

TSEA44: Computer hardware – a system on a chip

Lecture 8: Memories, lab4

Practical info

- Lab closed after 22/12
 - Opens again after new year (probably after 2/1-17)
 - Ask me or Erik to let you in (if we are at the work)
 - Remote login still works
- Office corridors locked during christmas/new year
 - Hard to get access to people (if they are not on vacation)
- Lab used for other courses in the spring
 - No access guaranteed after the course end
 - Will try to set up some limited access location
 - Probably a lab location with limited access only on non-scheduled hours

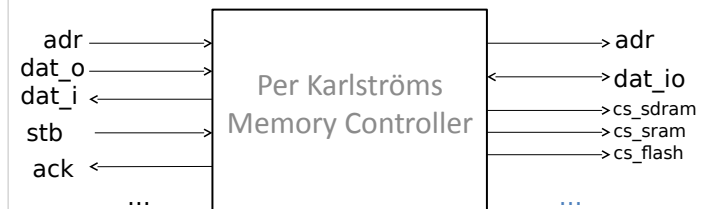
Today

- Memories/memory controller
- Lab4, new instruction

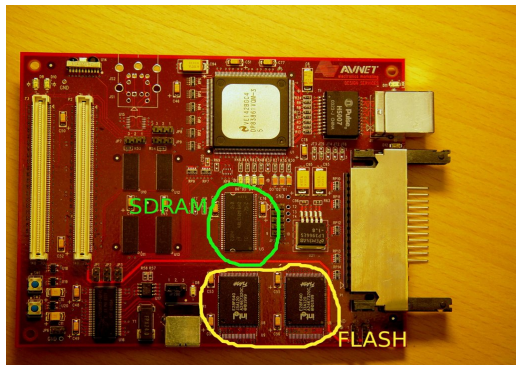
PKMC

Wishbone bus

Memory bus



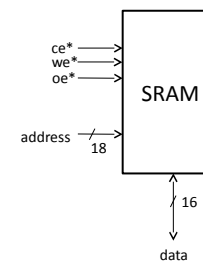
Memory on board



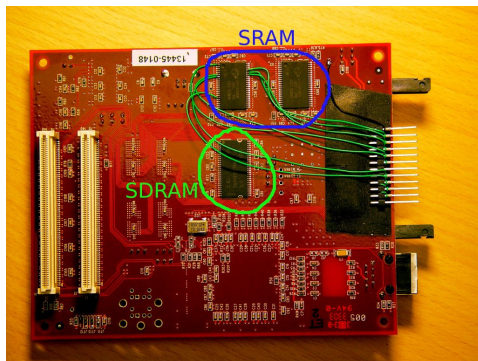
SRAM; Static RAM

- Asynchronous device
- Memory element: latch
- $2 \times (256k \times 16) = 1 \text{ MB}$

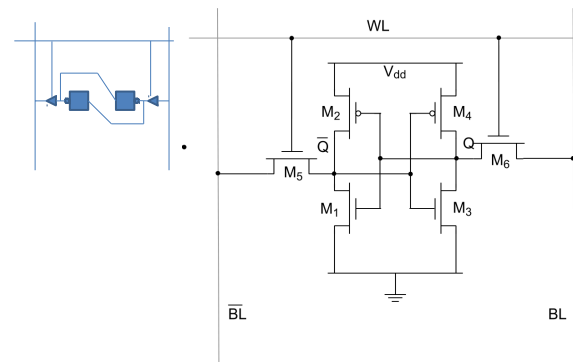
* = active low



Memory on board, cont.



SRAM - Cell



TSEA44: Computer hardware – a system on a chip 2017-12-04 9

SRAM - Read

- WL=0, Precharge bitlines to $V_{dd}/2$
- WL=1 connects inverters to bitlines
- Bitlines are driven to low and high

The diagram shows a 1T1R1C SRAM cell. The wordline (WL) is at 1, connecting the two inverters to the bitlines. The bitlines are precharged to $V_{dd}/2$. The data is read from the bitlines.

h.u. LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 11

SRAM - Write

- WL=0, Set logic values on bitlines
- WL=1 connects inverters to bitlines
- Bitline values override internal value

The diagram shows a 1T1R1C SRAM cell. The wordline (WL) is at 1, connecting the two inverters to the bitlines. The bitlines are driven to 0 and 1, overriding the internal value.

h.u. LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 10

SRAM - Read

- No clocking!

