EITF35: Introduction to Structured VLSI Design

Part 3.2.1: Storage Elements

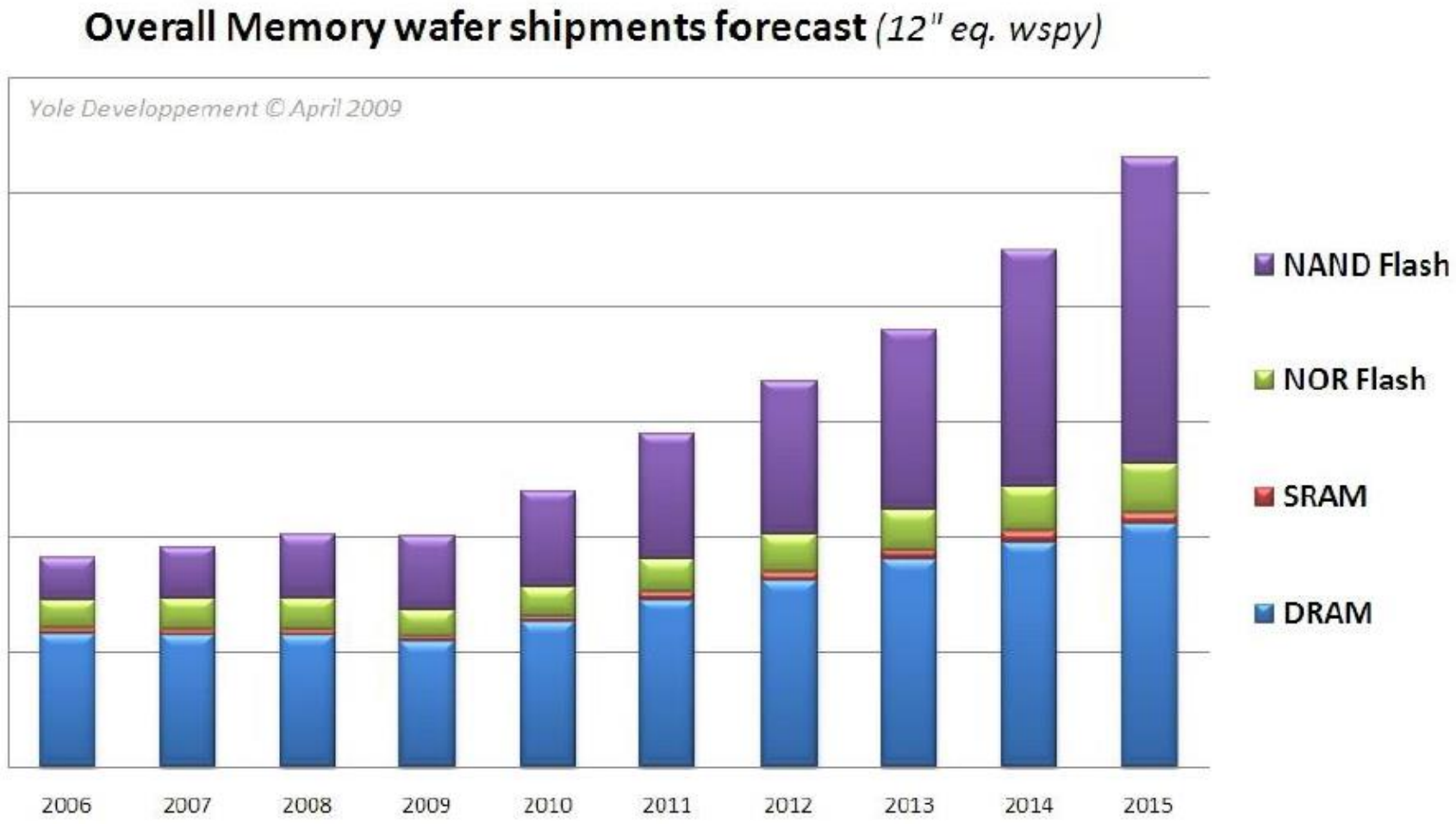
Liang Liu
liang.liu@eit.lth.se

Outline

- **Overview of Memory**
  - Application, history, trend
  - Different memory type
  - Overall architecture
- **Registers as Storage Element**
  - Register File
  - FIFO
- **Xilinx Storage Elements**

