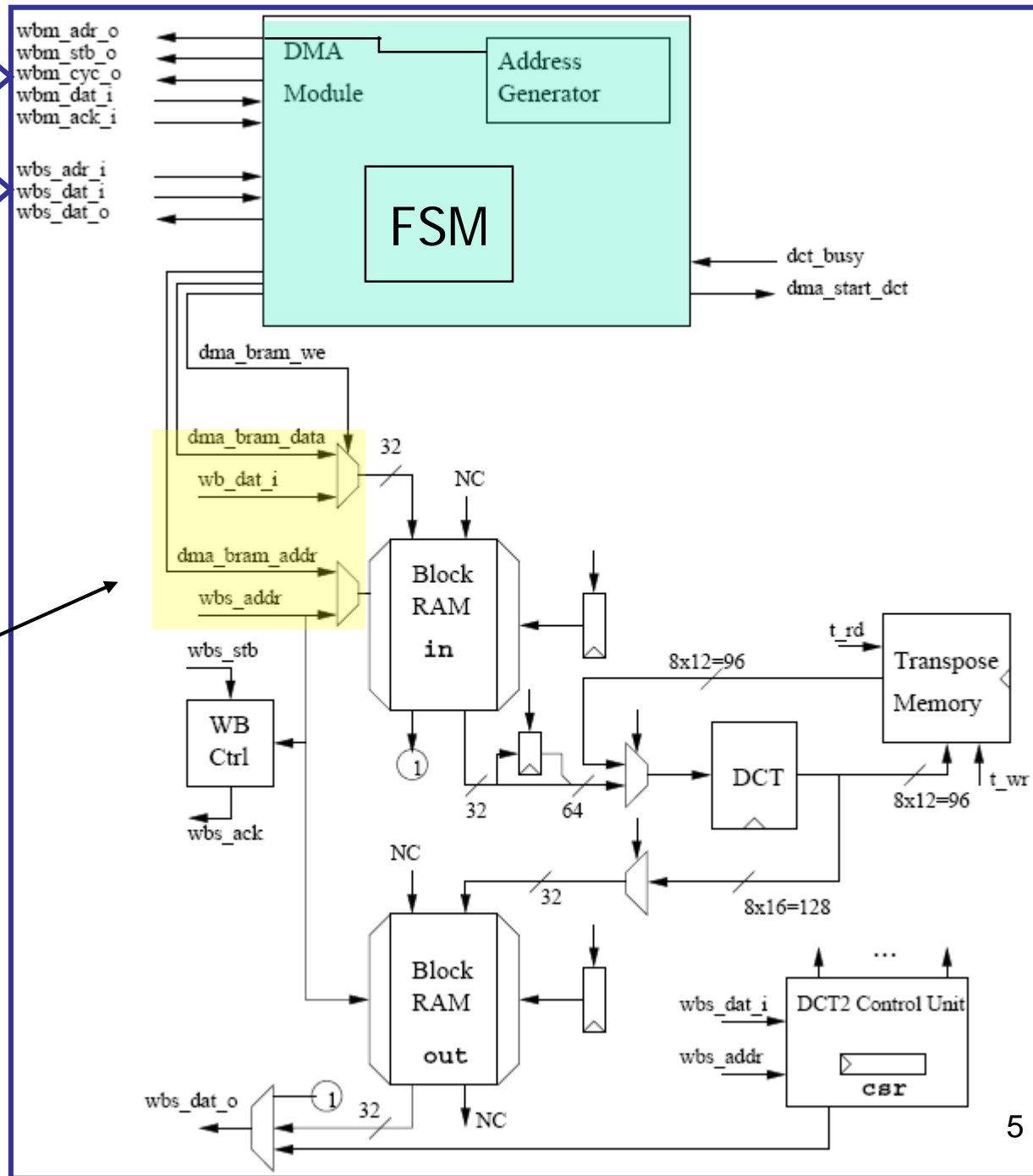
Use DMA to fill the acc!

wbm

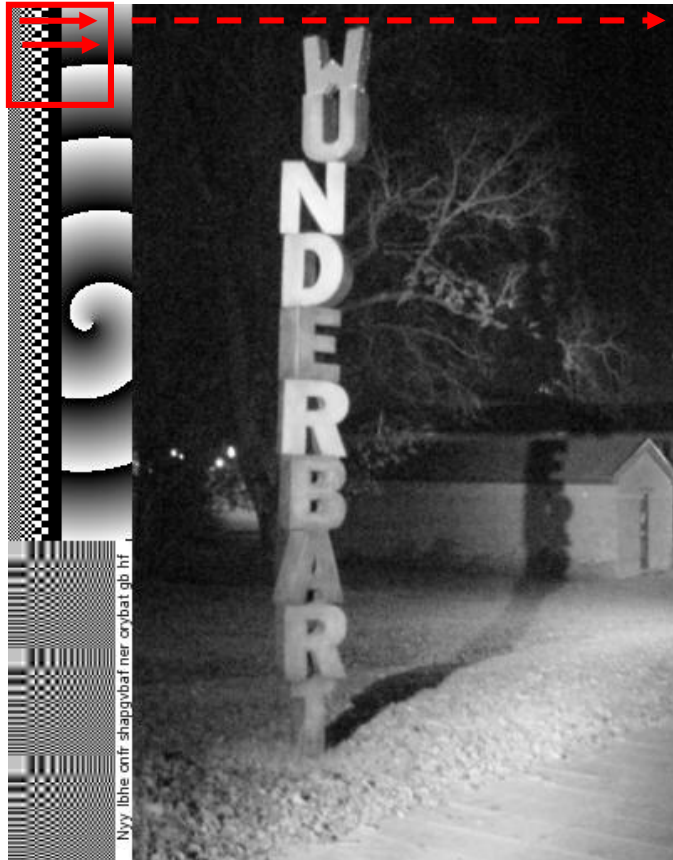
wbs

Proposed architecture

- Design FSM
- Change here
- Modify jpegfiles



Address generation



testbild.raw

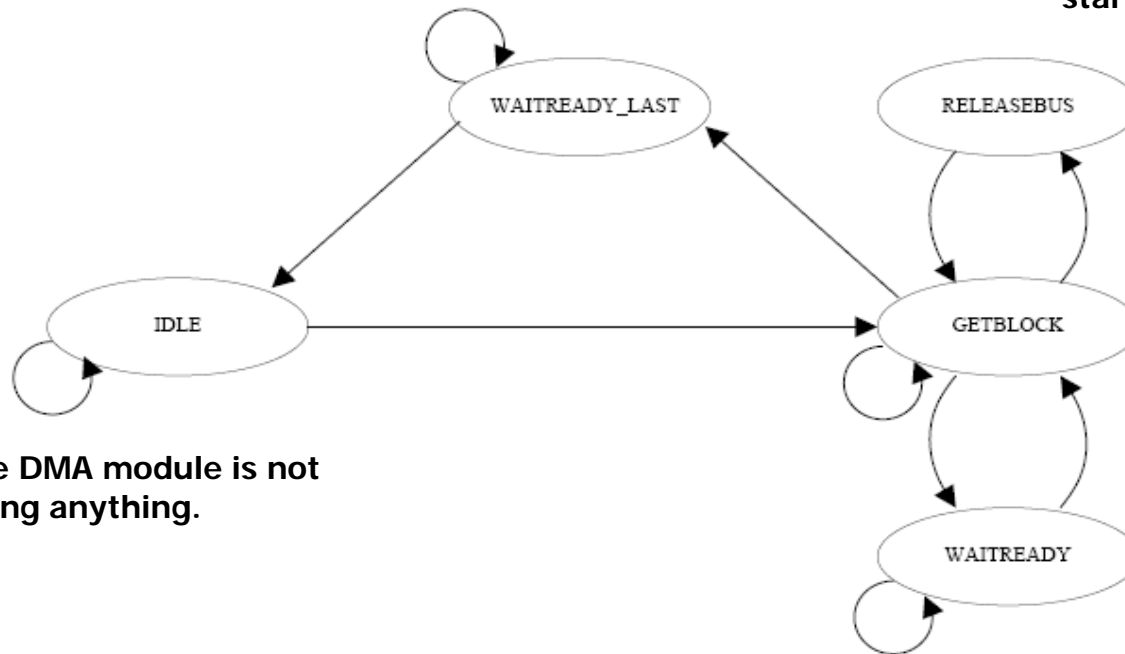
- We want to transfer block by block (8x8)
- Address generator must know format (width,height) of image

State diagram



Same as WAITREADY except that we go to the IDLE state when done.

The DMA accelerator has to release the bus regularly so that other components can access it. Do it for every line you read. When we finish the first block, we start the DCT accelerator.



The DMA module is not doing anything.

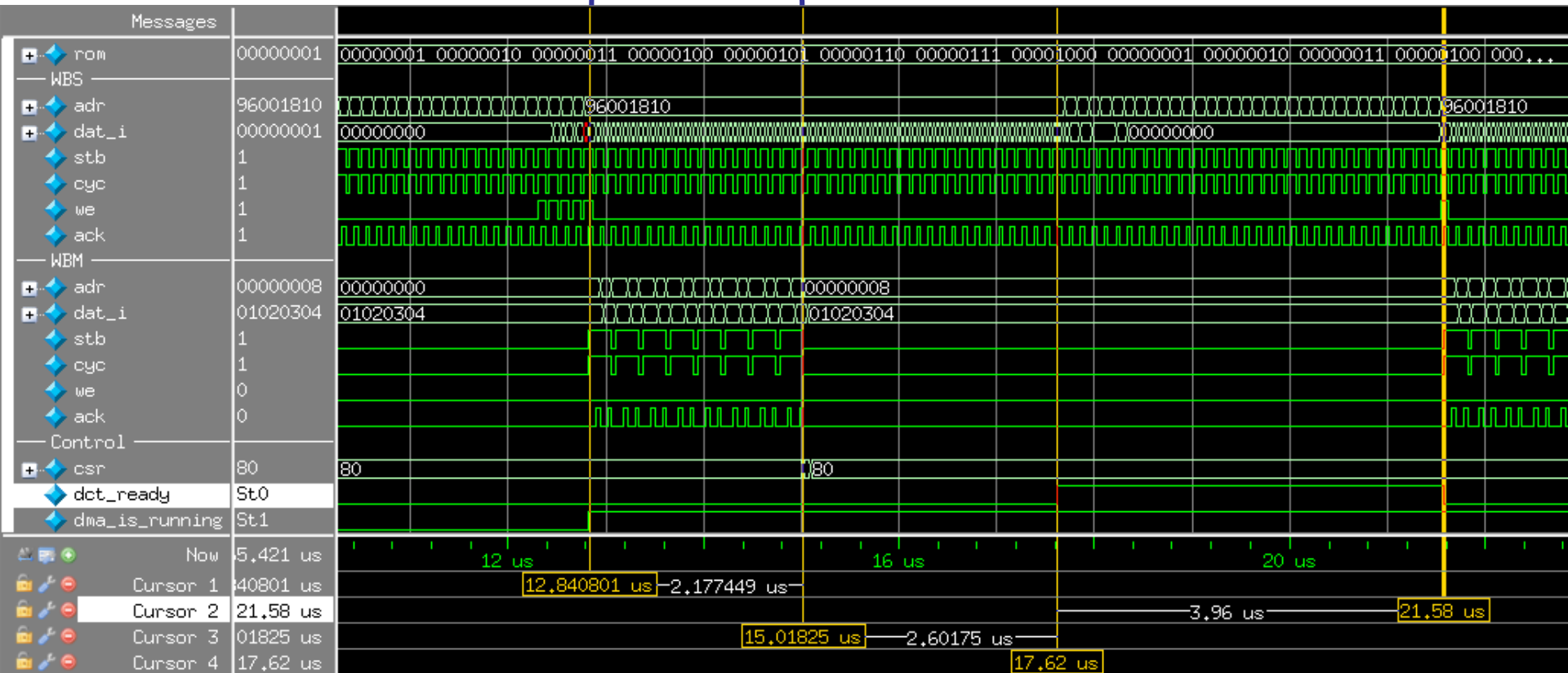
The DMA module is fetching an 8x8 block. Once the block is fetched we go to the WAITREADY state and start the DCT transform.

In this state we wait until the program tells us that it has read the result of the transform by writing to the control register.