The diagram shows the timing for a read operation. It includes signals for ADDRESS, CE, OE, BHE, BLE, and DATA OUT. Key timing parameters include t_{RC} , t_{ACE} , t_{DOE} , t_{LZOE} , t_{H2OE} , t_{H2CE} , t_{H2BE} , t_{LZCE} , t_{LZBE} , t_{PU} , t_{PD} , t_{SD} , and t_{SB} .

h.u. LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 12

SRAM - Write

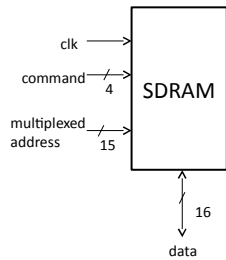
- Timing important (avoid writing to wrong address)

The diagram shows the timing for a write operation. It includes signals for ADDRESS, CE, WE, BHE, BLE, and DATA I/O. Key timing parameters include t_{WC} , t_{SA} , t_{SC} , t_{AW} , t_{PW} , t_{IA} , t_{BW} , t_{SD} , and t_{HD} .

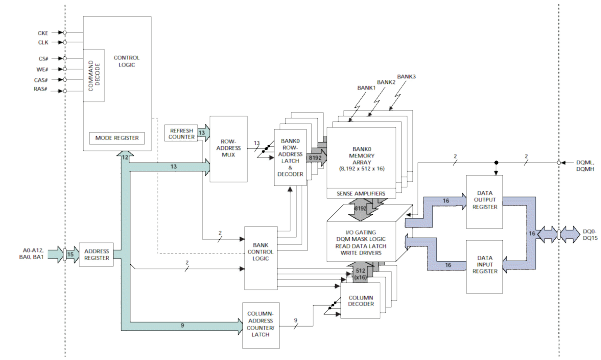
h.u. LINKÖPING UNIVERSITY

SDRAM; Synchronous Dynamic RAM

- Clocked device
- Memory element: Capacitance
- Needs periodic refreshing
- Pipelined operation
- Burst oriented
 - Single burst in our design
- 2 x (16M x 16) = 64MB

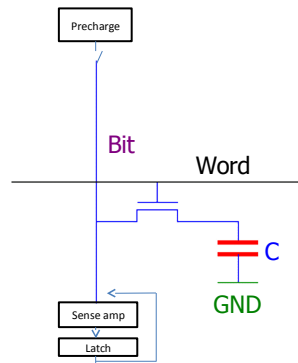


SDRAM Architecture

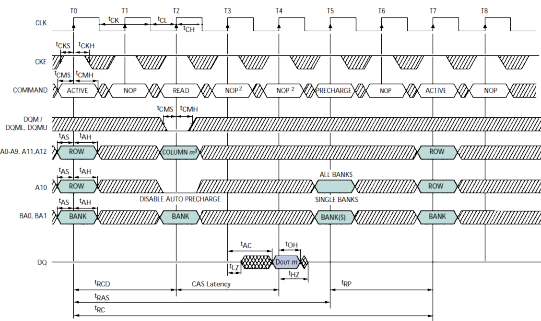


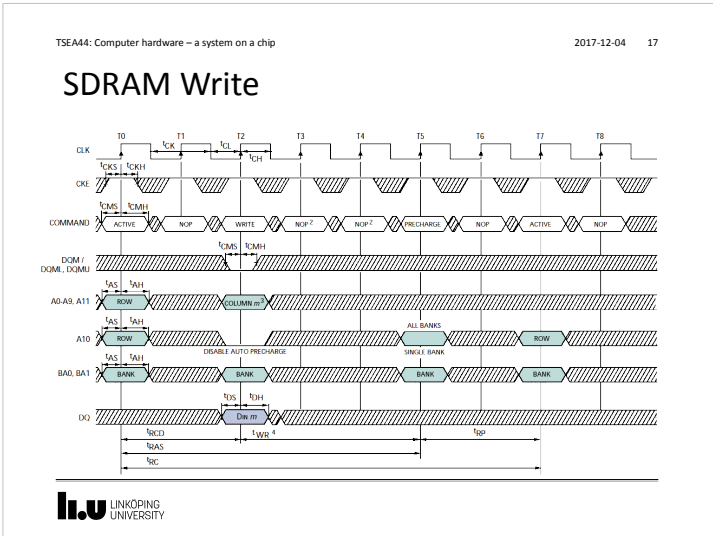
A measurement: make sim_jpeg

- Sketchy read cycle
 - Precharge bit line to Vdd/2
 - Let bit line float
 - Connect sense amp to bit line
 - Connect C to bit line
 - Hold value in latch
 - Write value back to C
- Refresh cycle
 - Dummy read cycle