Storage elements

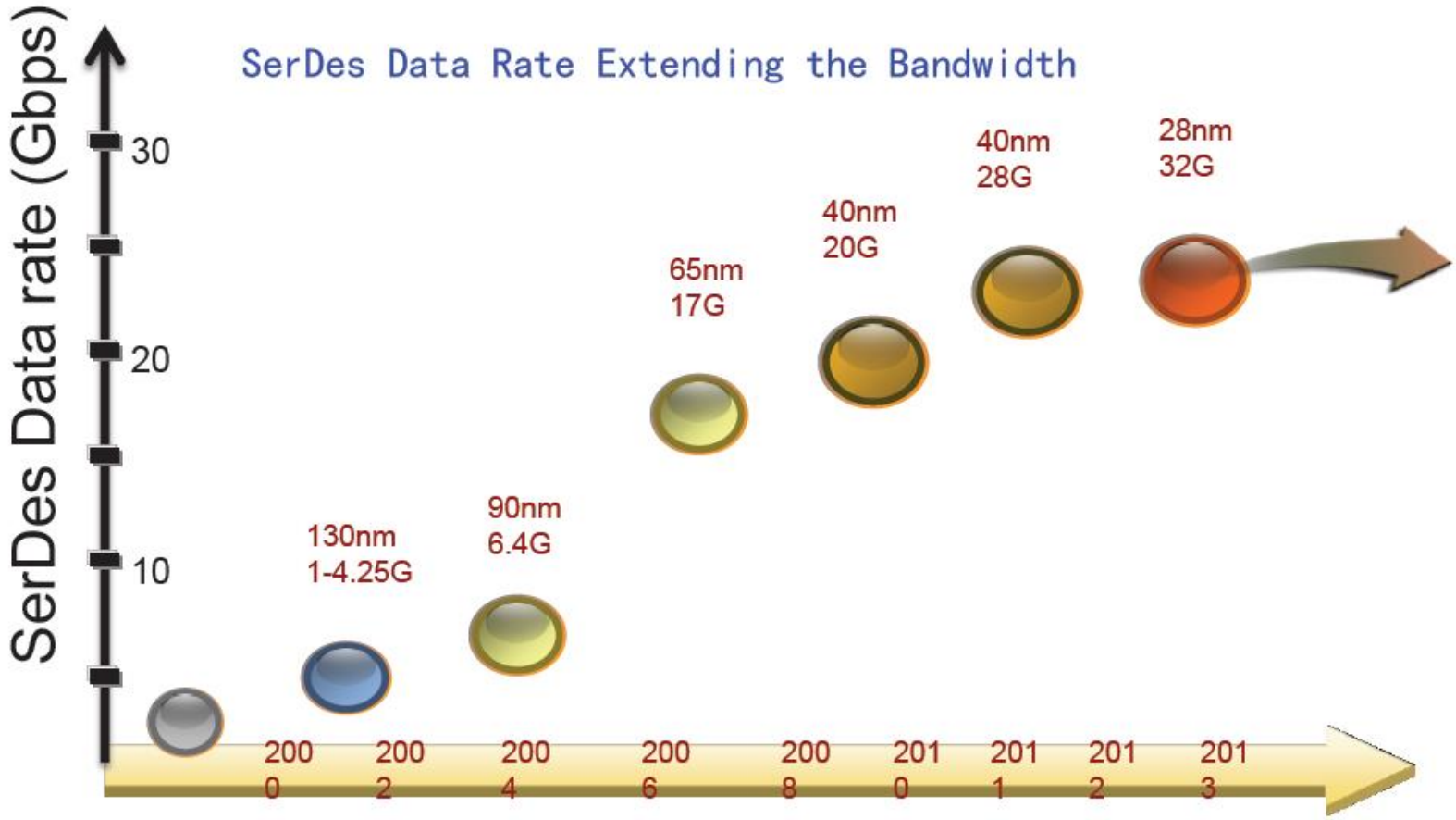
Memory Wafer Shipments Forecast



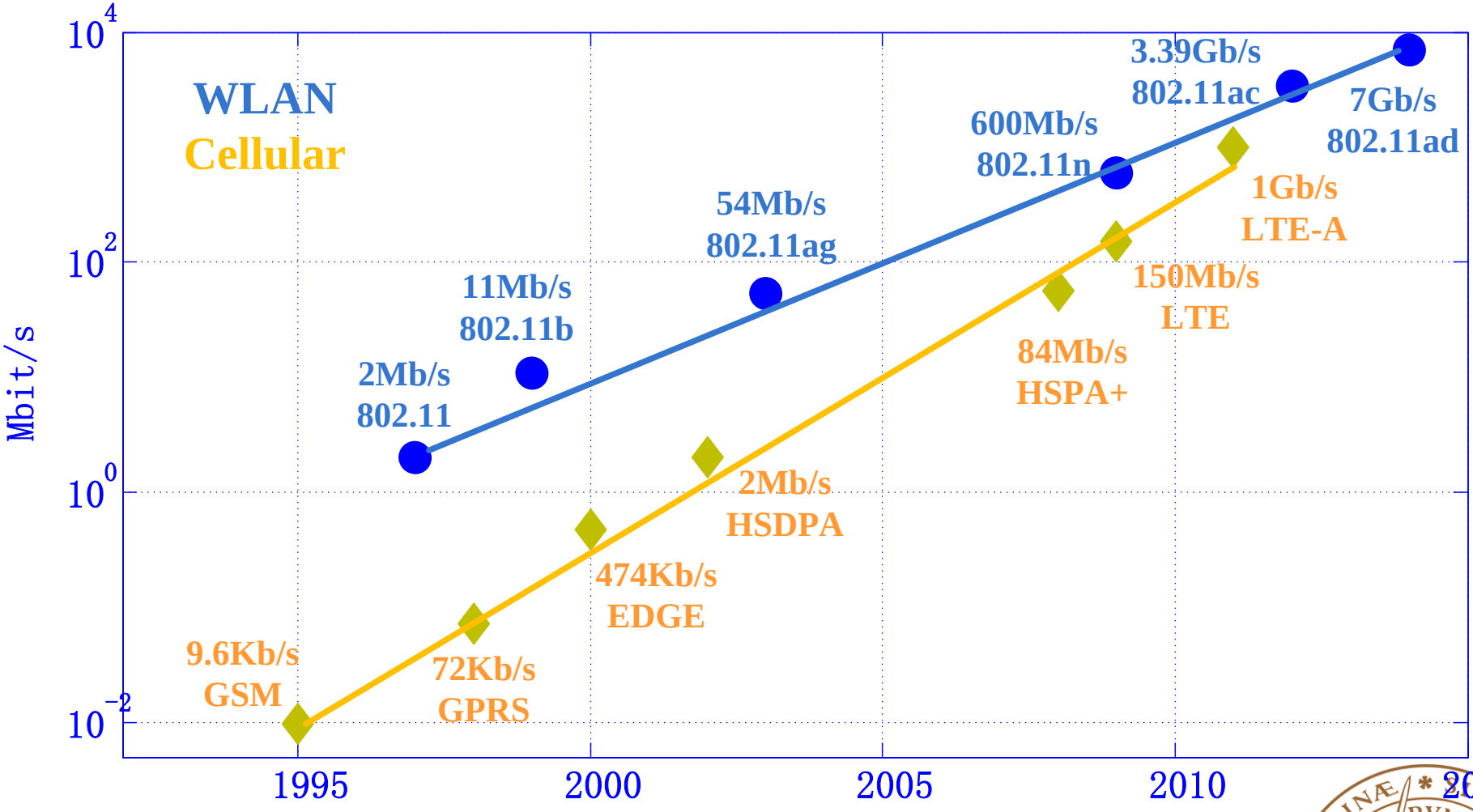
Faster-Than-Moore?

Bits shipped routinely doubles-to-triples year-over-year

Bandwidth

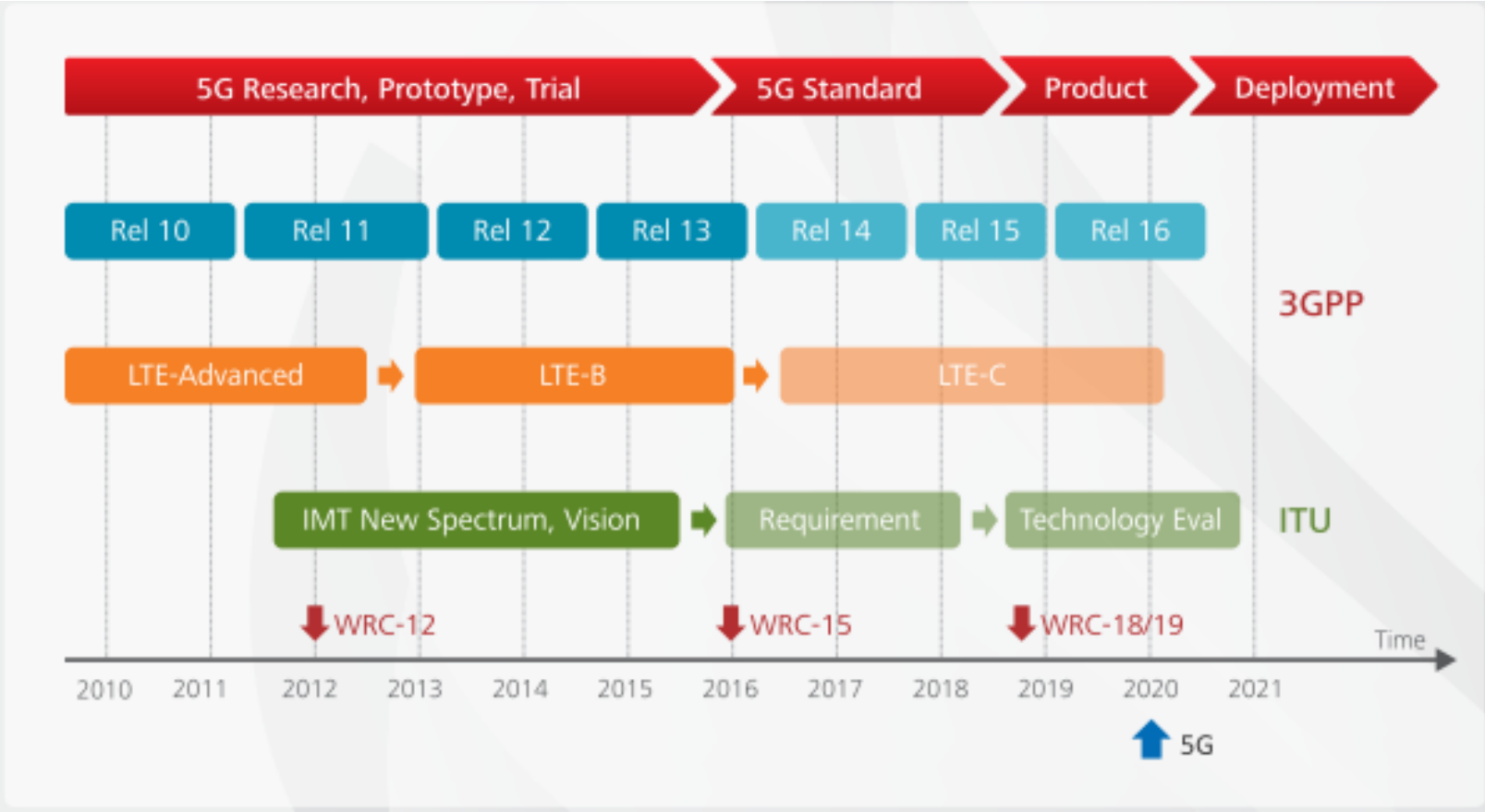

SerDes Data Rate Extending the Bandwidth
SerDes Data rate (Gbps)
30
20
10
130nm
1-4.25G
90nm
6.4G
65nm
17G
40nm
20G
40nm
28G
28nm
32G
2000
2002
2004
2006
2008
2010
2011
2012
2013

Bandwidth (cont'd.)



Bandwidth (cont'd.)

Bandwidth (cont'd.)

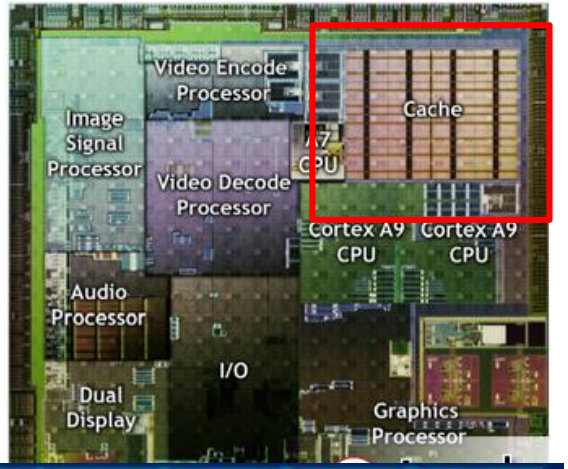


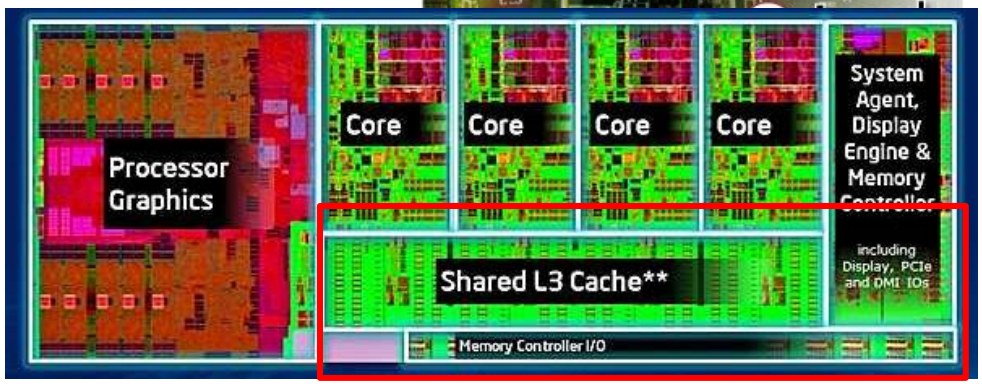
Memory Bandwidth: 1TB/s by 2017

Memories, on chip

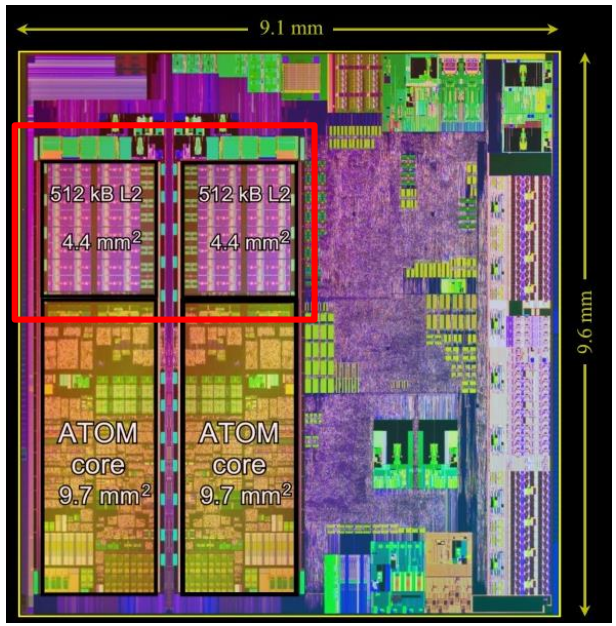
- Power and Bandwidth becomes bottleneck
- Everything is pointing to more and more “local” memory/storage at the device level