A measurement make sim_jpeg

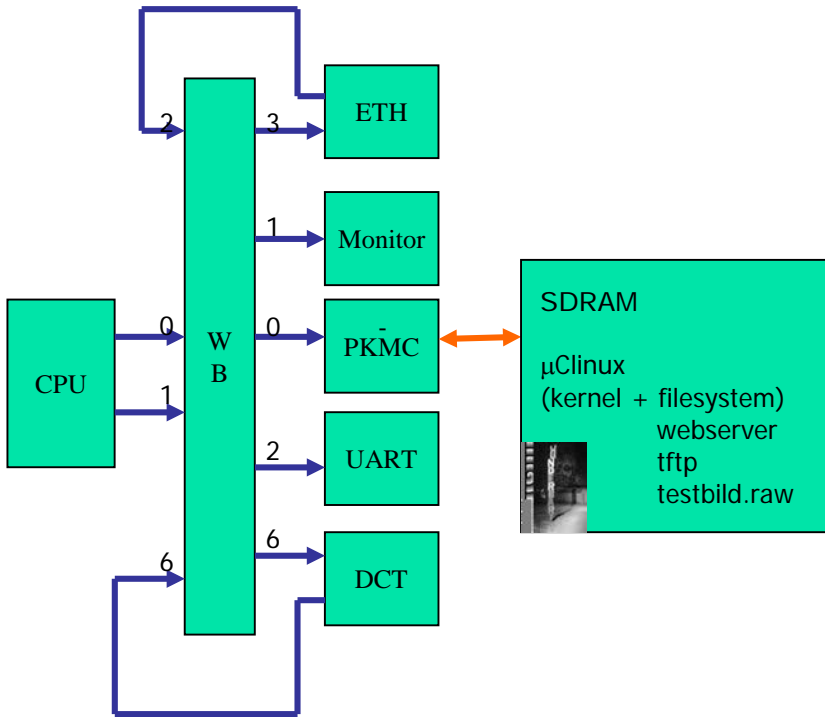
Copy 16 words
from **SDRAM**
to **DCT** (DMA)

Copy 32 words
from **DCT**
to **SDRAM**

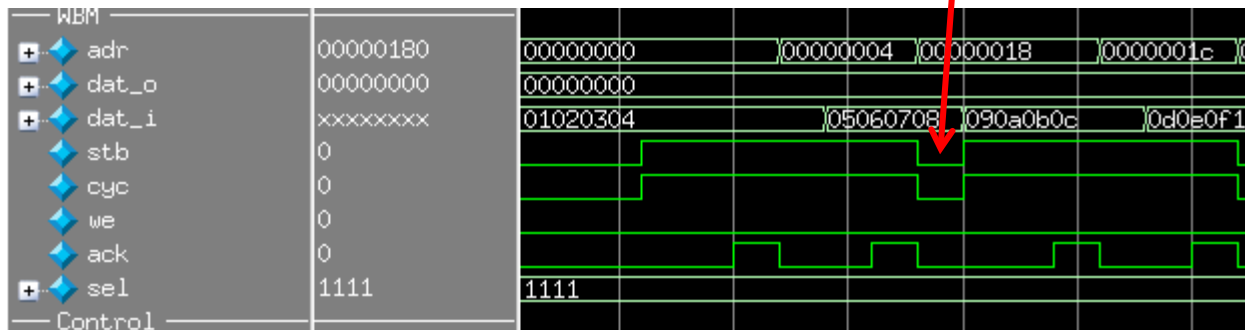


DCT

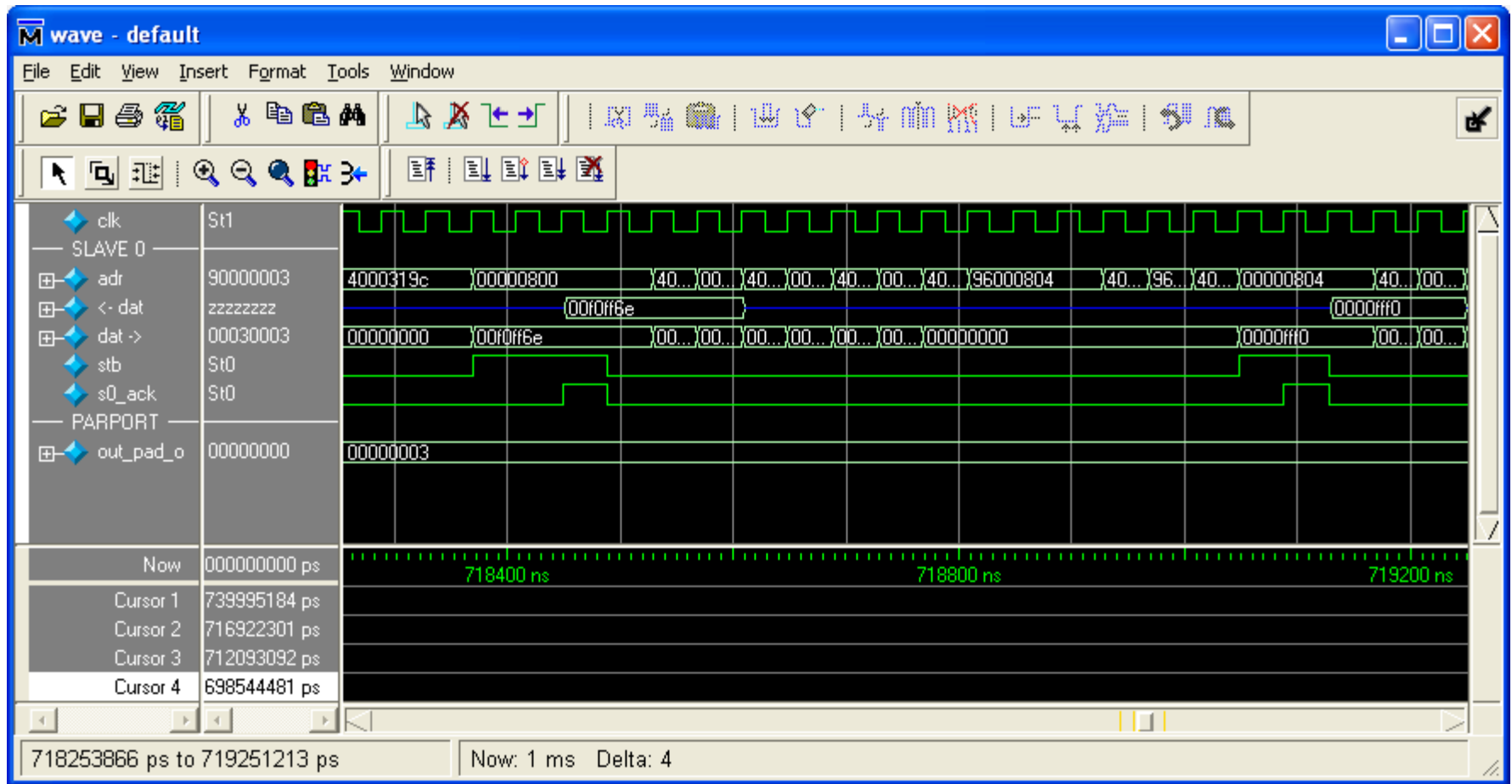
A closer look at the DMA



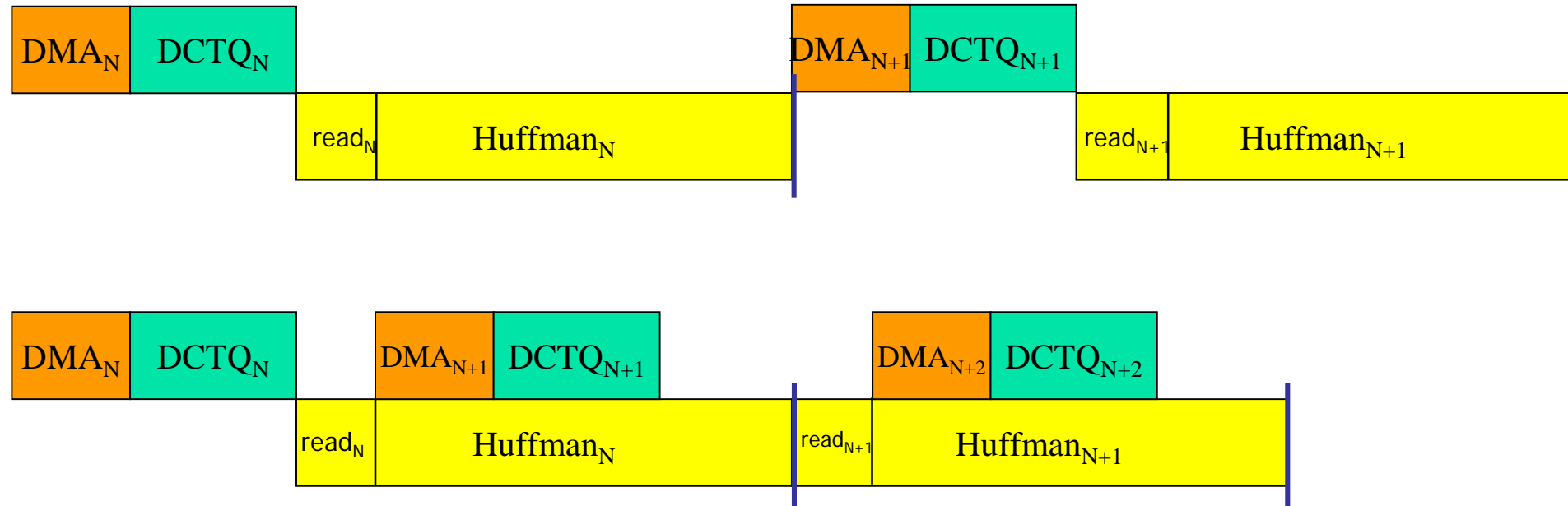
Release bus for
m0, m1, m2
⇒ If CPU is waiting it will
get the bus



DCT => Mem (Software)

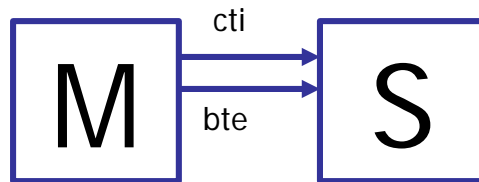
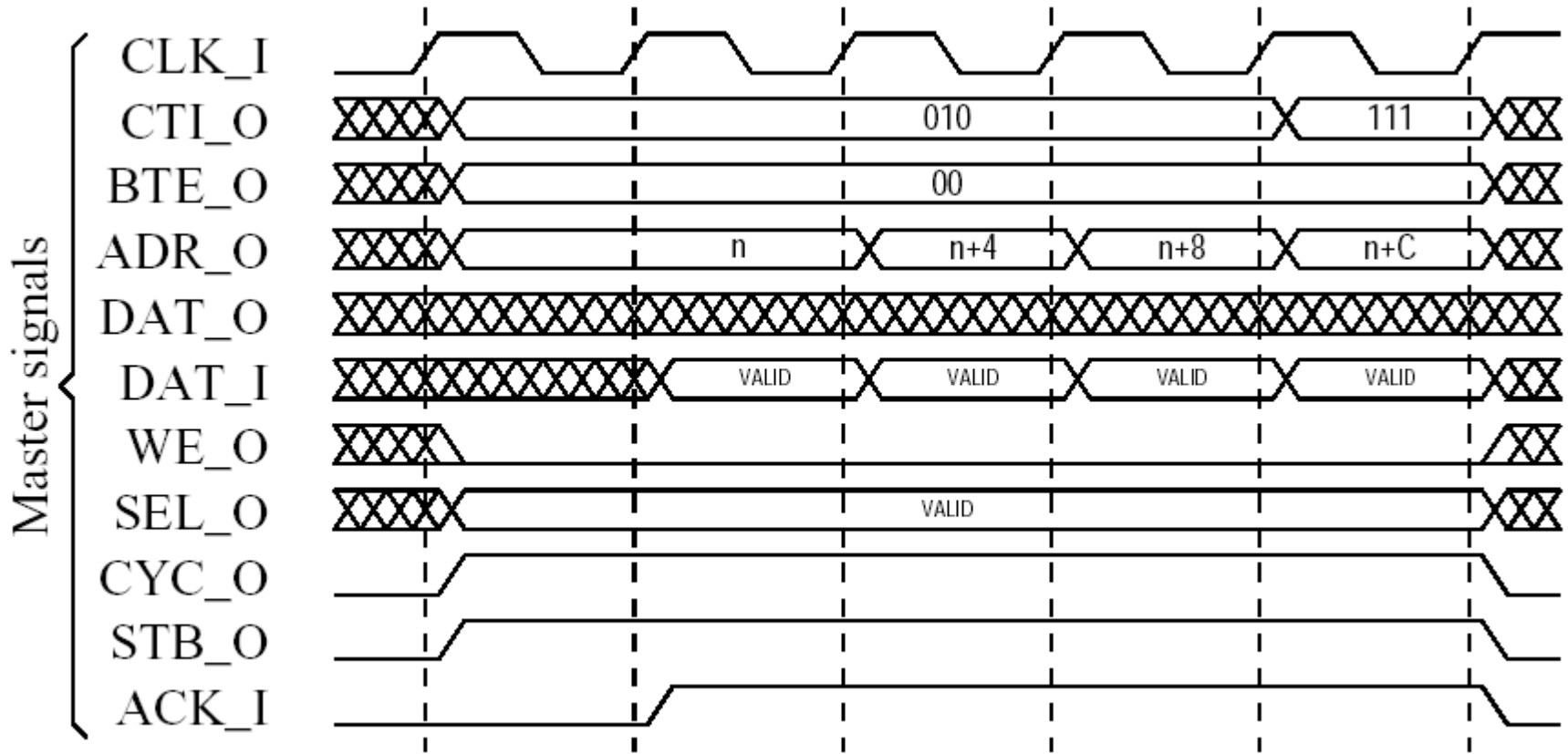


A hint



How long time do these blocks take?