SDRAM Read



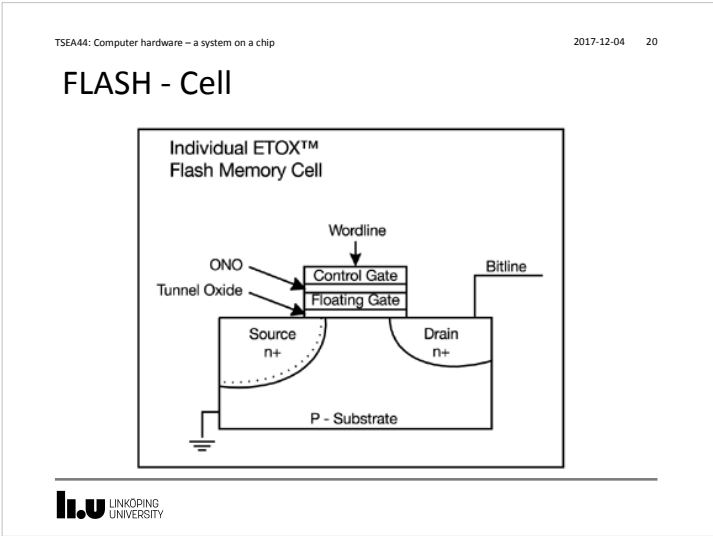
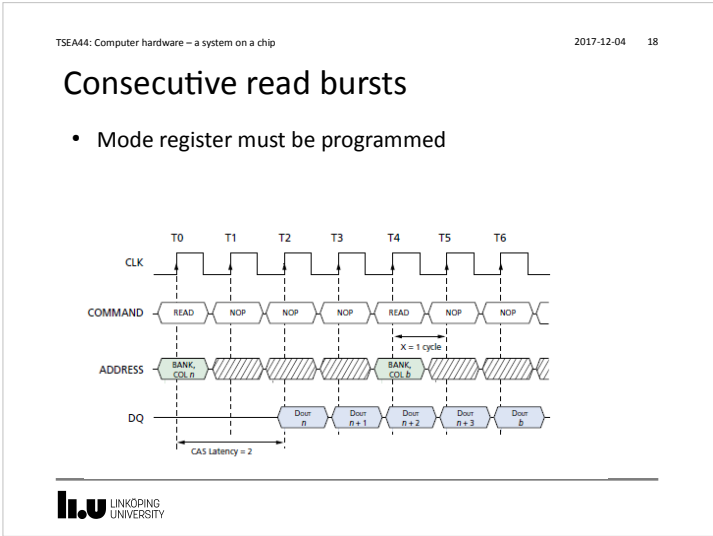


TSEA44: Computer hardware – a system on a chip 2017-12-04 19

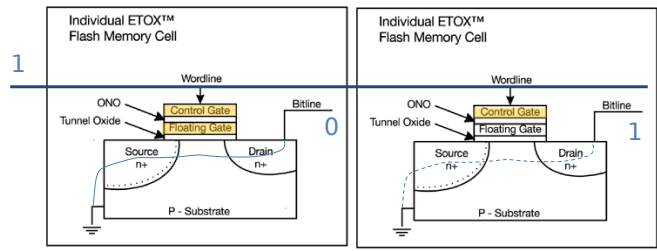
FLASH - Interface

- Looks like SRAM
 - Read
 - Write commands
- Erase is done in blocks
- Contains uLinux kernel + file system

LINKÖPING UNIVERSITY

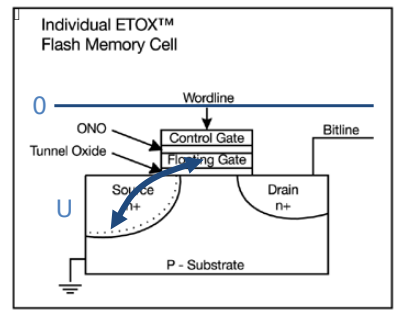


FLASH – Read (NOR)

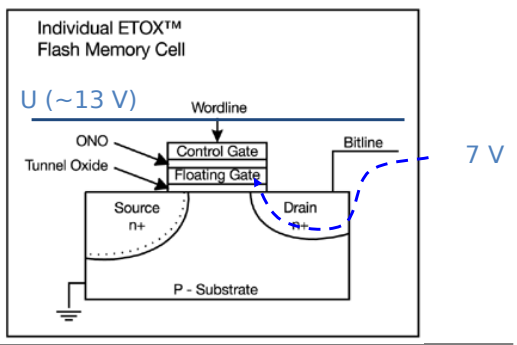


FLASH – Erase (to 1)