Nvidia Tegra 2





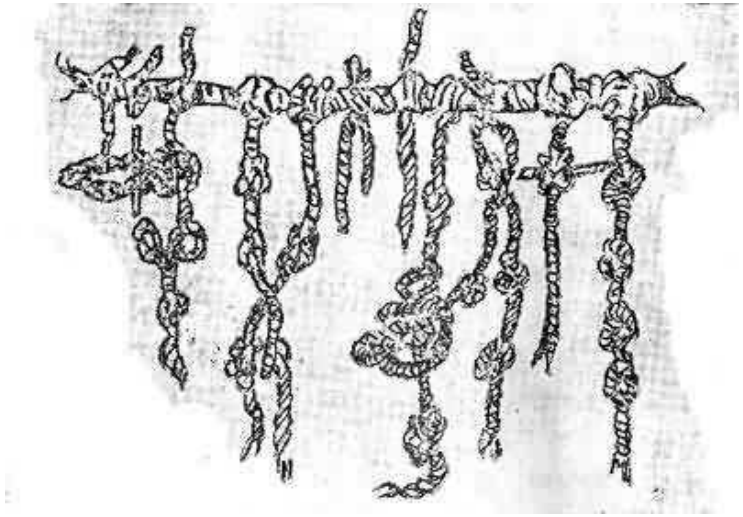
Intel Haswell



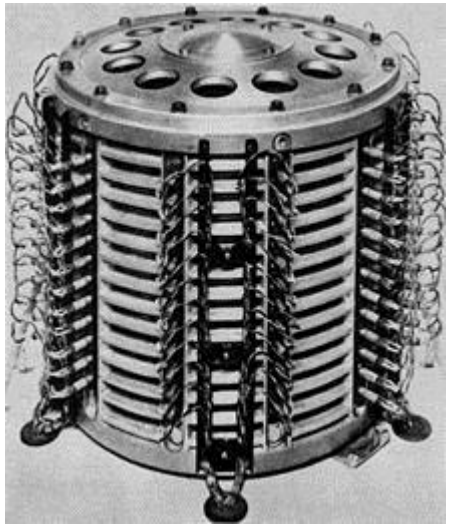
Intel ATOM

Memories, History

- **First Storage?**

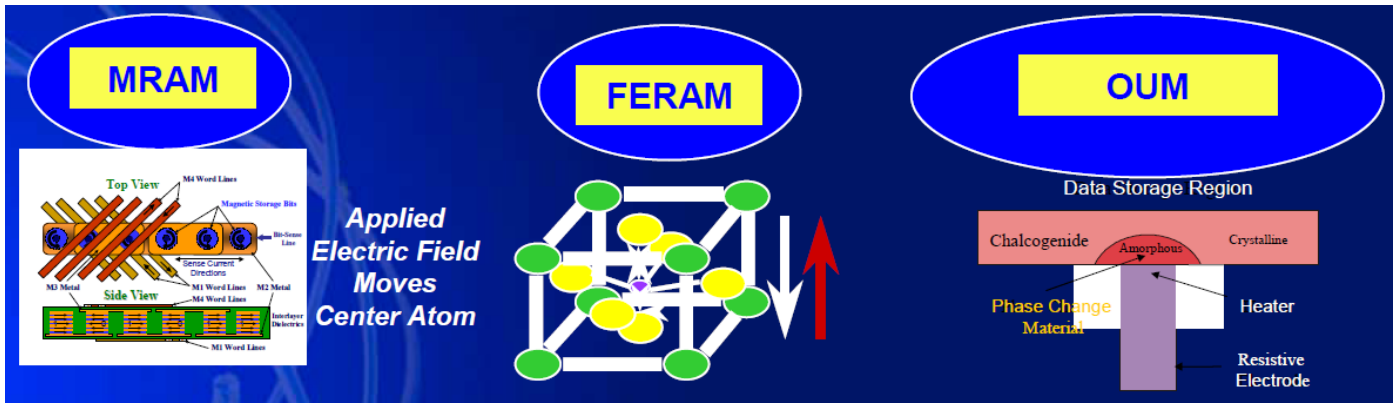


- **Early Memory**
  - Drum memory: magnetic data storage device.
  - Gustav Tauschek (1932)
  - Widely used in the 1950s and into the 1960s as computer memory



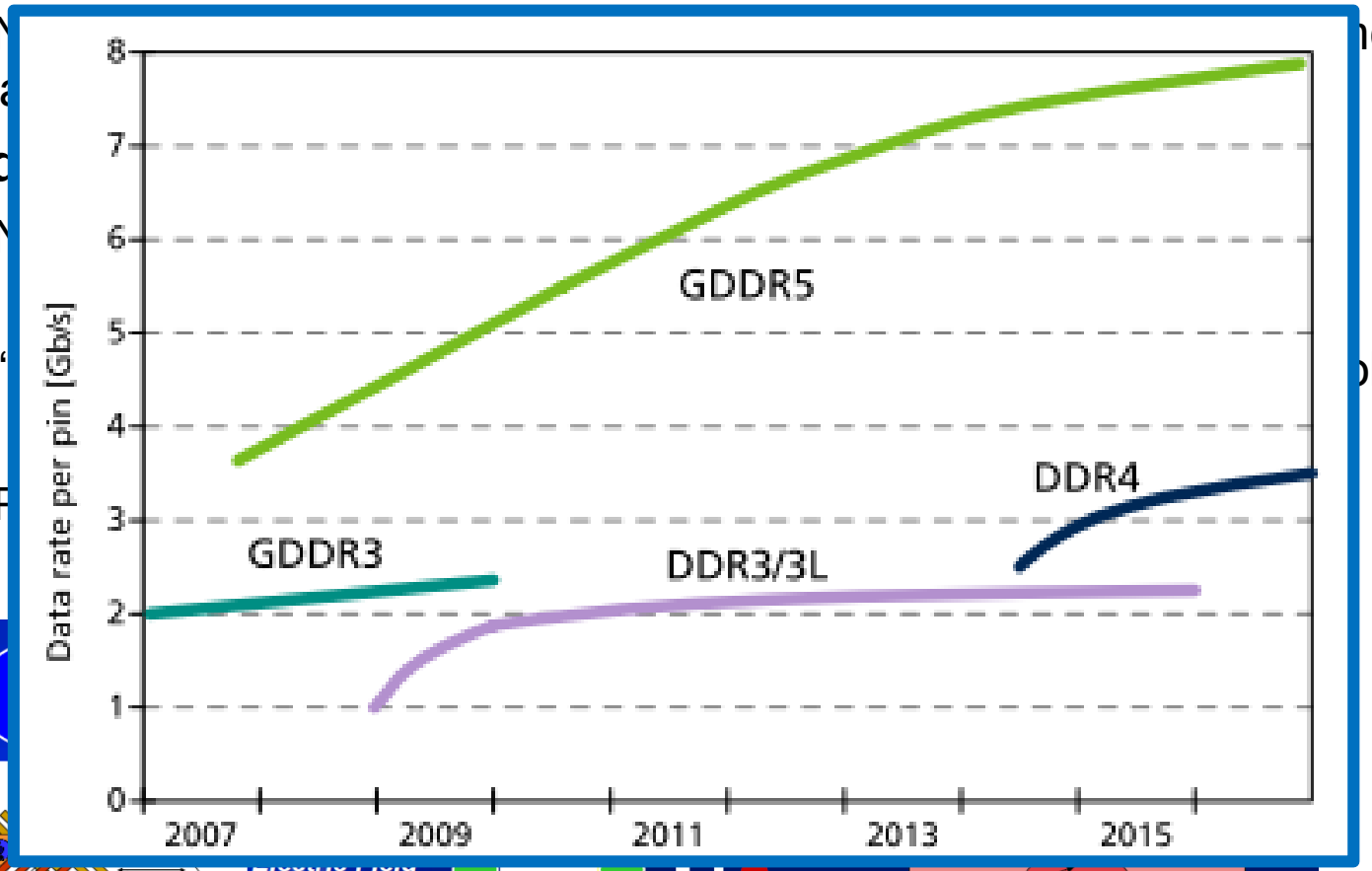
Memory, current state

- **Yesterday:**
  - RAM memories are historically driven by computing applications
  - NOR/NAND Flash is used in most of consumer devices (cell-phone, digital camera, USB stick …)
- **Today:**
  - New generation memories
    - ***PRAM, FeRAM, MRAM..***
  - "Solid State" memory is the killer application for NAND Flash in volume:
    - ***SSDs to replace HDD (hard disk magnetic drives)***
  - RAM (SRAM / DRAM)
    - ***DDR3 / DDR4 /GDDR5/GDDR6***