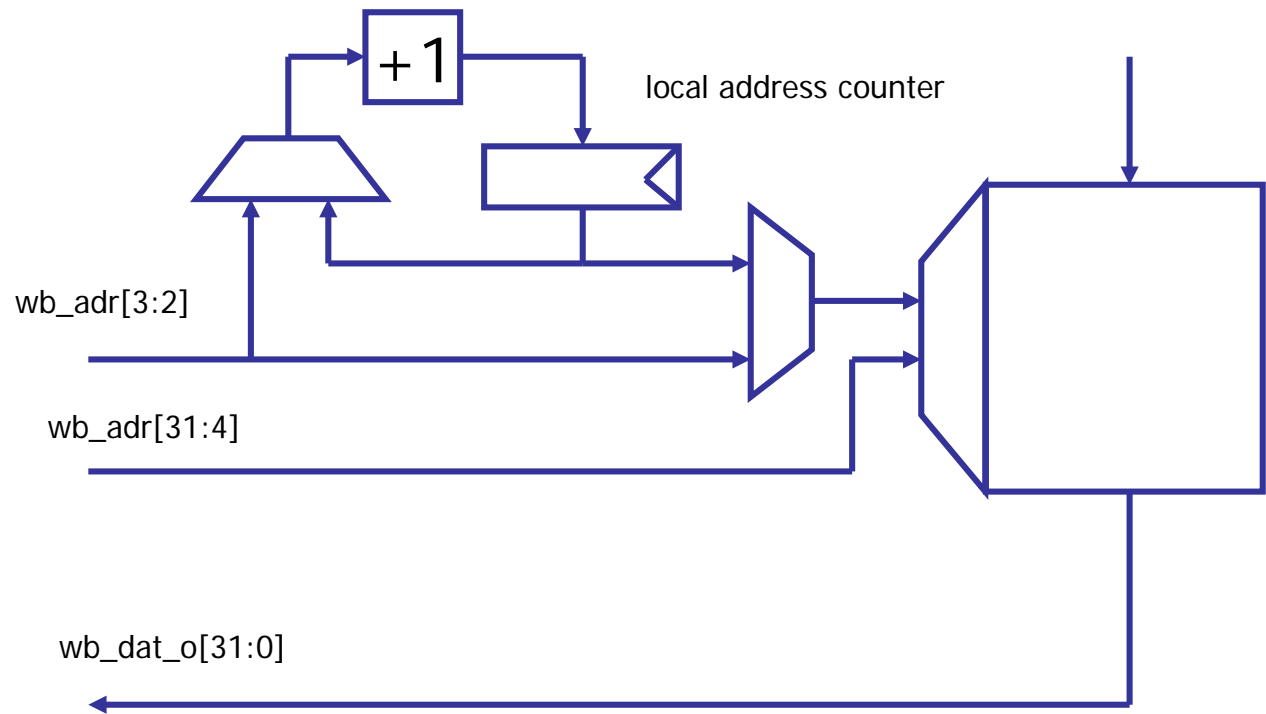
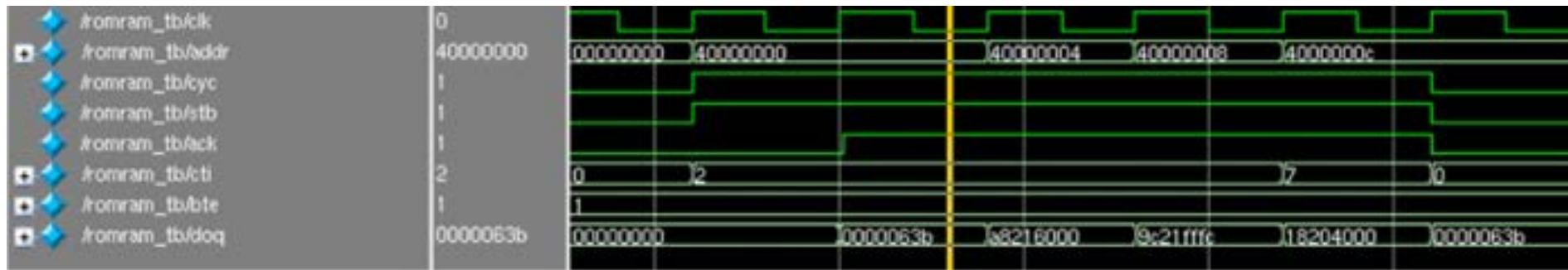
Burst Read



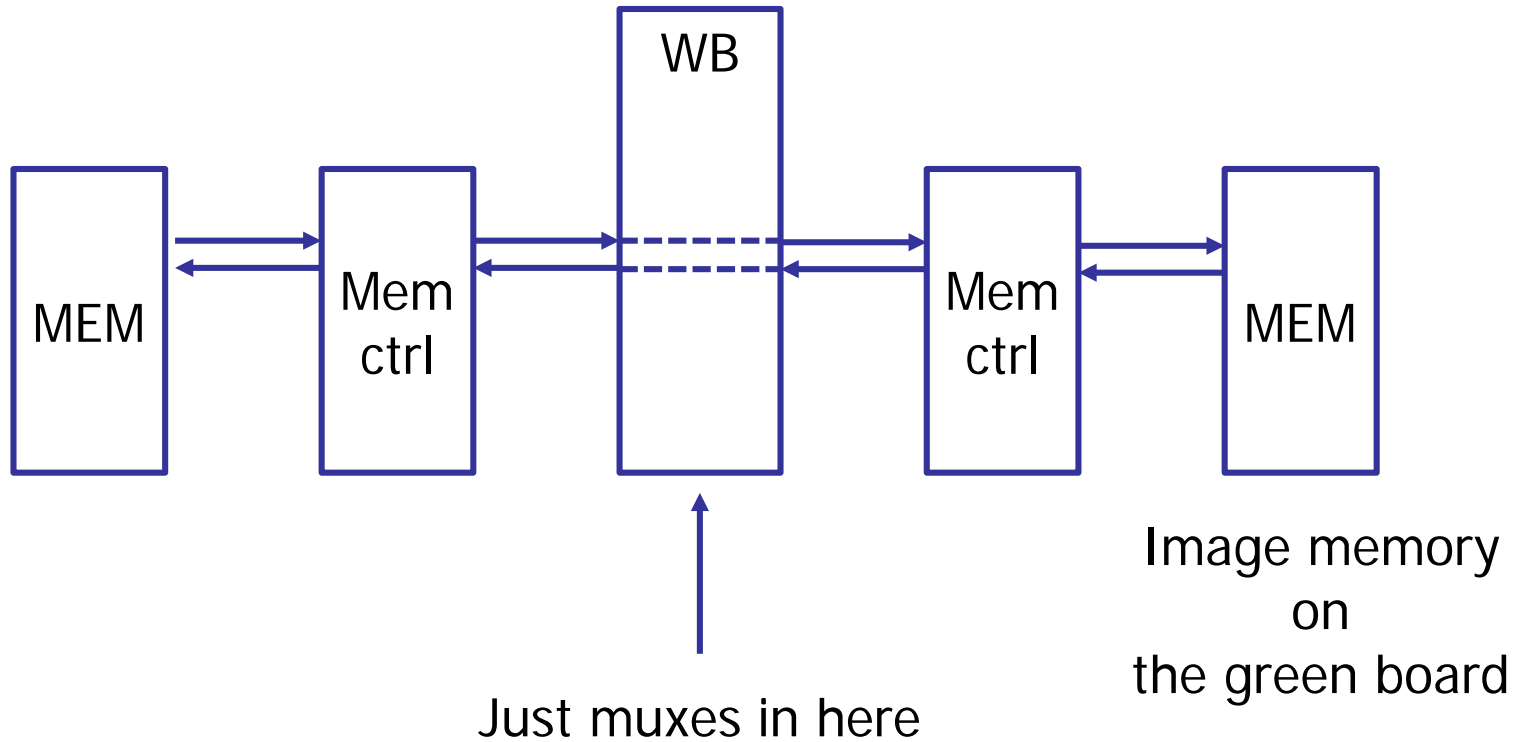
burst - cycle types

Signal group	Value	Description
cti cycle type identifier	000	Classic cycle
	001	Constant address burst cycle
	010	Incrementing burst cycle
	011-110	<i>Reserved</i>
bte burst type extension	111	End of burst
	00	Linear burst
	01	4-beat wrap burst
	10	8-beat wrap burst
	11	16-beat wrap burst

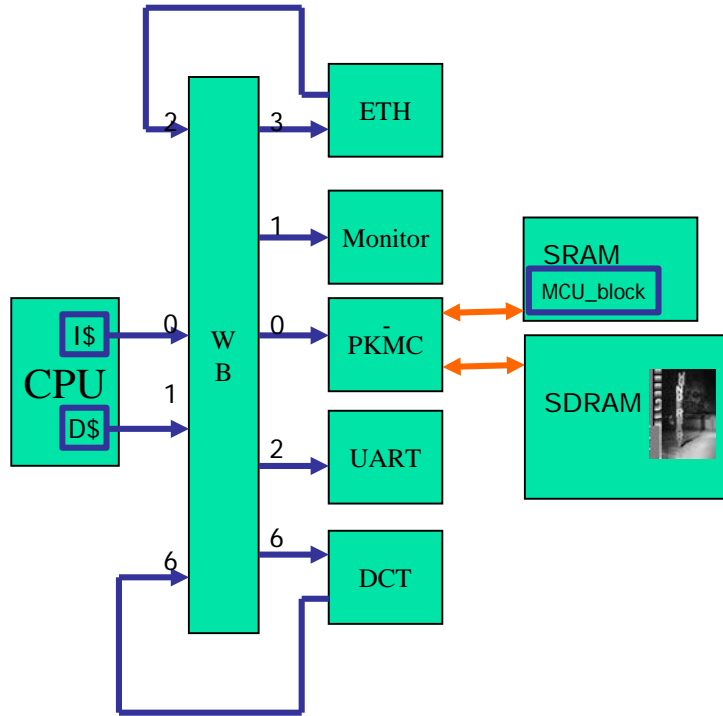
Changes in the slave



Do you really wanna burst?



Why not write? (acc->mem)

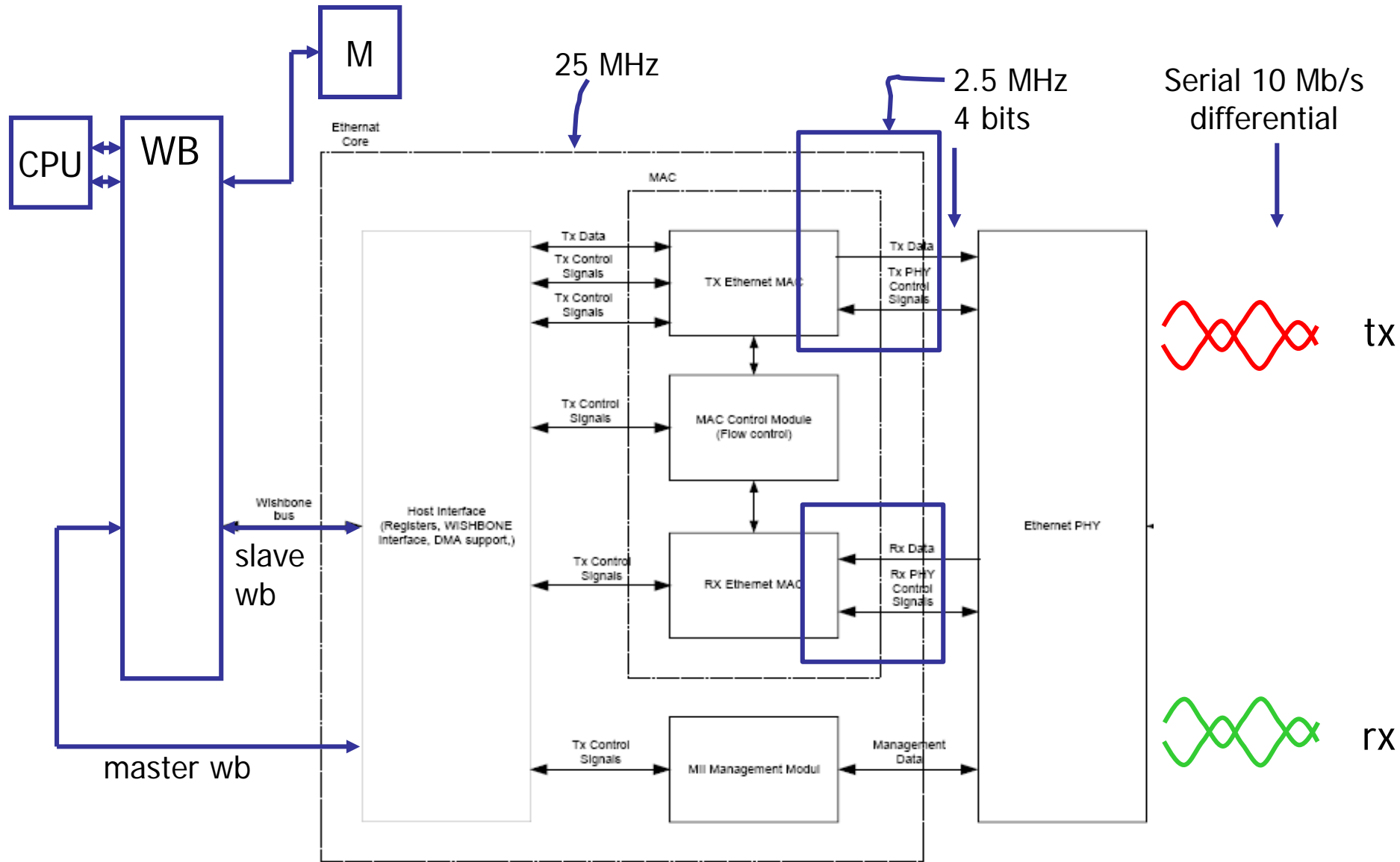


```
/ This the main encoding loop
void encode_image(void)
{
    int i;
    int MCU_count = width*height/DCTSIZE2;
    short MCU_block[DCTSIZE2];

    for(i = 0; i < MCU_count; i++)
    {
        forward_DCT(MCU_block);
        encode_mcu_huff(MCU_block);
    }
}
```