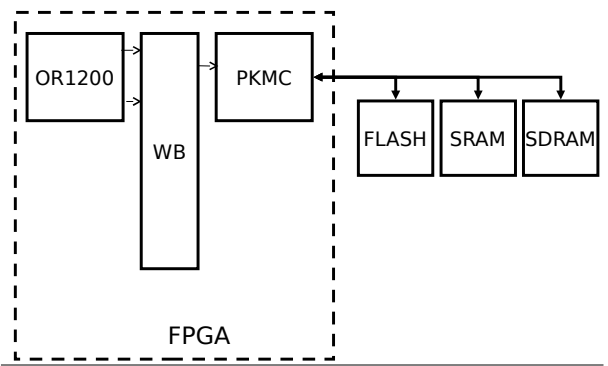
- Only blockwise



FLASH – Program (to 0)



System Overview



TSEA44: Computer hardware – a system on a chip 2017-12-04 25

PKMC internals

The diagram illustrates the internal structure of the PKMC. On the left, a dashed line represents the WB (Write Back) interface. Inside the PKMC block, there is an Addr Dec (Address Decoder) at the top, followed by three controllers: SRAM Ctrl, FLASH Ctrl, and SDRAM Ctrl. These controllers are connected to a central FPGA block. The FPGA is further connected to external components on the Board: SDRAM (S), Flash (F), and another SDRAM (S). Arrows indicate the flow of data and control signals between these components.

h.u LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 27

Refresh cycle

The diagram shows a timing diagram for the refresh cycle. The left side lists various signals: nL_cyc, nL_addr, nL_ack, state, refcount, clk, first, out, state, Slave 0 (SDRAM), sdr, ack, addr, cpr, data_o, err, sel, and we. The right side shows the corresponding waveforms. A yellow arrow points to the 'Start of refresh cycle' at approximately 112.065984 us. Another yellow arrow points to the 'Start of WB cycle' at approximately 111.6 us. The diagram shows that the refresh cycle occurs during a period where the WB cycle is active.

h.u LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 26

SDRAM Controller Internals

The diagram shows the internal components of the SDRAM Controller. A box labeled 'WB commands' has bidirectional arrows connecting to a central 'FSM' (Finite State Machine) block. The FSM block has a 'Command' output arrow pointing to the right. Below the FSM is a 'Refresh counter' block, which is connected to the FSM. A note next to the refresh counter states '7.2 μs between refreshes'.

h.u LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 28

Lab 4, Custom instruction

- Increase performance by adjusting instruction set
 - Specific for application domain
 - General purpose processor is general purpose
 - Not exceptionally good at anything
- Use profiling to find out the most timeconsuming part of the application code

h.u LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 29

Huffman Encoding/Decoding

1) After Q

22	12	0	-12	0	0	0	0
0	0	-8	0	0	0	0	0
4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

2) After zig-zag

```

22
12
04
00-12
-8
0000000000000000
0000000000000000
000000000000001
                
```

3) After RLE

Value	raw bits	(amplitude value)
05	10110	
04	1100	-12 =>
13	100	12-1, force MSB=0
24	0011	=> 0011
04	0111	
F0		-8 =>
F0		8-1, force MSB=0
D1	1	=> 0111
00		

Run of 0's Magnitude

4) Huffman coding

- Value are HC (variable length) using table lookup
- raw bits are left untouched

li.U LINKÖPING UNIVERSITY reg

TSEA44: Computer hardware – a system on a chip 2017-12-04 31

jpegfiles

jpegtest.c, jcdctmgr.c, jdct.c, jchuff.c

```

draw_image()
  |
  |-----> init_huffman() -----> "write header,
  |                                     init your HW"
init_encoder() |
  |-----> init_image() -----> "init some variables"
  |
  |-----> encode_image() -----> forward_DCT() -----> "quantize"
  |                                     |
  |                                     |-----> encode_mcu_huff() -----> emit_bits()
  |
  |-----> finish_pass_huff() -----> flush_bits() -----> "flush remaining bits"
                
```