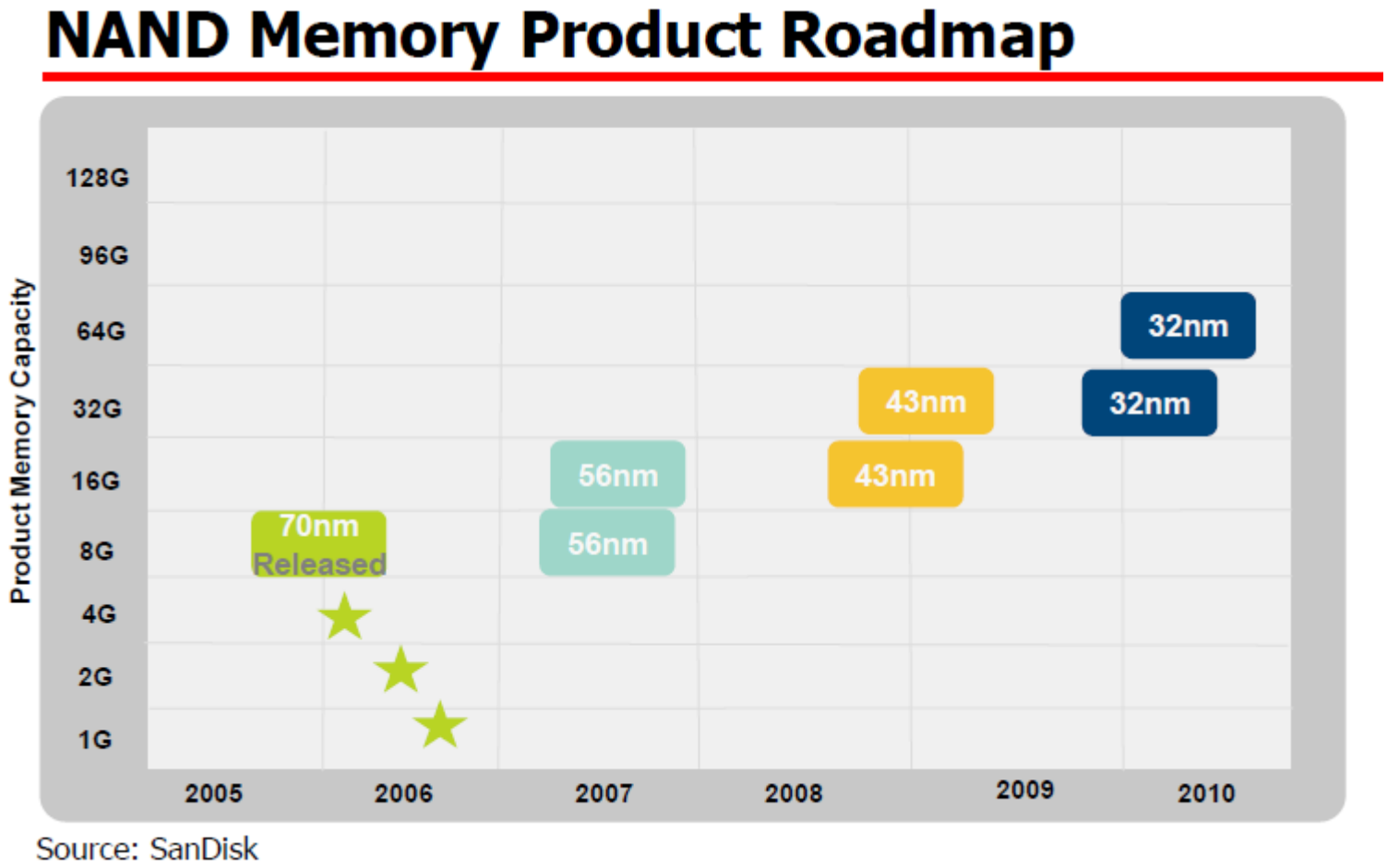


Memory, current state

- **Yesterday:**
  - RAM memories are historically driven by computing applications
  - N… …e, digital ca…
- **Toc…**
  - N…
  - "… …olume:
  - P…



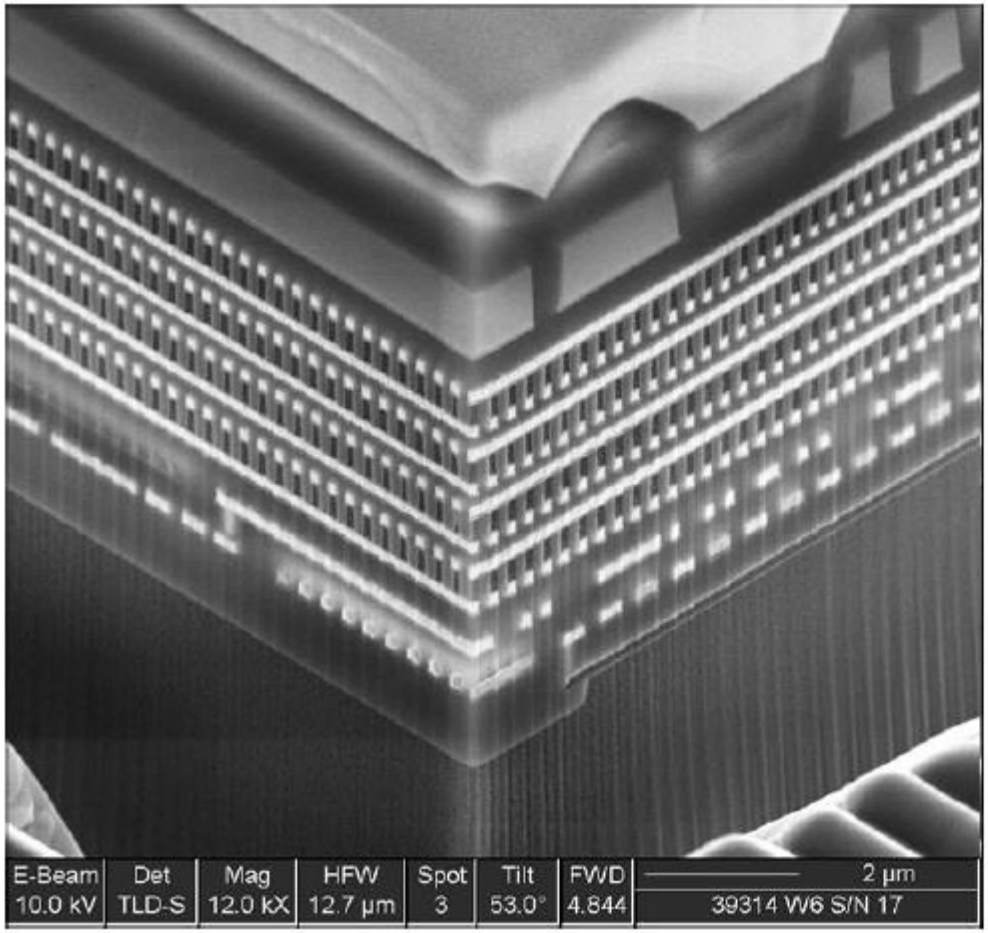
Memory, leading the semiconductor tech.



First 32nm NAND Flash memory, 2009, Toshiba

First 32nm CPU released, 2010, Intel Core i3

Memory, leading the semiconductor tech.