- 1) I/O is on 0x90, 0x91, ..., 0x99
other addr to PKMC
- 2) Noncacheable data mem addr \geq 0x8000_0000,
SDRAM 0x0, SRAM 0x2000_0000 or 0xc000_0000
- 2) **MCU_block** must be in noncacheable area
- 3) Skip **MCU_block**, let **encode_mcu_huff** read from **acc**

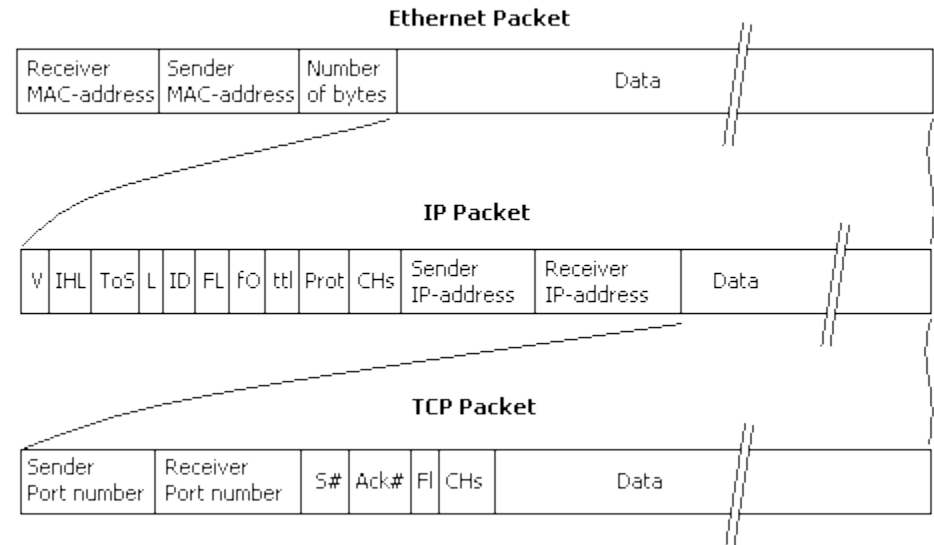
Ethernet controller



FIFOs 2 x 16*32

Ethernet controller

- Transmits and receives Ethernet frames
- 10 Mbit/s and 100 Mbit/s
- Half duplex and full duplex



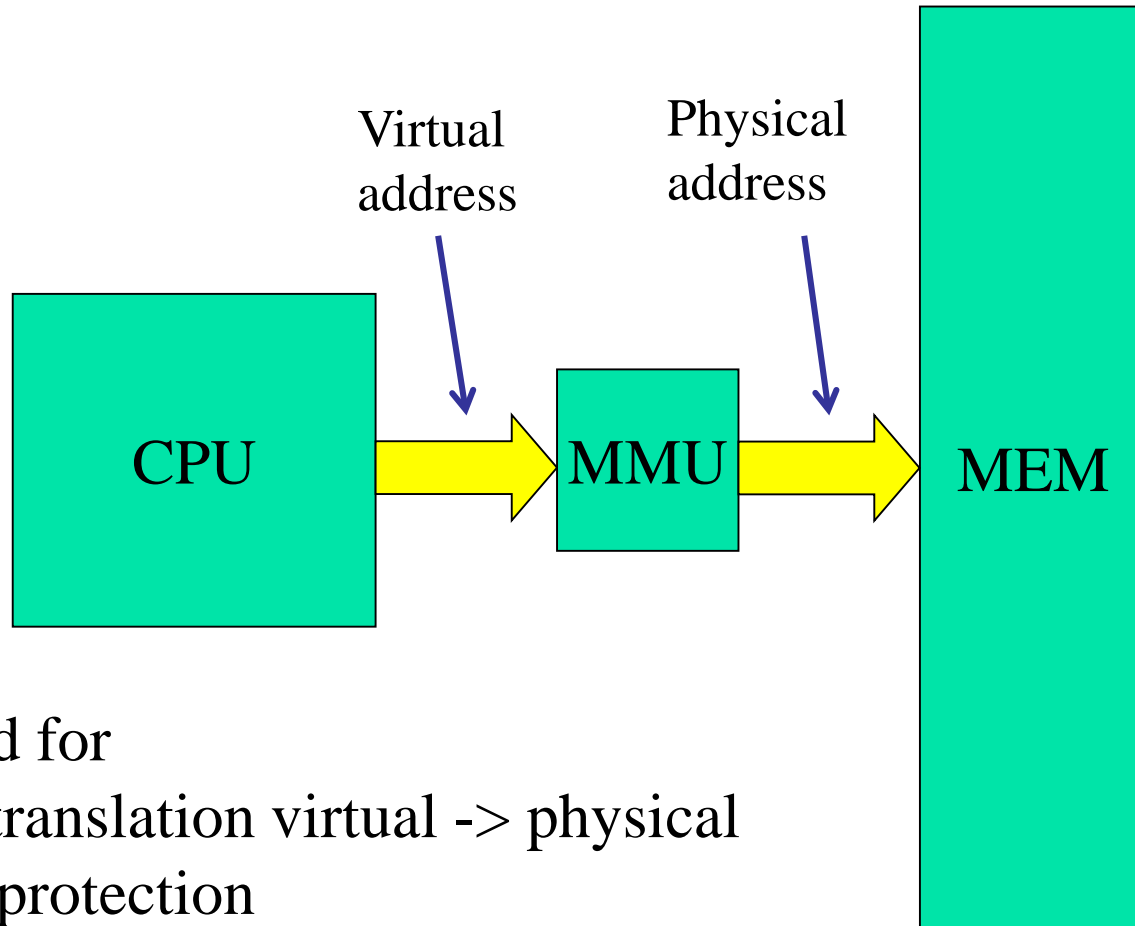
- Wishbone I/F "similar" to our JPEG acc
- Up to 128 buffer descriptors (rx and tx)

length	control & status
address to buffer in mem	

- Tx: automatically reads (DMA) and transmits length bytes
adds length to address
- Proceeds to next BD

or1200

Memory Management

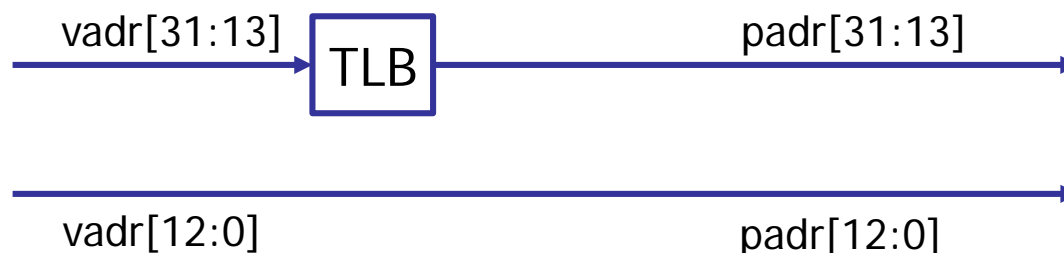


MMU needed for

- 1) Address translation virtual \rightarrow physical
- 2) Memory protection
(OS protected from user processes, ...)
- 3) "each process runs in its own memory"

Memory Management Unit

- Harvard model with split instruction and data MMU
- Instruction/data TLB (translation lookaside buffer)
 - size scalable from 16 to 256 entries
- TLB organized as a direct-mapped cache
- Page size 8KB with per-page attributes
 - LS 13 address lines left untouched
 - MS (32-13) = 19 address lines translated



A sketchy explanation

- * The OS administers a list of page translations for each process.
- * These are kept in memory, page tables
- * The translations are automatically loaded into the TLB when the process executes.

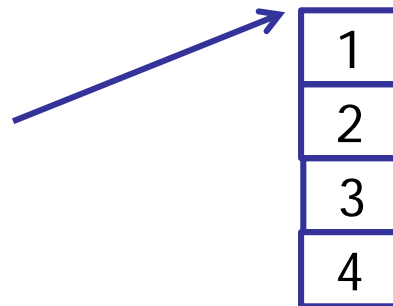
```
#include<stdio.h>
int main()
{
    int *ptr, i;

    ptr = (int *) malloc(4*2048*sizeof(int));

    for (i=0; i<8192; i++)
        *ptr++ = i;
    ...

    free(ptr_one);
    return 0;
}
```