li.U LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 30

Huffman in JFIF

- Output: 1 – 16 bits
- Encodes bytes
- 2 tables used
 - Y DC
 - Y AC

li.U LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 32

Emit_bits()

```

/* Only the right 24 bits of put_buffer are used; the valid bits are left-justified in
 * this part. At most 16 bits can be passed to emit_bits in one call, and we never retain
 * more than 7 bits in put_buffer between calls, so 24 bits are sufficient.
 */
static void emit_bits (unsigned int code, int size)
{
    unsigned int startcycle;
    new_put_buffer = (int) code;

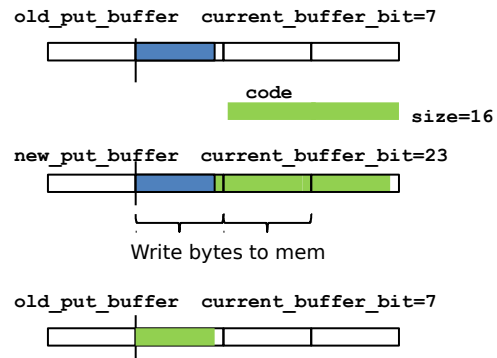
    // Add new bits to old bits. If at least 8 bits then write a char to buffer,
    // save the rest until we get more bits.

    new_put_buffer &= (1<<size) - 1; /* mask off any extra bits in code */
    current_buffer_bit += size; /* new number of bits in buffer */
    new_put_buffer = new_put_buffer << (24 - current_buffer_bit); /* align incoming bits */
    new_put_buffer = new_put_buffer | old_put_buffer; /* and merge with old buffer contents */

    while (current_buffer_bit >= 8) {
        int c = (new_put_buffer >> 16) & 0xFF; /* Mask out the 8 bits we want
        buffer[next_buffer] = (char) c;
        next_buffer++;
        if (c == 0xFF) { /* 0xFF is a reserved code for tags, if we get image data
            buffer[next_buffer] = 0x00; /* with an FF value it has to be followed by 0x00.
            next_buffer++;
        }
        new_put_buffer <<= 8;
        current_buffer_bit -= 8;
    }
    old_put_buffer = new_put_buffer; /* update state variables */
}
                
```

li.U LINKÖPING UNIVERSITY

Emit_bits()



Instruction Selection

- I.custx
 - No operands
- Instructions for 64 bit
 - Not used
 - Assembler can understand
 - I.sd I(rA),rB

Adding an Instruction

1. Instruction Selection
2. Hardware modification
3. Assembler modification
4. Compiler modification

Hardware Modifications

- Instruction decoder modifications
 - Legal instruction
 - or1200_ctrl.v
- Special purpose register
 - New group
 - or1200_sprs.v
- Data path
 - New hardware
 - or1200_lsu.v
 - or1200_vlx_top.v

TSEA44: Computer hardware – a system on a chip 2017-12-04 37

Or1200 Pipeline

code size
↓ ↓
l.sd (rA), rB

align

IF
ID
EX
WB

Register File
ALU
Data Memory
vix

li.u LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 39

Align reg2mem

li.u LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 38

Or1200 Pipeline

- Remember stall

	1	2	3	4	5	6	7
IF	ld	add	sub	-			
ID/RR		ld	add	-	sub		
EX/M			ld	ld	add	sub	
W				-	ld	add	sub

li.u LINKÖPING UNIVERSITY

TSEA44: Computer hardware – a system on a chip 2017-12-04 40

Proposed Architecture

Internal regs (mapped as SPR)
bit_reg
bit_reg_wr_pos
vix_addr_o

size code
↑stall_cpu_o

Data path ↔ Control Unit

Store unit

adr ↓ data ↓ store_byte_o ↓

li.u LINKÖPING UNIVERSITY

Inline asm

In jpegfiles insert:

```
asm volatile("l.sd 0x0(%0),%1" : : "r"(code), "r"(size));
```

template

input input

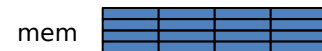
=> code and size will show up at your vlx

Control

- May not be needed
- May be an FSM

Data Path

- Fill buffer alternatives
 - One bit/clock cycle
 - All bits at once
- Write to mem alternatives
 - One byte
 - One 32 bit word, must be on word boundaries



Store Unit

- Stores the data
- 0xFF stored as 0xFF00
 - JPEG markers
- Only byte alignment!
 - Parallel stores faster

Software

- New assembler
 - Easy
- New compiler
 - Hard problem for complex instructions
 - Compiler knows functions
- C
 - Inline Assembler

Instruction Usage

```
unsigned char* sb_get_buff_pos(void)
{
    unsigned char* pos;
    asm volatile("l.mfspr %0,%1,0x2" : "=r" (pos) : "r" (0xc000));
    return pos;
}
```

output

```
00000250 <sb_get_buff_pos>:
250: 9c 21 ff fc    l.addi r1,r1,0xffffffffc
254: d4 01 10 00    l.sw 0x0(r1),r2
258: 9c 41 00 04    l.addi r2,r1,0x4
25c: a9 60 c0 00    l.ori r11,r0,0xc000
260: b5 6b 00 02    l.mfspr r11,r11,0x2
264: 84 41 00 00    l.lwz r2,0x0(r1)
268: 44 00 48 00    l.jr r9
26c: 9c 21 00 04    l.addi r1,r1,0x4
```