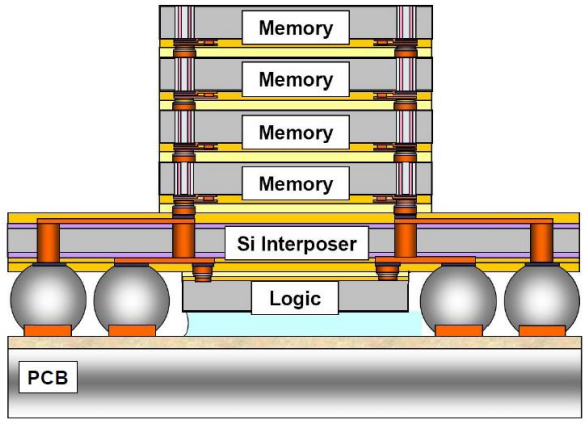


Al top metal

4 layers of memory cells + tungsten interconnect

2 levels of tungsten routing

LV + HV CMOS logic



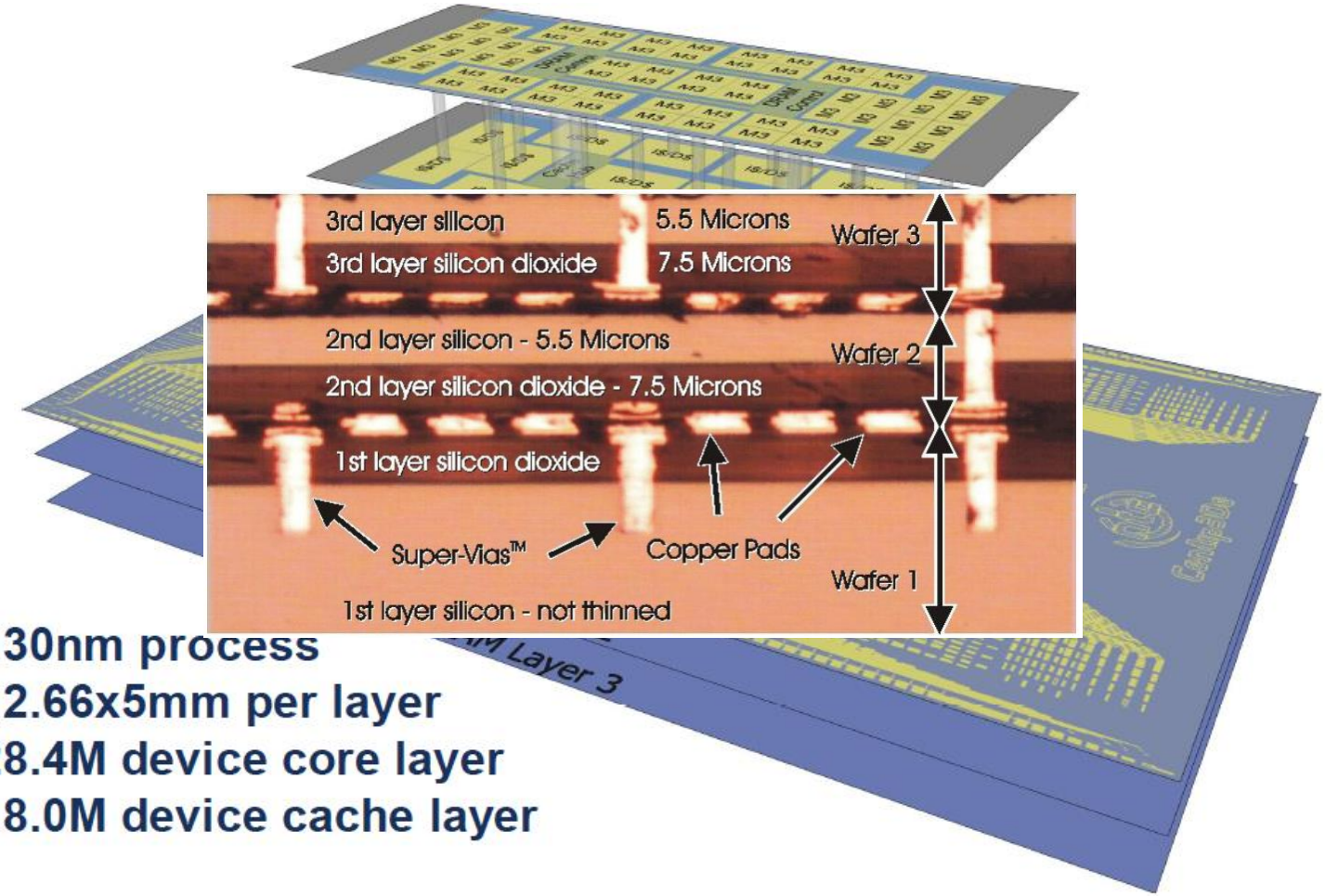
Example of 3-D integrated construction *(Image courtesy of DuPont Electronics)*

First 22-nm SRAMs using Tri-Gate transistors, in Sept.2009
First 22-nm Tri-Gate microprocessor (Ivy Bridge), released in 2013

3D Processors for Massive Parallel Computing

❑ Centip3De: University of Michigan

- Configurable 3D stacked system with 64 ARM Cortex-M3 cores
- 7-layer system (2 core layers, 2 cache layers, 3 DRAM layers)



Intel again…

Intel and Micron Produce Breakthrough Memory Technology

Posted by IntelPR in Intel Newsroom on Jul 28, 2015 9:00:28 AM

Tweet 1,210 Like 5.5k G+1 517

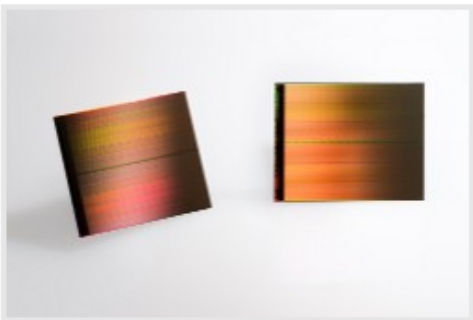
New Class of Memory Unleashes the Performance of PCs, Data Centers and More

NEWS HIGHLIGHTS

- Intel and Micron begin production on new class of non-volatile memory, creating the first new memory category in more than 25 years.
- New 3D XPoint™ technology brings non-volatile memory speeds up to 1,000 times faster[1] than NAND, the most popular non-volatile memory in the marketplace today.
- The companies invented unique material compounds and a cross point architecture for a memory technology that is 10 times denser than conventional memory[2].
- New technology makes new innovations possible in applications ranging from machine learning to real-time tracking of diseases and immersive 8K gaming.

SANTA CLARA, Calif., and BOISE, Idaho, July 28, 2015 – Intel Corporation and Micron Technology, Inc. today unveiled 3D XPoint™ technology, a non-volatile memory that has the potential to revolutionize any device, application or service that benefits from fast access to large sets of data. Now in production, 3D XPoint technology is a major breakthrough in memory process technology and the first new memory category since the introduction of NAND flash in 1989.

The explosion of connected devices and digital services is generating massive amounts of new data. To make this data useful, it must be stored and analyzed very quickly, creating challenges for service providers and system builders who must balance cost, power and performance trade-offs when they design memory and storage solutions. 3D XPoint technology combines the

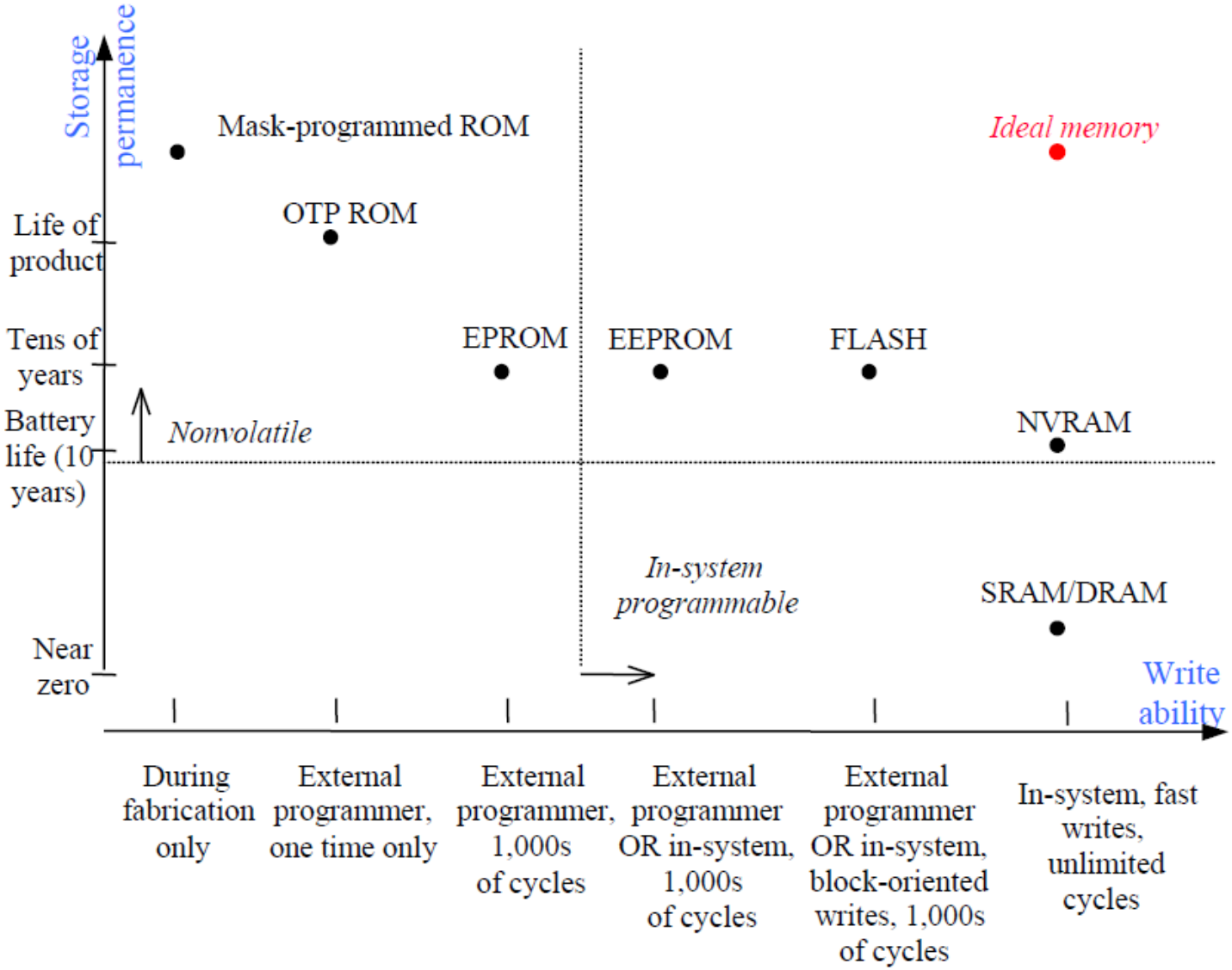


3D Xpoint™ technology is up to 1000x faster than NAND and an individual die can store 128Gb of data

Memory Classification

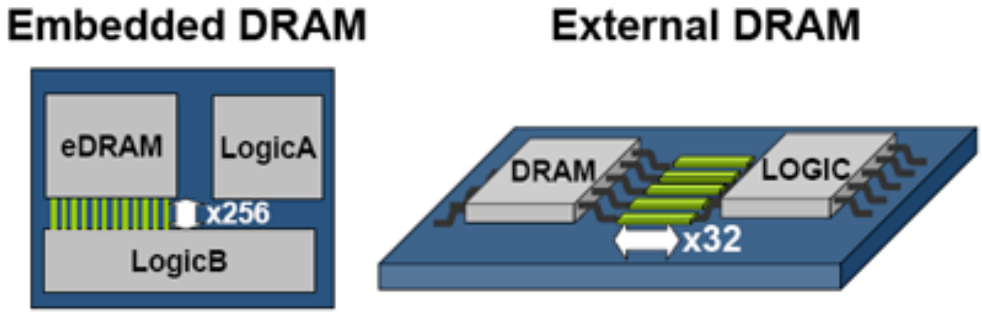
Read-Write Memory		Non-Volatile Read-Write Memory	Read-Only Memory
Random Access	Non-Random Access	EPROM E^2PROM FLASH	Mask-Programmed Programmable (PROM)
SRAM DRAM	FIFO LIFO Shift Register CAM		