Virtual address space

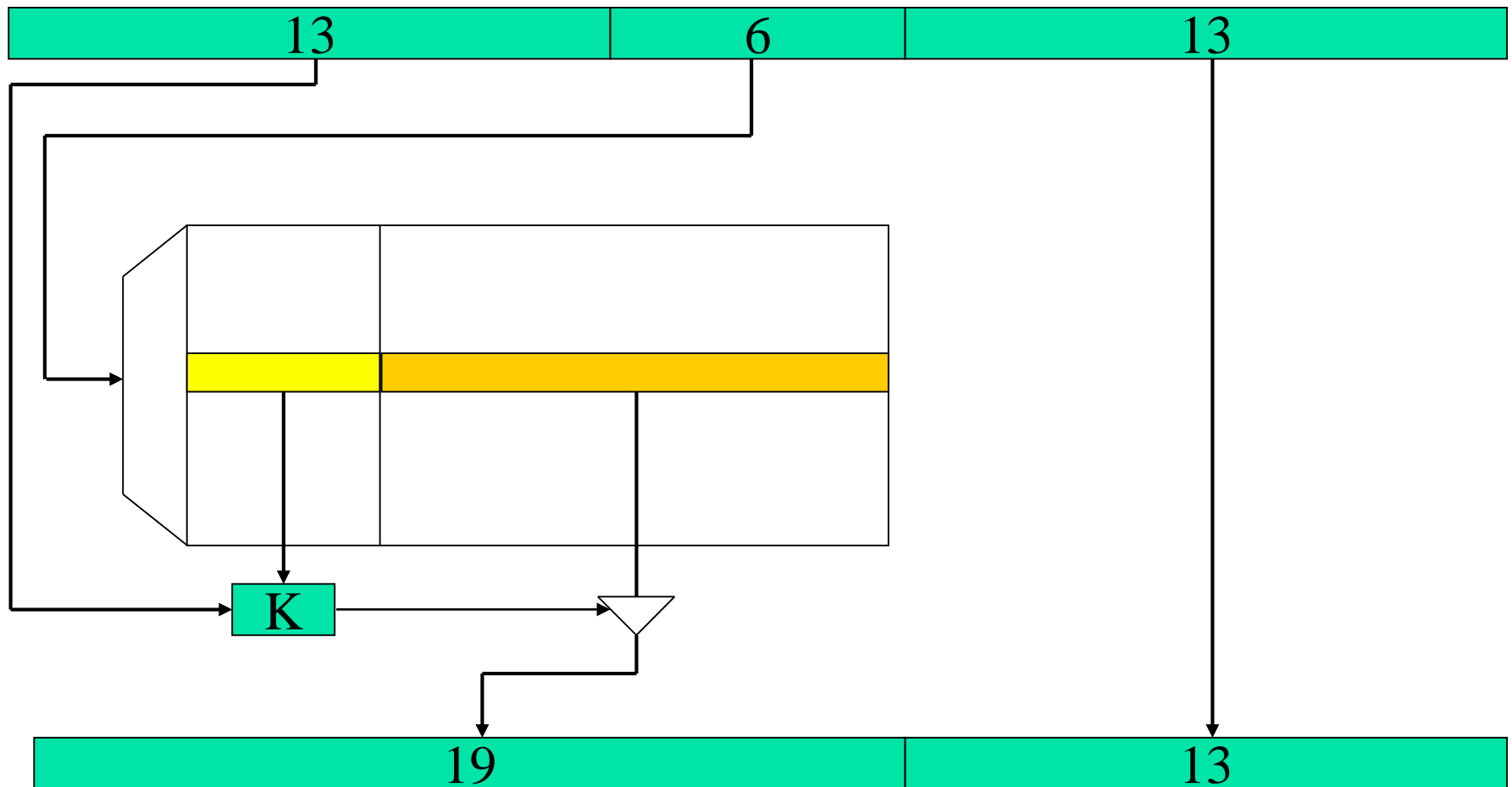


Physical address space

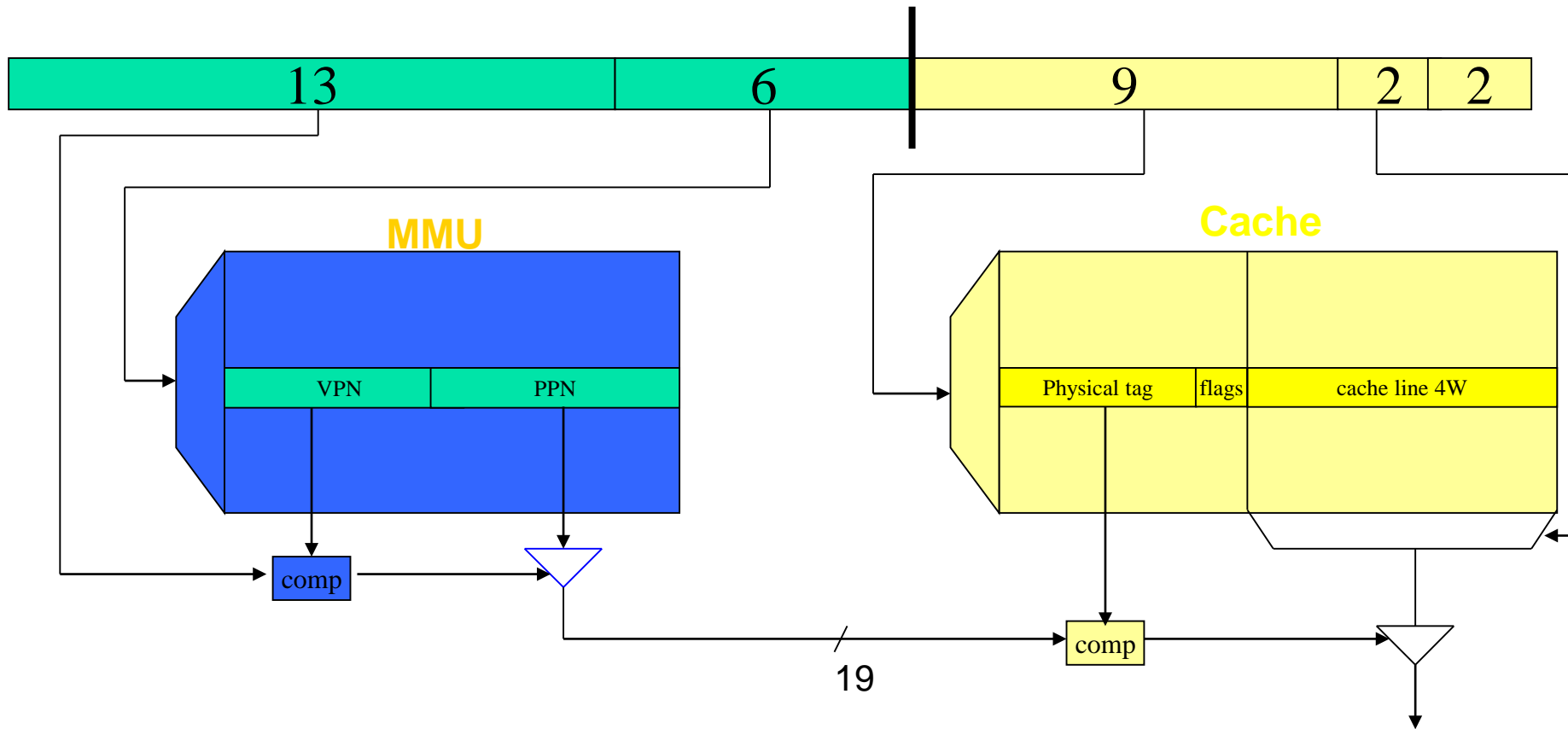


I/D-TLB = Translation Lookaside buffer

Implemented as a direct mapped cache, 64 entries

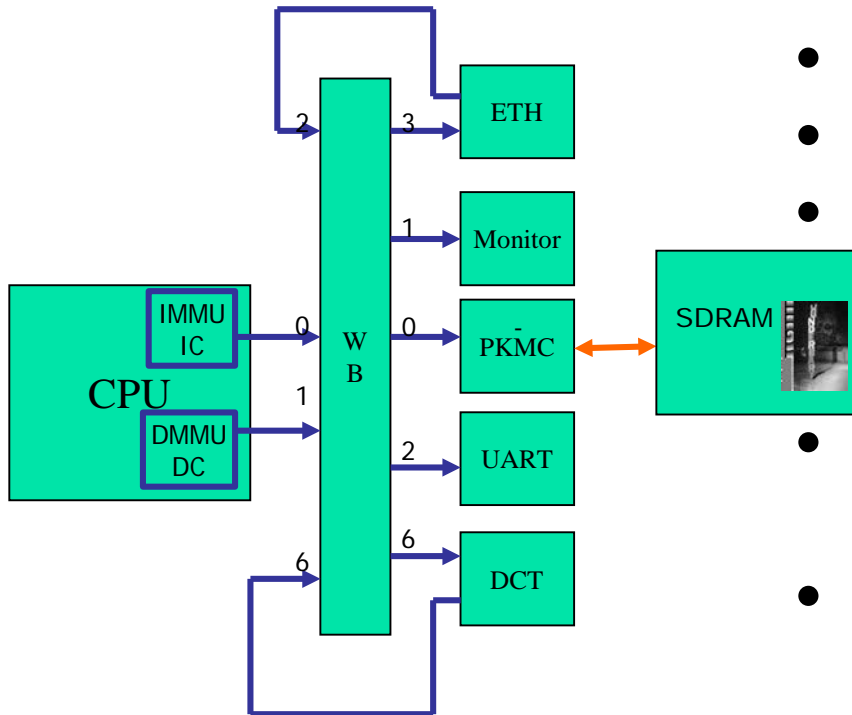


Does the MMU need an extra pipeline stage?

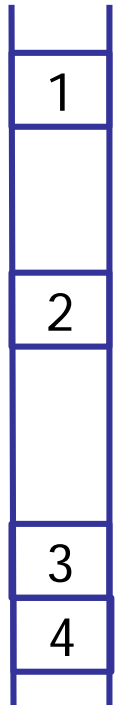
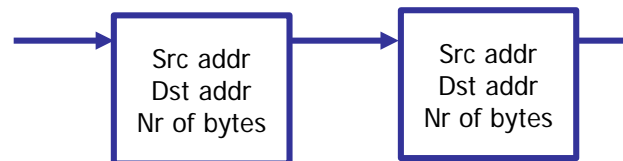


NO, the cache is physical and works in parallel with the MMU!

What about DMA and MMU?



- μ Clinux does not use the MMUs
- Real Linux must use MMUs
- System call to find the P addr for a V addr
- DMA ctrl must handle scatter/gather
- DMA ctrl typically executes a linked list of commands



other soft CPUs

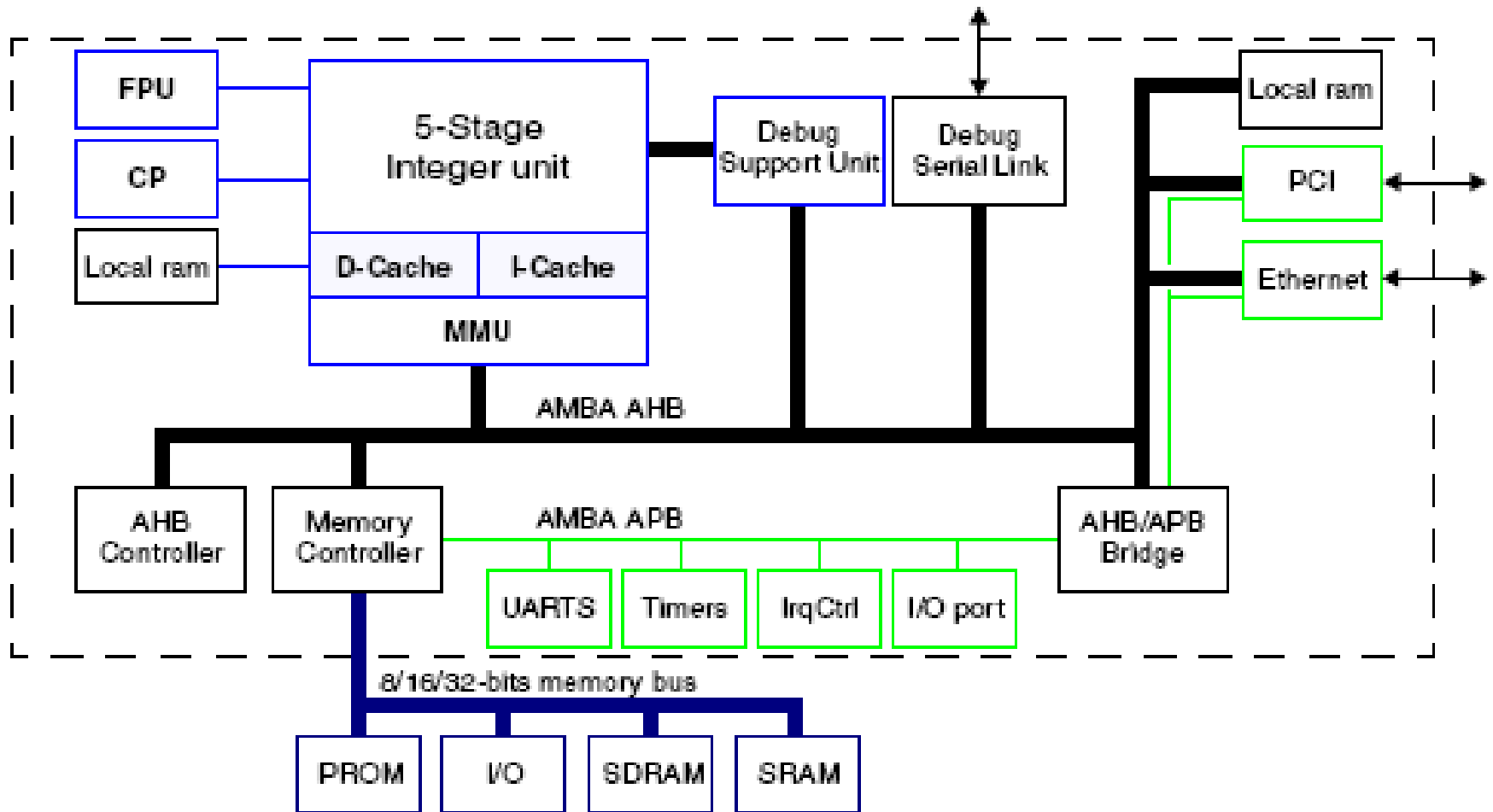
	Open RISC	Leon	Nios	Micro-Blaze
who	opencores	gaisler	altera	Xilinx
what	verilog	VHDL	netlist	netlist
CPU stages	RISC 5	RISC 5	RISC 6/5/1	RISC 3
cache	Direct IC/DC	IC/DC	IC/DC	IC/DC
MMU	Split IMMU DMMU			
bus	Wishbone simple/Xbar	AMBA (AHP/APB)		LMB/OPB/ FSL

leon - open source processor

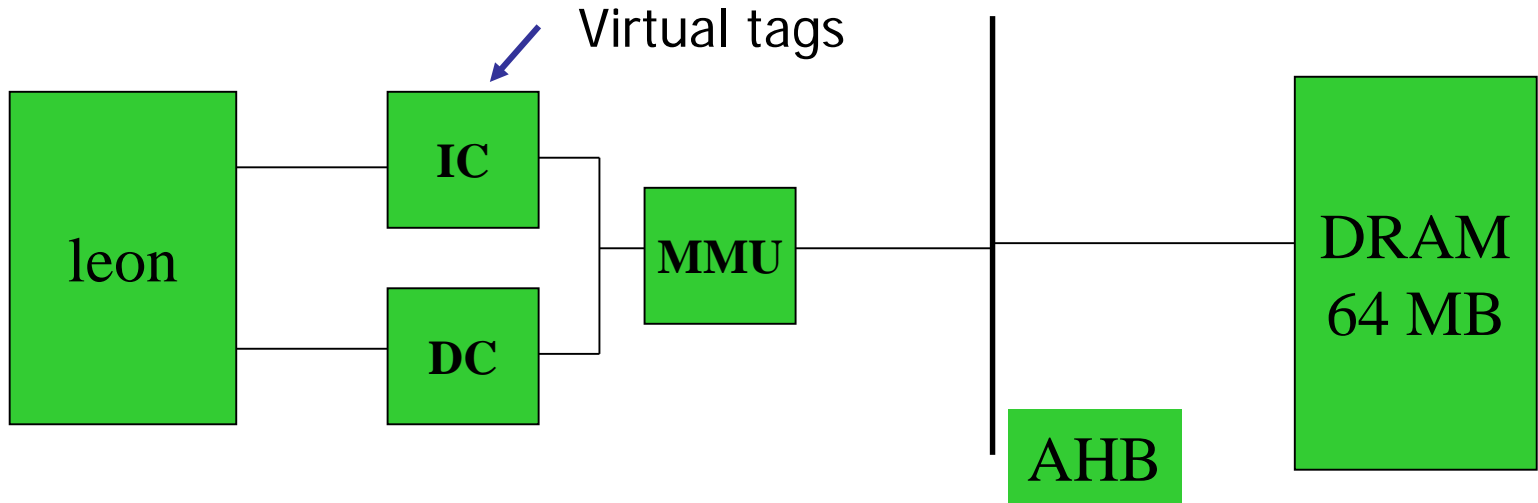
www.gaisler.com

- The full source code is available under the **GNU LGPL** license
- LEON2 is a synthesisable VHDL model of a 32-bit processor compliant with the SPARC V8 architecture
- **SPARC V8 compliant** integer unit with 5-stage pipeline
- Hardware multiply, divide and MAC units
- Separate instruction and data cache (Harvard architecture)
- Set-associative caches: 1 - 4 sets, 1 - 64 kbytes/set.
Random, LRR or LRU replacement
- Data cache snooping
- **AMBA-2.0** AHB and APB on-chip buse
- 8/16/32-bits memory controller for external PROM and SRAM
- 32-bits PC133 SDRAM controller
- On-chip peripherals such as uarts, timers, interrupt controller and 16-bit I/O port

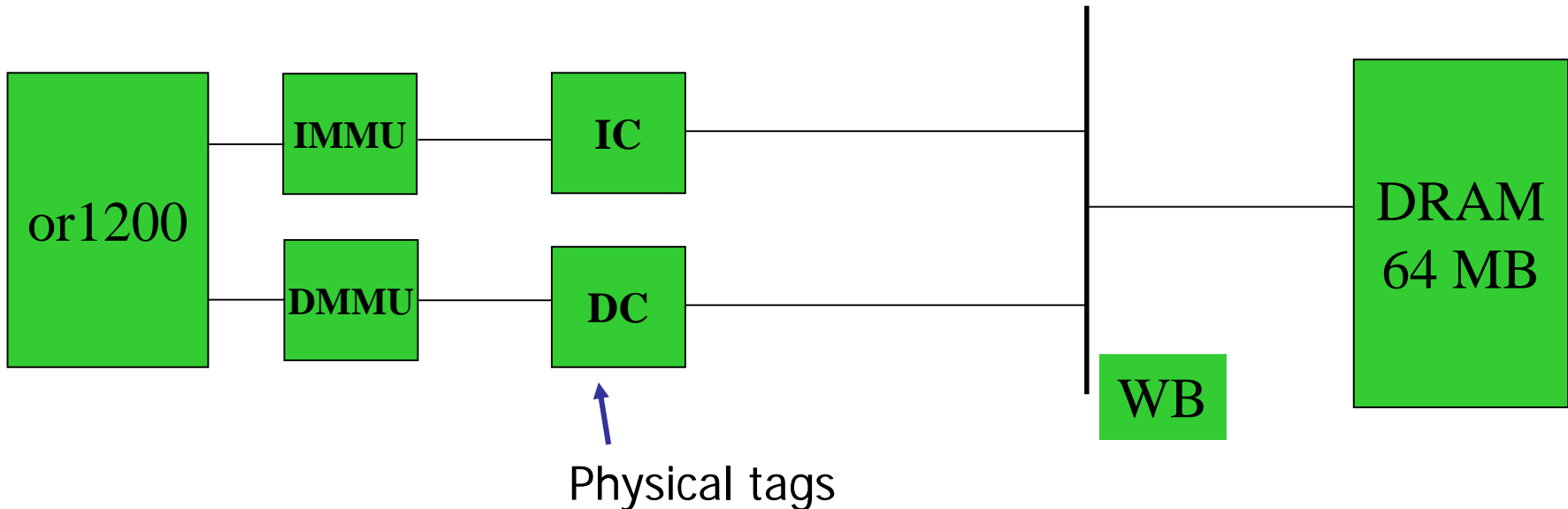
leon



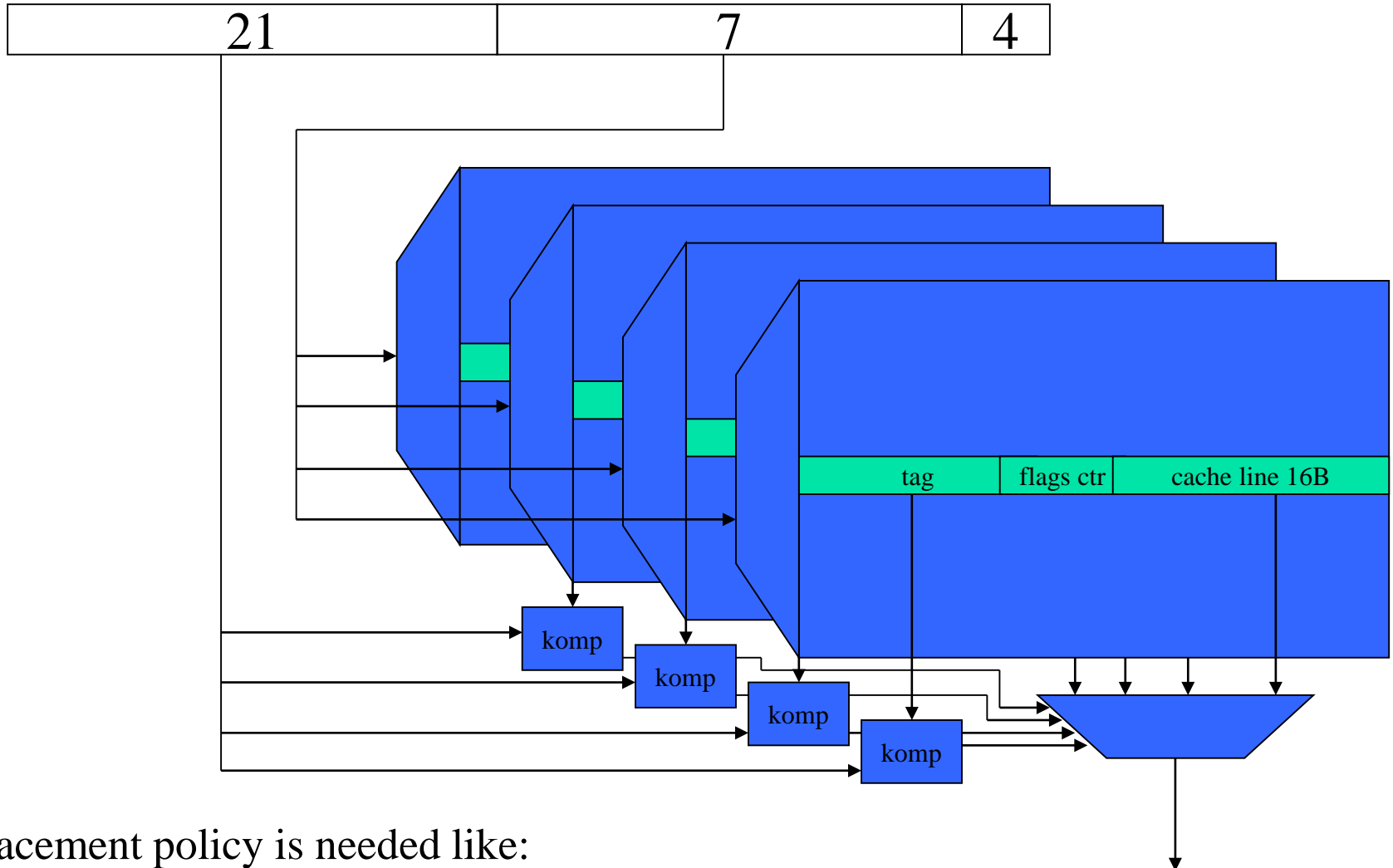
leon has virtual caches!



- + Address translation only at cache miss!
- Cache flush needed at task switch



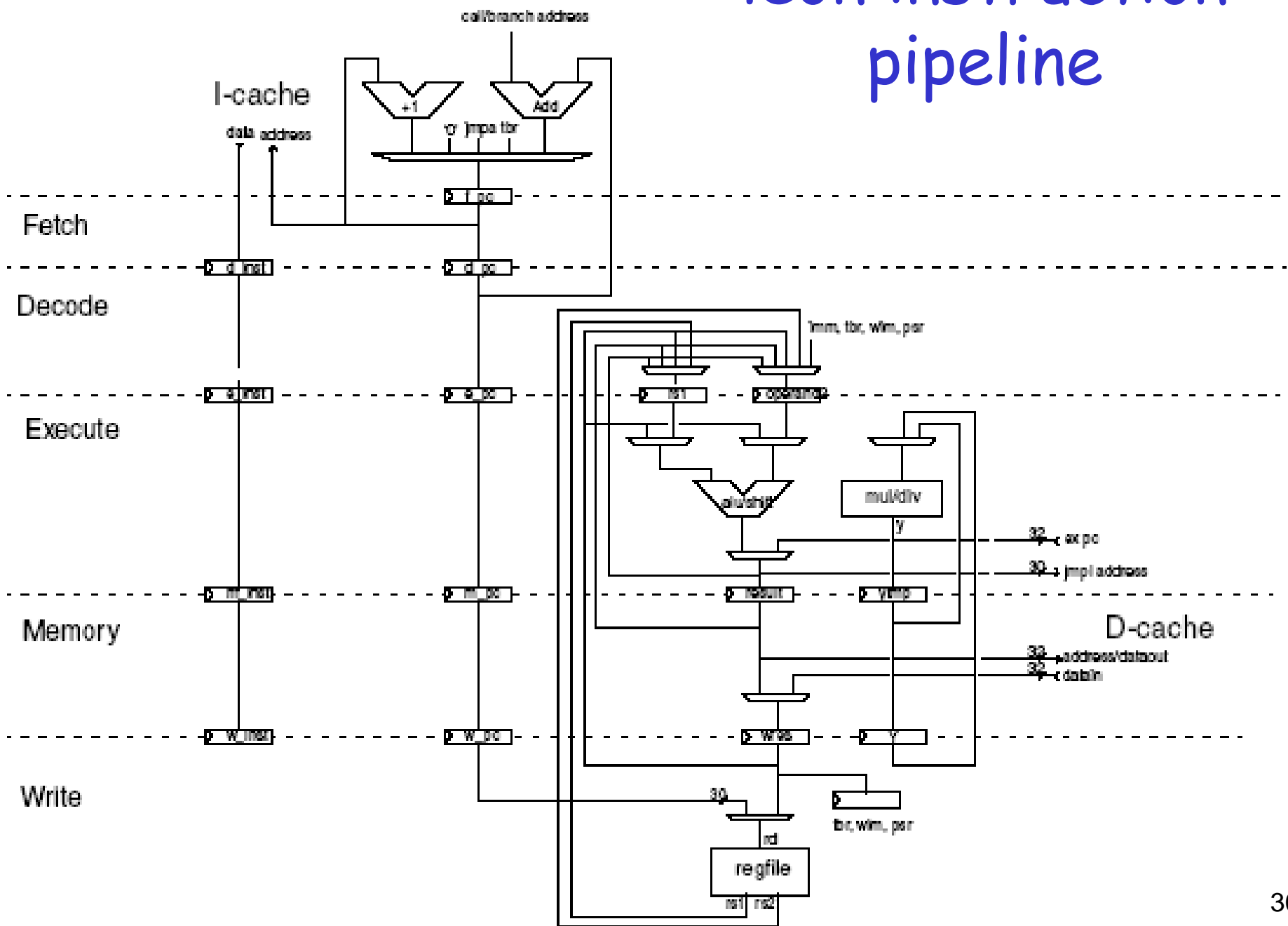
A 4-way 8kb instruction cache



A replacement policy is needed like:

- * LRU = least recently used
- * LRR = least recently replaced

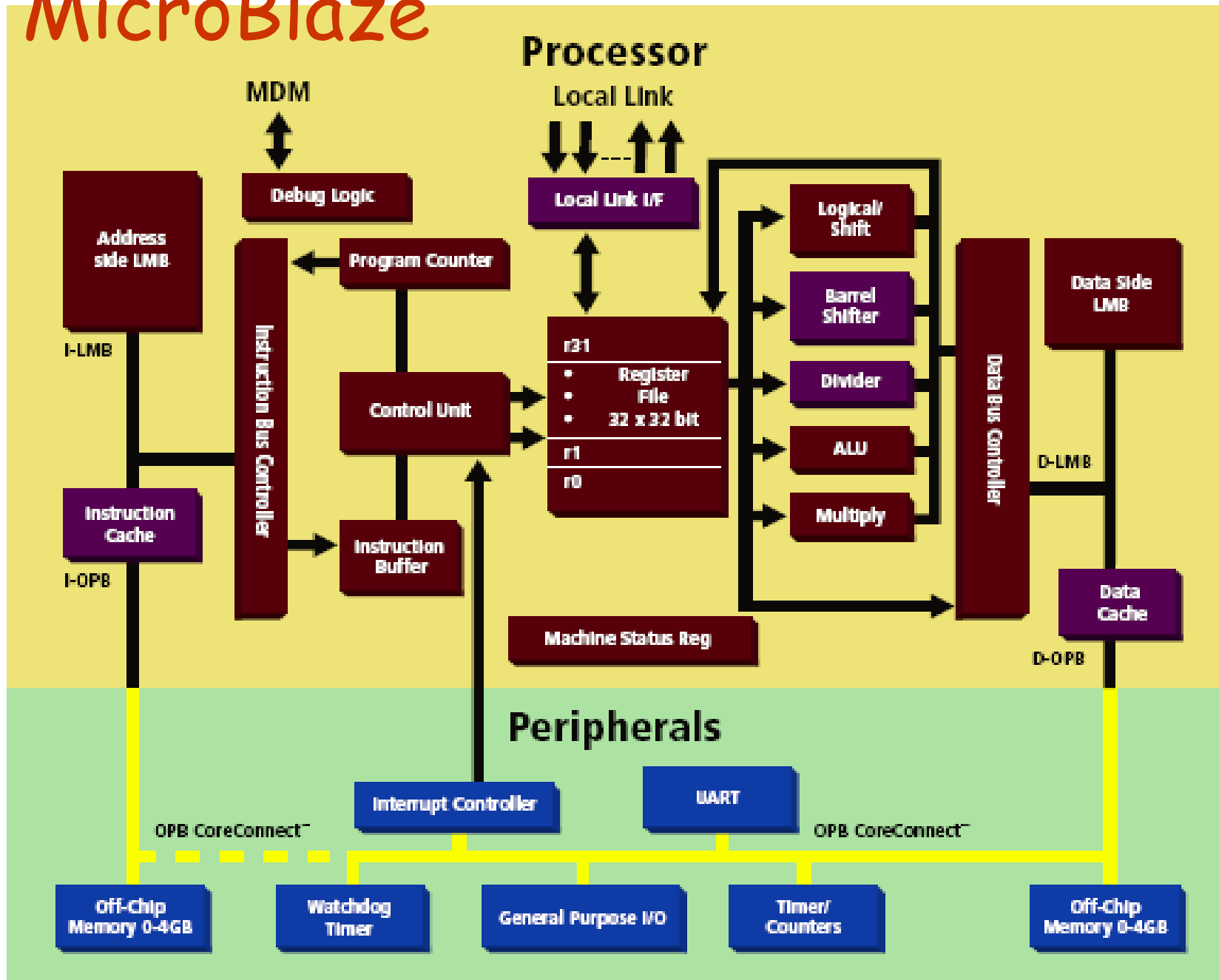
leon instruction pipeline



MicroBlaze Processor

- Thirty-two 32-bit general purpose registers
- 32-bit instruction word with three operands and two addressing modes
- Separate 32-bit instruction and data buses that conform to IBM's OPB (On-chip Peripheral Bus) specification
- Separate 32-bit instruction and data buses with direct connection to on-chip block
- RAM through a LMB (Local Memory Bus)
- 32-bit address bus
- Single issue pipeline
- Instruction and data cache
- Hardware debug logic
- **FSL (Fast Simplex Link) support**
- Hardware multiplier (in Virtex-II and subsequent device)

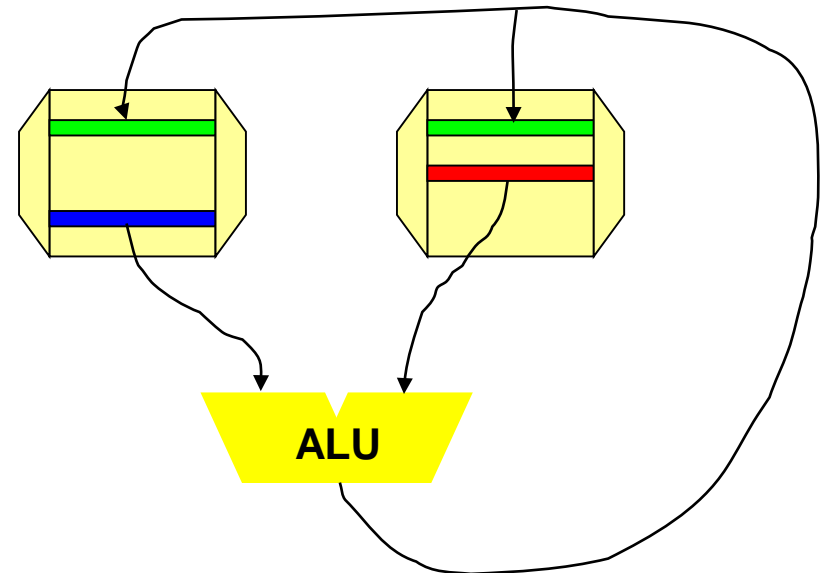
MicroBlaze



MicroBlaze Pipeline

	cycle 1	cycle 2	cycle 3	cycle 4	cycle 5
instruction 1	Fetch	Decode	Execute		
instruction 2		Fetch	Decode	Execute	
instruction 3			Fetch	Decode	Execute

- Execute stage will dominate the pipeline
- + No data hazards
- Delay slot still needed



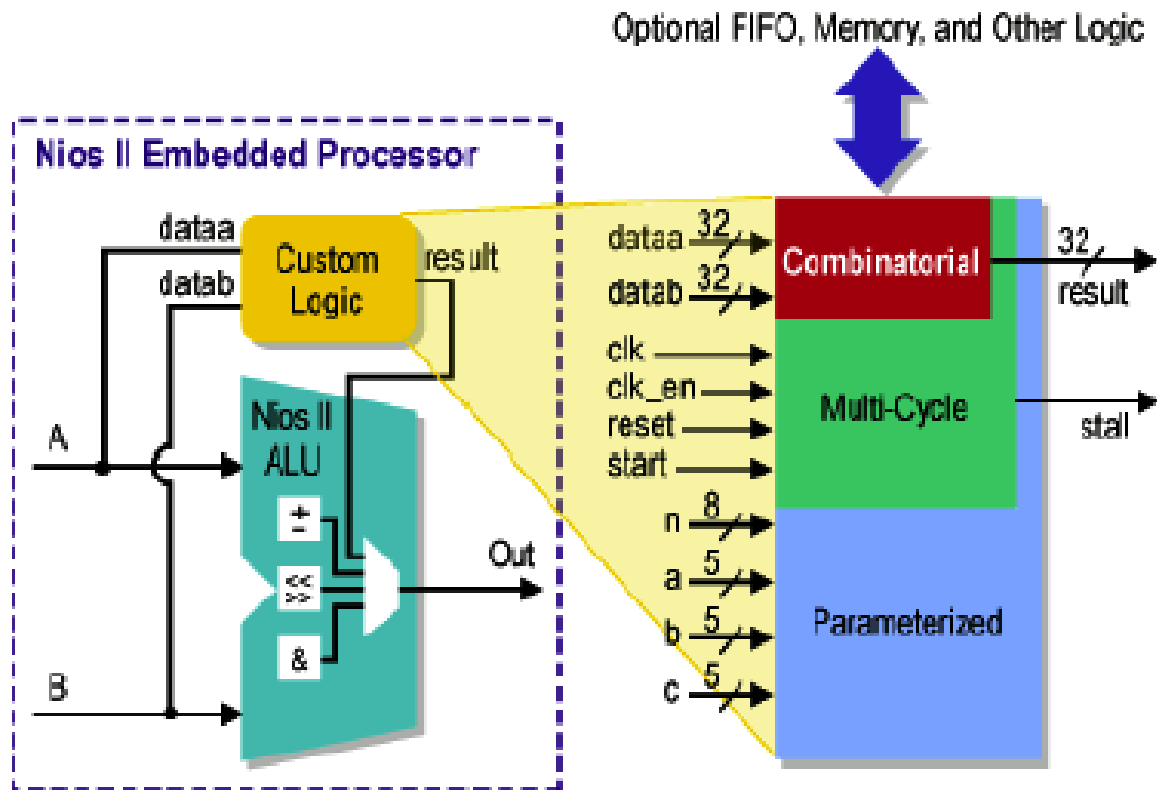
Nios (Altera)

Table 1: Key features of the Nios II family members

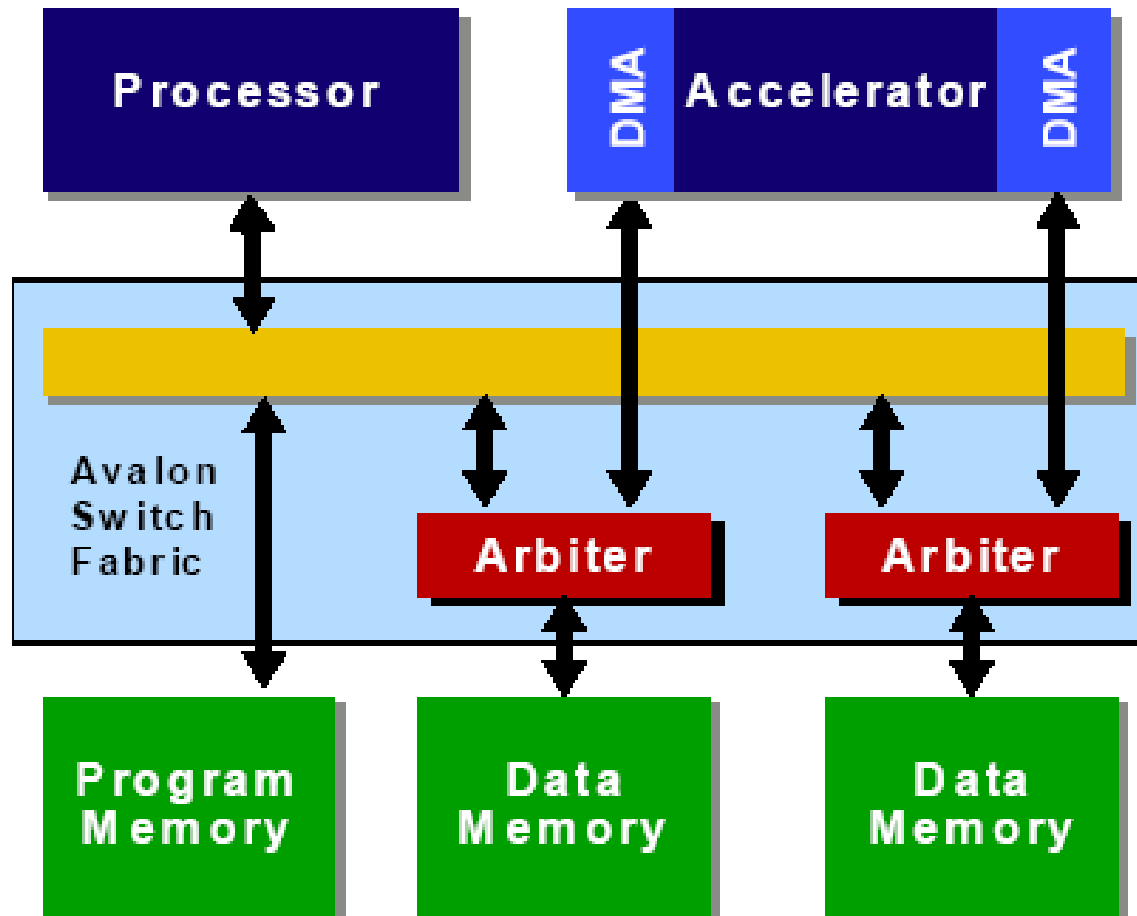
	Nios II /f Fast	Nios II /s Standard	Nios II /e Economy
Pipeline	6 Stage	5 Stage	None
Multiplier *	1 Cycle	3 Cycle	None
Branch Prediction	Dynamic	Static	None
Instruction Cache	Configurable	Configurable	None
Data Cache	Configurable	None	None