Memory Classification

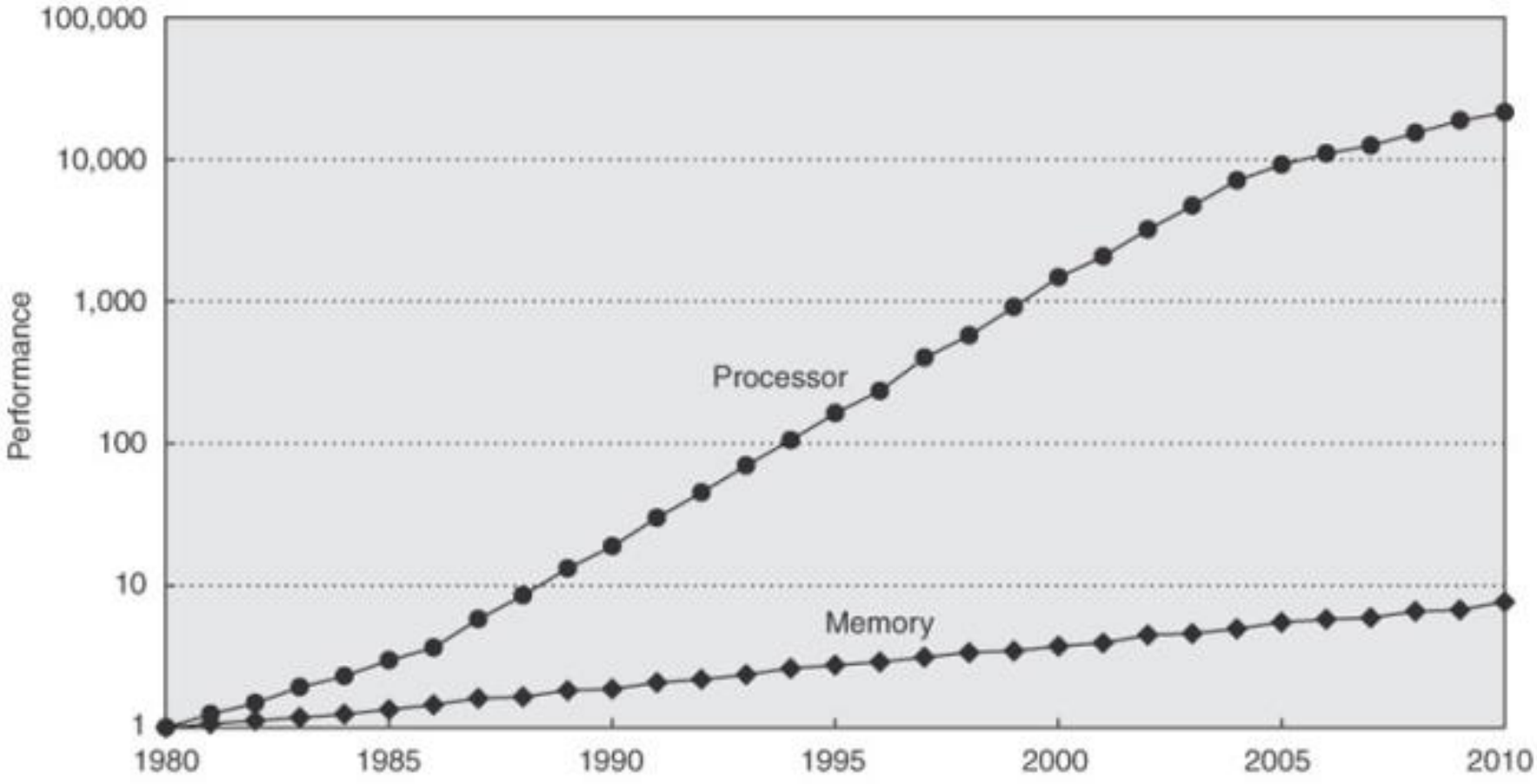


Picture from Embedded Systems Design: A Unified Hardware/Software Introduction

Embedded DRAM

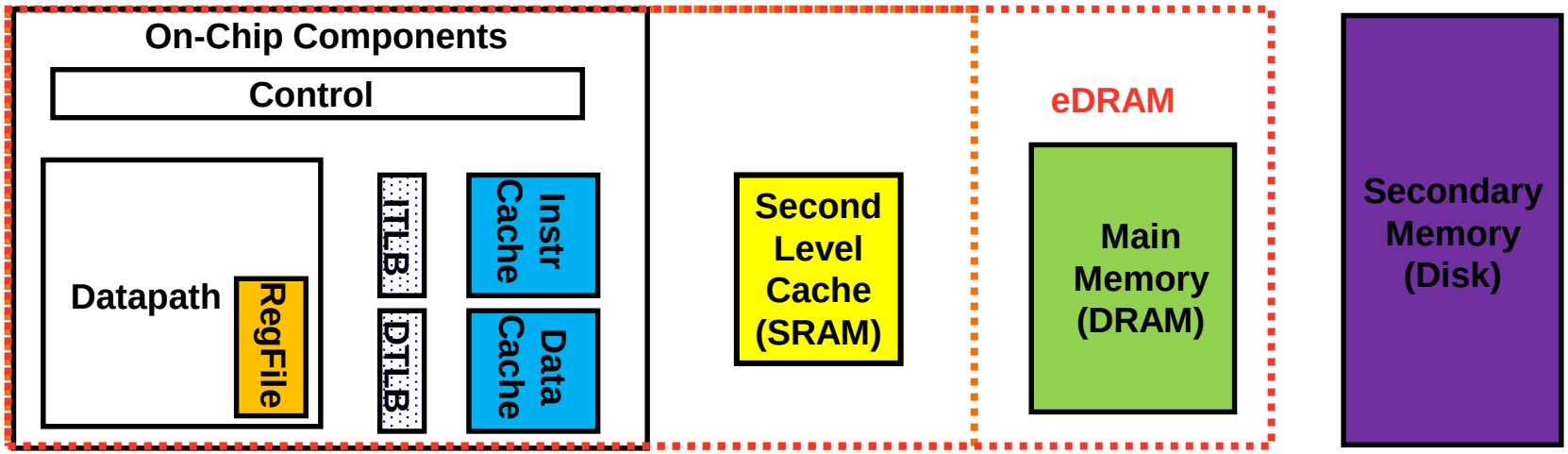


Memory Processor Gap



Memory Hierarchy

Speed (ns):	.1's	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	T's
Cost:	highest				lowest



Heterogeneous is important

The concept of "most suitable"