*Uses digital signal processing (DSP) blocks in Stratix and Stratix II FPGAs

Custom Instructions

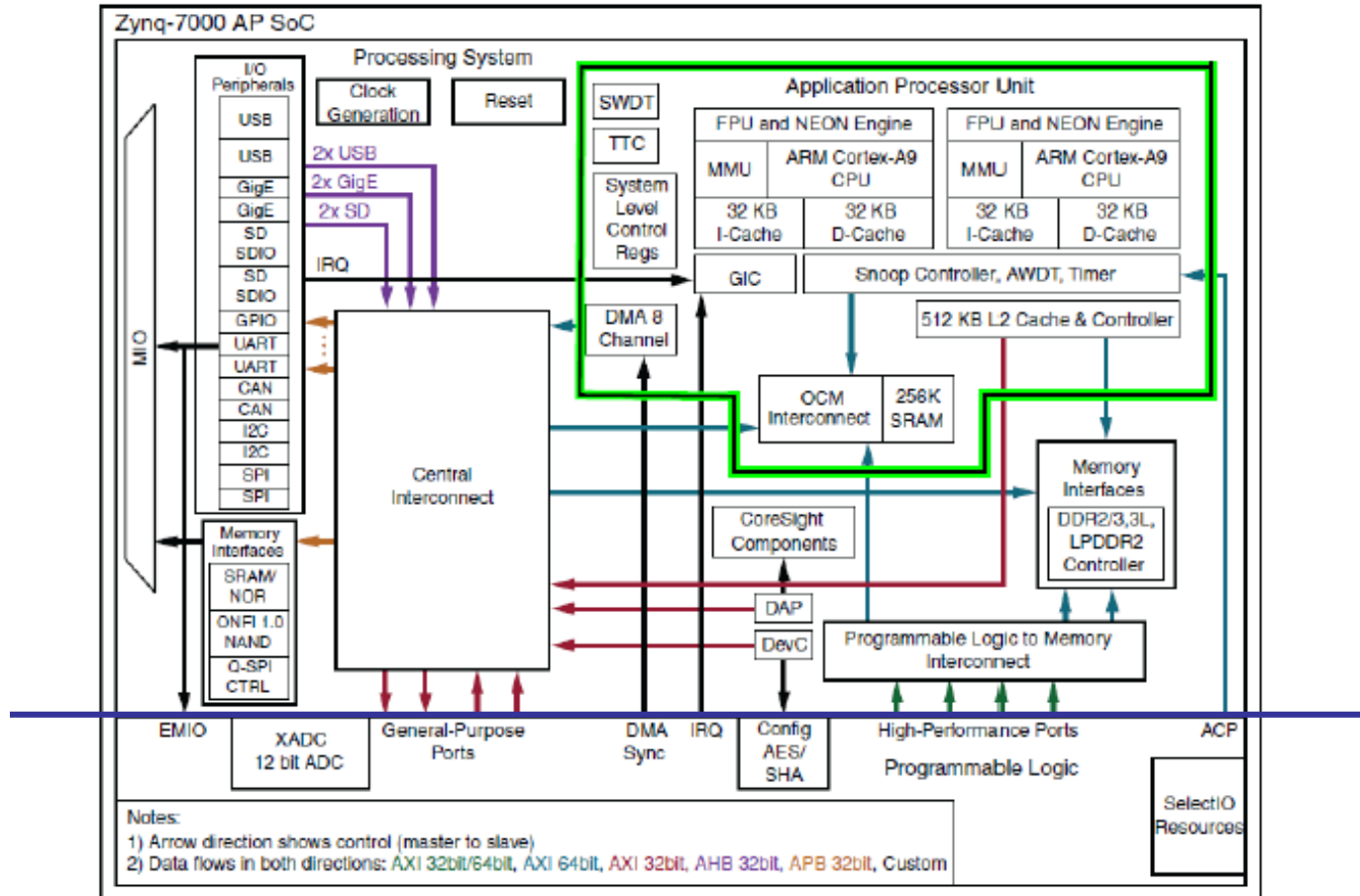


HW Accelerator



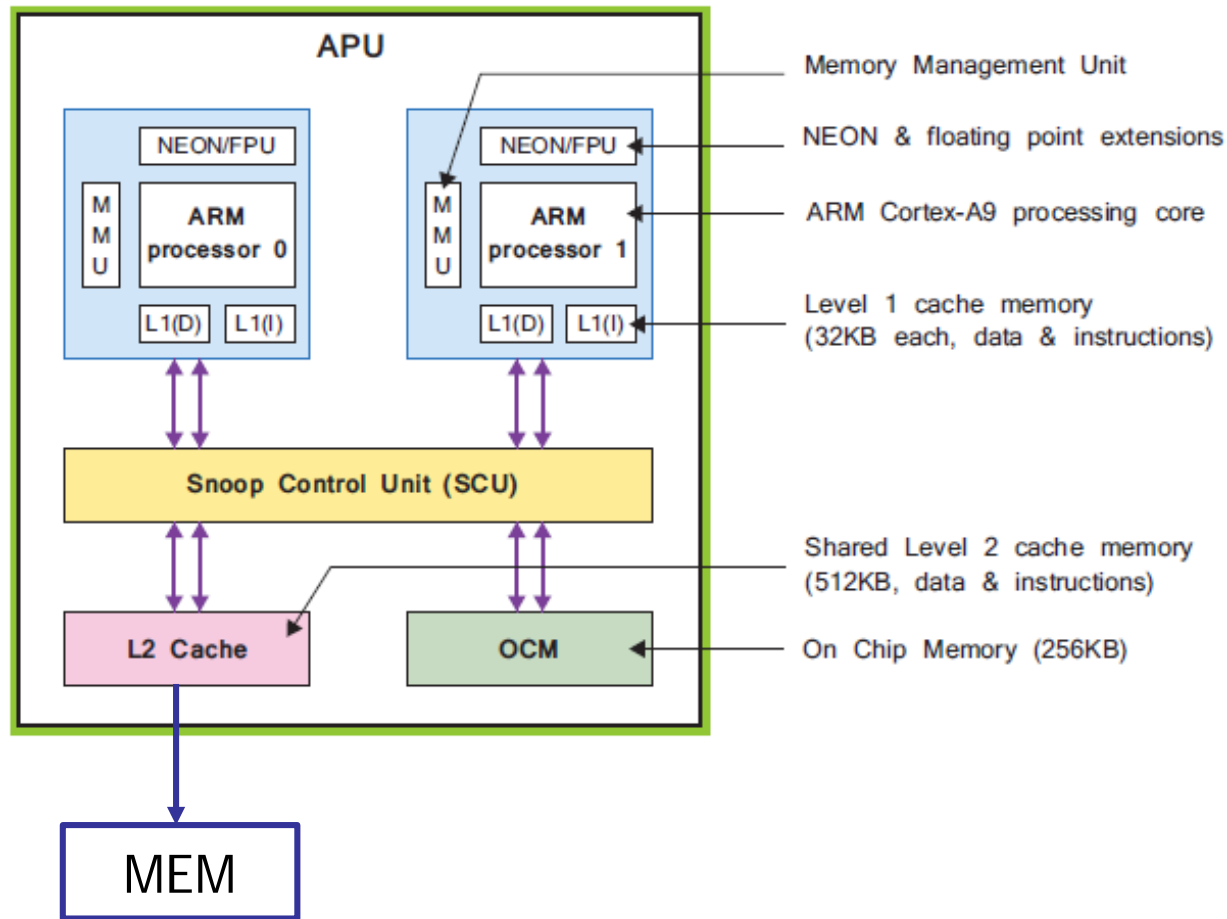
Zynq - a programmable SOC

PS = processing system



PS = programmable logic

PS = processing system



SCU keeps cachelines in the two L1(D) synchronised (if they refer to the same mem position)

Snoop Control Unit

For each cacheline we keep track of:

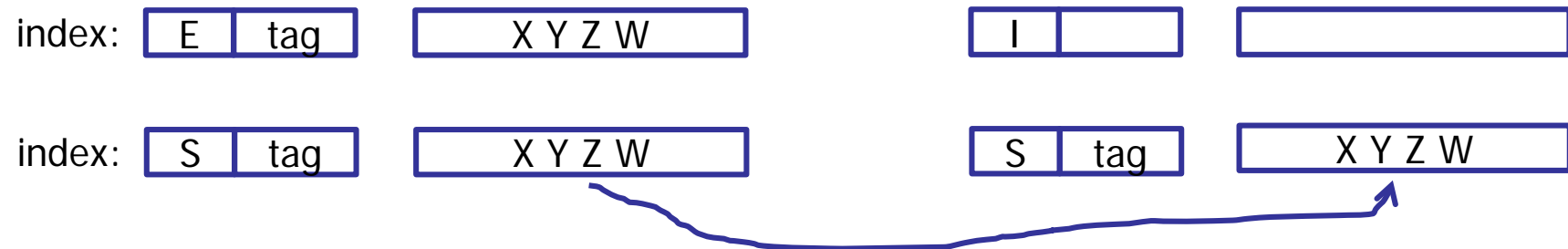
Modified : Unique & Dirty (only in this cache, has been changed)

Owned : Shared & Dirty

Exclusive : Unique & Clean

Shared : Shared & Clean

Invalid : Nothing here yet



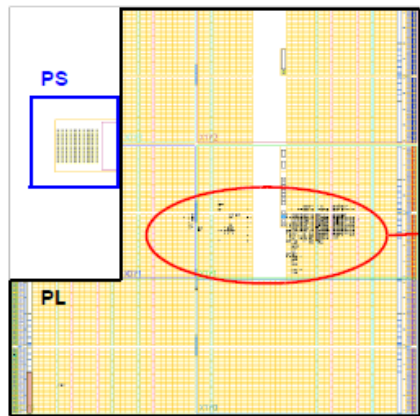
The SCU listens to the bus, has a copy of the tag RAMs, ...

PL = programmable logic



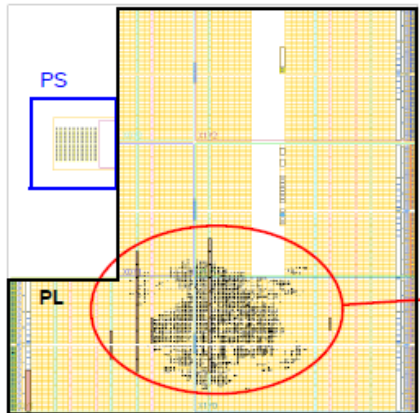
	Z-7010	Z-7015	Z-7020	Z-7030	Z-7045	Z-7100
Processor	Dual core ARM Cortex-A9 with NEON and FPU extensions					
Max. processor clock frequency	866MHz			1GHz		
Programmable Logic	Artix-7			Kintex-7		
No. of FlipFlops	35,200	96,400	106,400	157,200	437,200	554,800
No. of 6-input LUTs	17,600	46,200	53,200	78,600	218,600	277,400
No. of 36Kb Block RAMs	60	95	140	265	545	755
No. of DSP48 slices (18x25 bit)	80	160	220	400	900	2020
No. of SelectIO Input/Output Blocks ^a	HR: 100 HP: 0	HR: 150 HP: 0	HR: 200 HP: 0	HR: 100 HP: 150	HR: 212 HP: 150	HR: 250 HP: 150
No. of PCI Express Blocks	-	4	-	4	8	8
No. of serial transceivers	-	4	-	4	8 or 16 ^b	16
Serial transceivers maximum rate	-	6.25Gbps	-	6.6Gbps / 12.5Gbps ^c	6.6Gbps / 12.5Gbps ^b	10.3Gbps

Performance Soft vs Hard



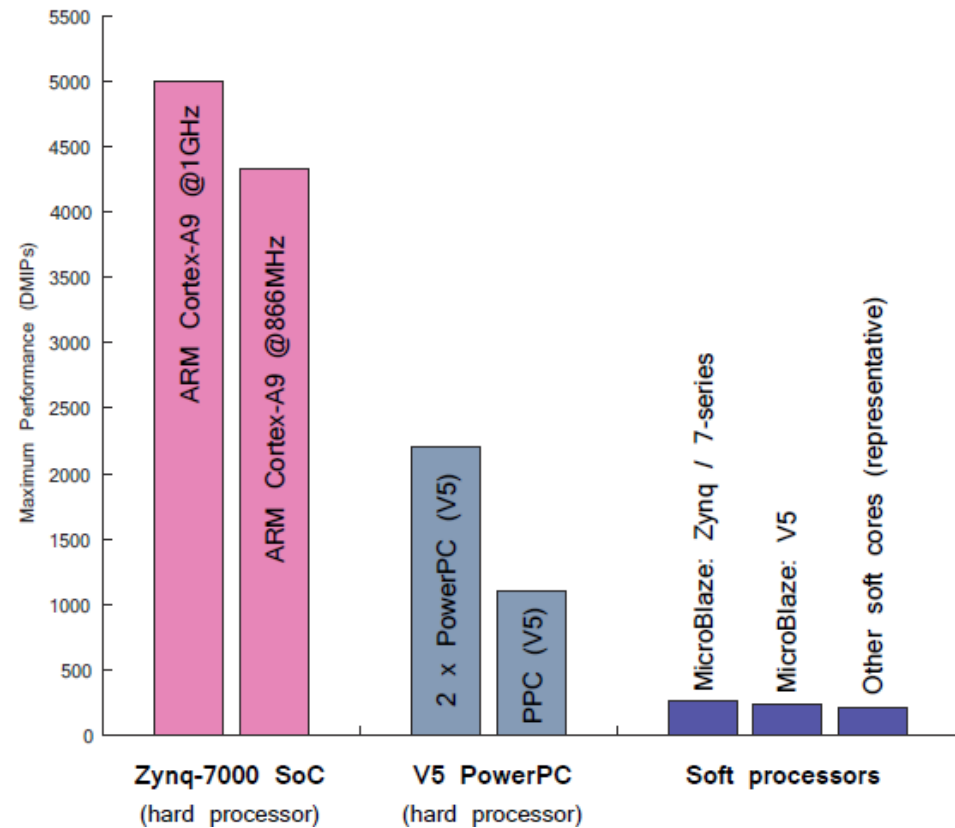
Minimum Area MicroBlaze

footprint of MicroBlaze processor
in logic fabric



Maximum Performance MicroBlaze

footprint of MicroBlaze processor
in logic fabric



ACP = accelerator coherency port

Acc connected like a CPU to mem
On a read hit data is read from
one of the caches