Hierarchy, Heterogeneous

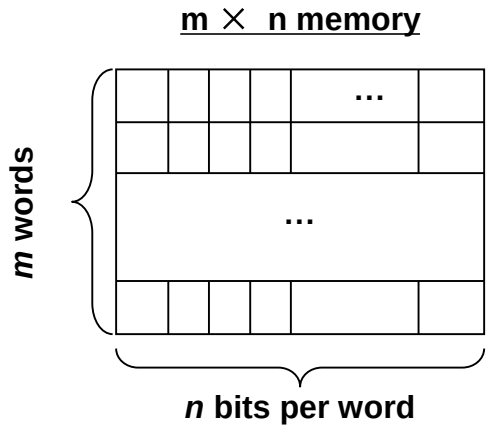
Hierarchy, Heterogeneous

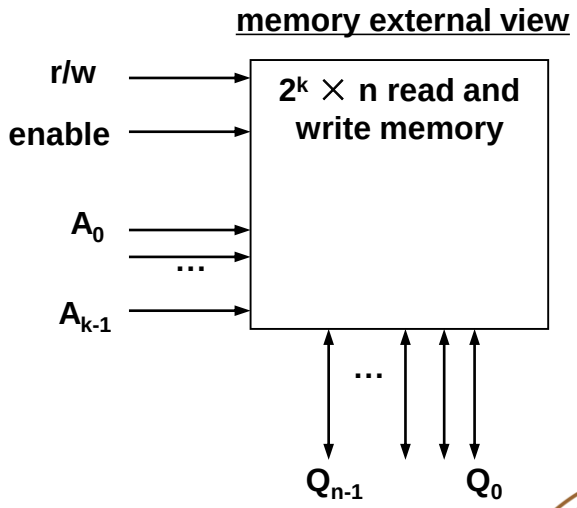
Linköping

Copenhagen

Memory Basic Concept

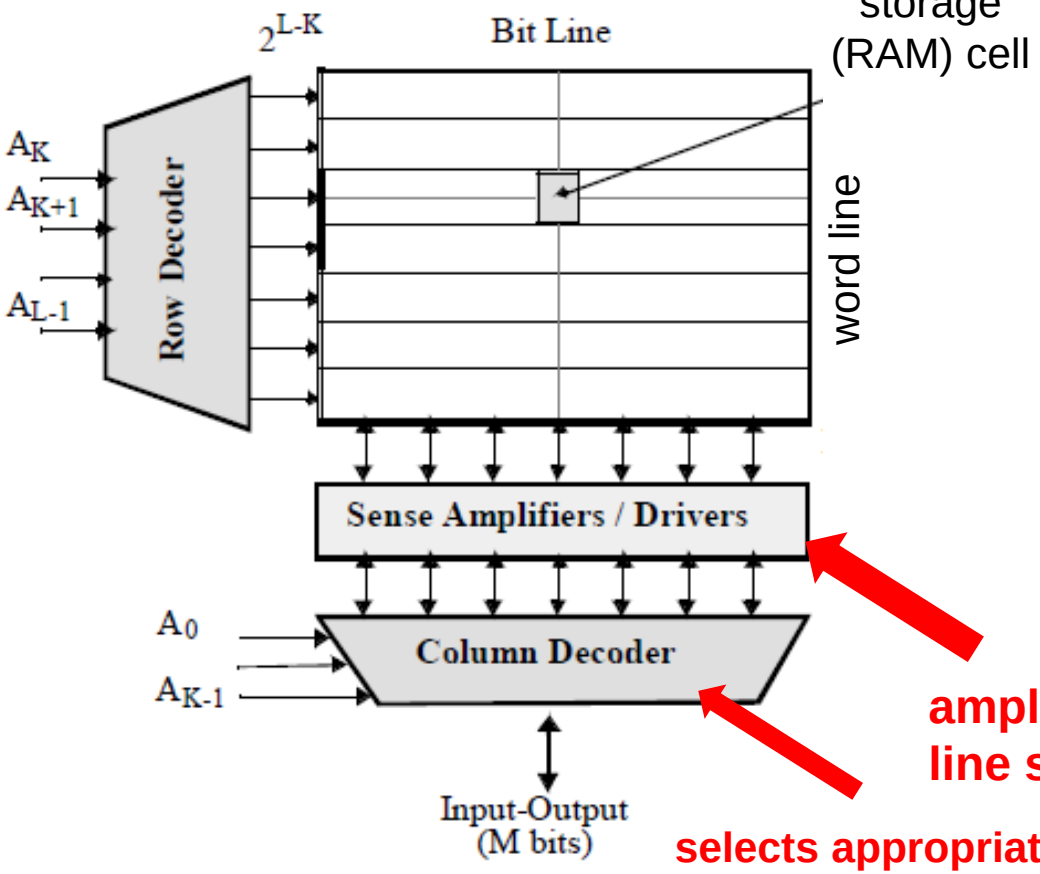
- **Stores large number of bits**
  - m x n: m words of n bits each
  - k = $Log_2(m)$ address input signals
  - or m = 2^k words
  - e.g., 4096 x 8 memory:
    - ***32,768 bits***
    - ***12 address input signals***
    - ***8 input/output data signals***
- **Memory access**
  - r/w: selects read or write
  - enable: read or write only when asserted
  - Address
  - Data-port

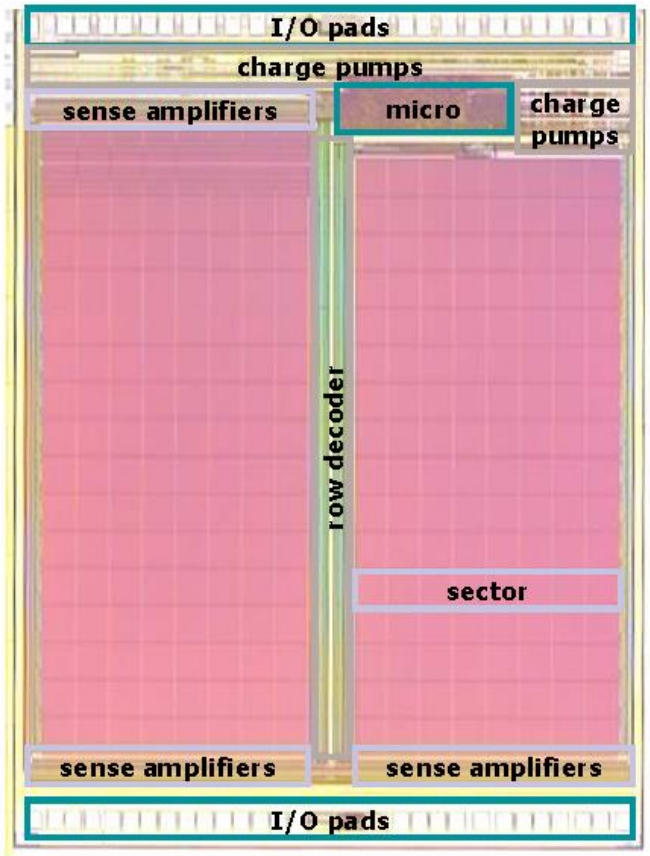




We stay at higher-level, gate-level view of memory will be taught at Digital IC Design

Memory Architecture





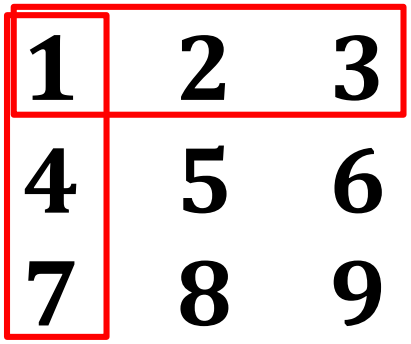
amplifies bit line swing

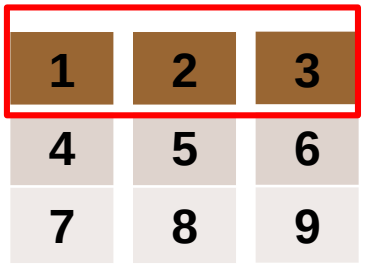
selects appropriate word from memory row

Flexible access

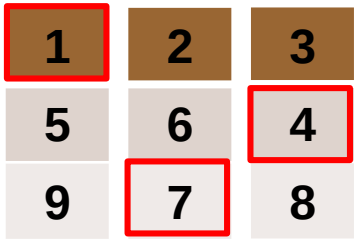
Matrix storage

- Accesss row-wise or colunm-wise in one clock cycle





Multi-bank memory

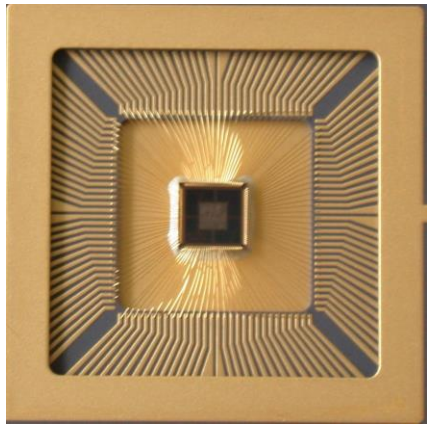


Interleaved storage

Outline

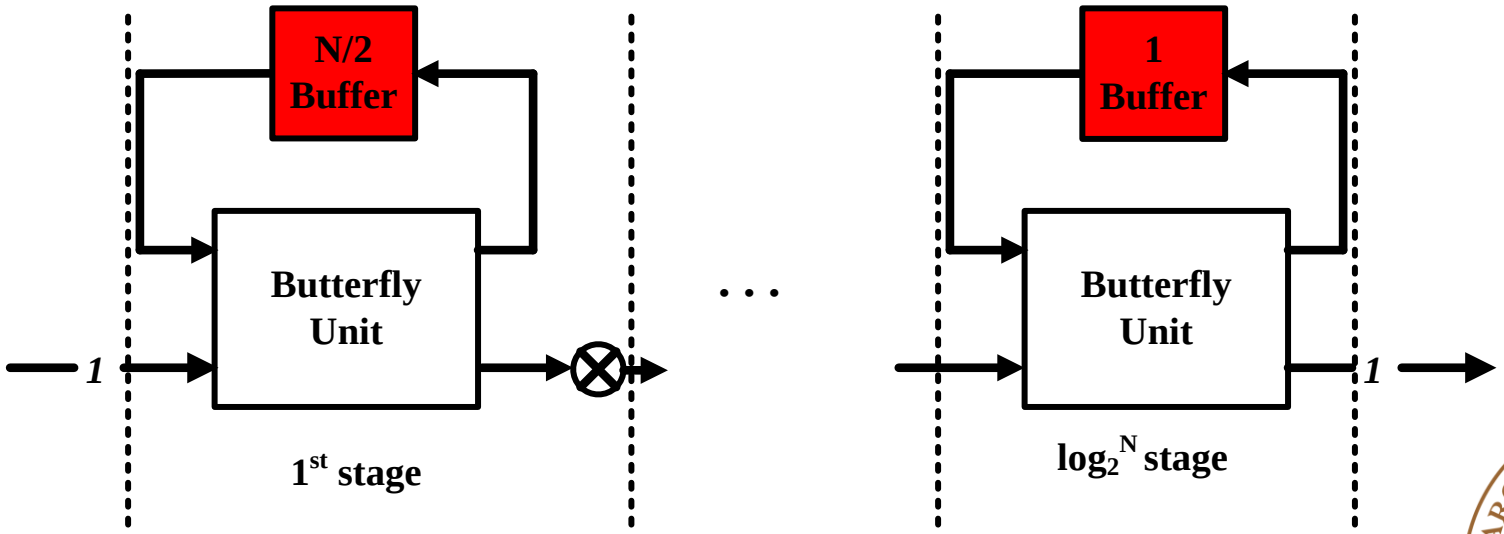
- Overview of Memory
  - Application, history, trend
  - Different memory type
  - Overall architecture
- **Registers as Storage Element**
  - Register File
  - FIFO
- Xilinx Storage Elements

Storage Examples 1



Register File

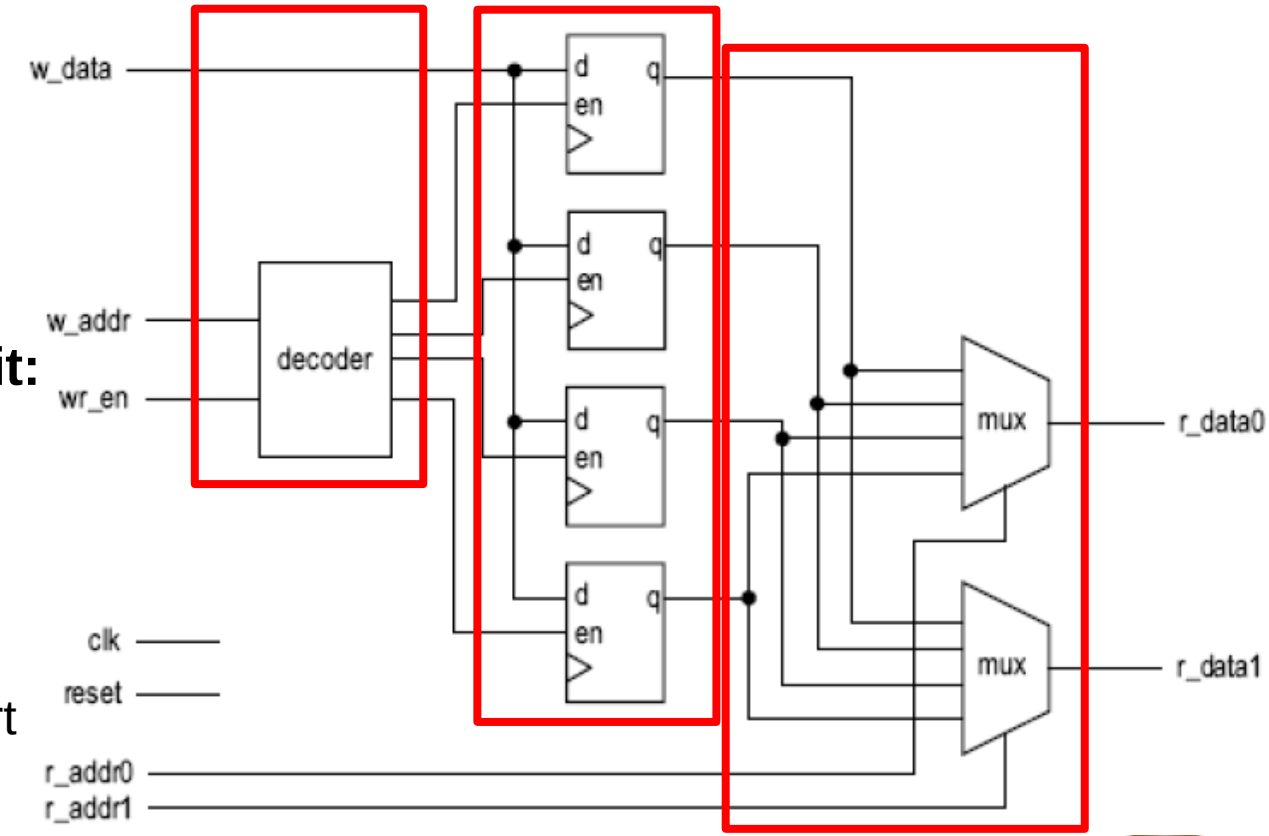
- Used as ***fast temporary*** storage
- Registers arranged as ***array***
- Each register is identified with an address
- Normally has 1 write port (with write enable signal)
- Can has multiple read ports



Register File

- **Example:** 4-word register file with 1 write port and two read ports

- **Register array:**
  - 4*16bit registers
  - Each register has an enable signal
- **Write decoding circuit:**
  - 0000 if wr_en is 0
  - 1 bit asserted according to w_addr if wr_en is 1
- **Read circuit:**
  - A mux for each read port



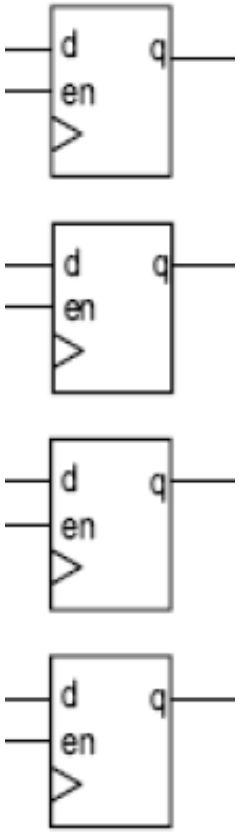
VHDL: a parameterized 2^W-by-B register file

```
library ieee;
use ieee.std_logic_1164.all;
entity reg_file is
   port(
      clk, reset: in std_logic;
      wr_en: in std_logic;
      w_addr: in std_logic_vector(1 downto 0);
      w_data: in std_logic_vector(15 downto 0);
      r_addr0, r_addr1: in std_logic_vector(1 downto 0);
      r_data0, r_data1: out std_logic_vector(15 downto 0)
      );
end reg_file;

architecture no_loop_arch of reg_file is
   constant W: natural:=2; -- number of bits in address
   constant B: natural:=16; -- number of bits in data
   type reg_file_type is array (2**W-1 downto 0) of
        std_logic_vector(B-1 downto 0);
   signal array_reg: reg_file_type;
   signal array_next: reg_file_type;
   signal en: std_logic_vector(2**W-1 downto 0);
```

A user-defined array-of-array data type is introduced

VHDL: a parameterized 2^W-by-B register file

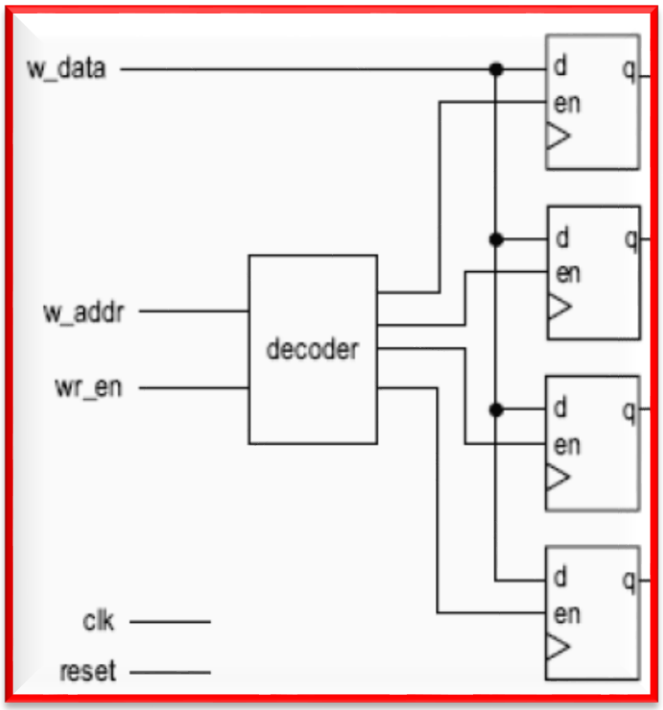



```
process(clk, reset)
begin
   if (reset='1') then
      array_reg(3) <= (others=>'0');
      array_reg(2) <= (others=>'0');
      array_reg(1) <= (others=>'0');
      array_reg(0) <= (others=>'0');
   elsif (clk'event and clk='1') then
      array_reg(3) <=  array_next(3);
      array_reg(2) <=  array_next(2);
      array_reg(1) <=  array_next(1);
      array_reg(0) <=  array_next(0);
   end if;
end process;
```

- **Index to access an element in the array**
  - ***s(i)*** to access the ith row of the array s
  - ***S(i)(j)*** to access the jth element of ith row in the array

VHDL: a parameterized 2^W-by-B register file

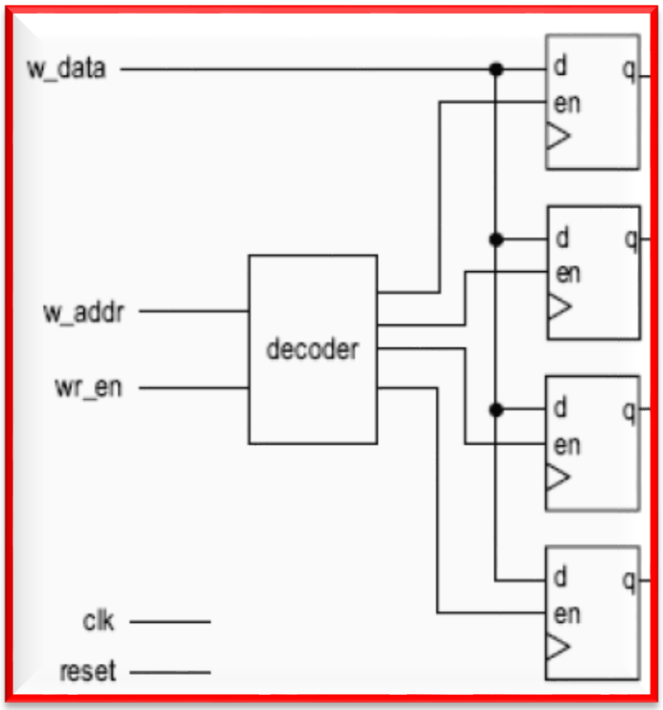
Enable logic for register




```
process(array_reg, en, w_data)
begin
    array_next(3) <= array_reg(3);
    array_next(2) <= array_reg(2);
    array_next(1) <= array_reg(1);
    array_next(0) <= array_reg(0);
    if en(3)='1' then
        array_next(3) <= w_data;
    end if;
    if en(2)='1' then
        array_next(2) <= w_data;
    end if;
    if en(1)='1' then
        array_next(1) <= w_data;
    end if;
    if en(0)='1' then
        array_next(0) <= w_data;
    end if;
end process;
```

VHDL: a parameterized 2^W-by-B register file

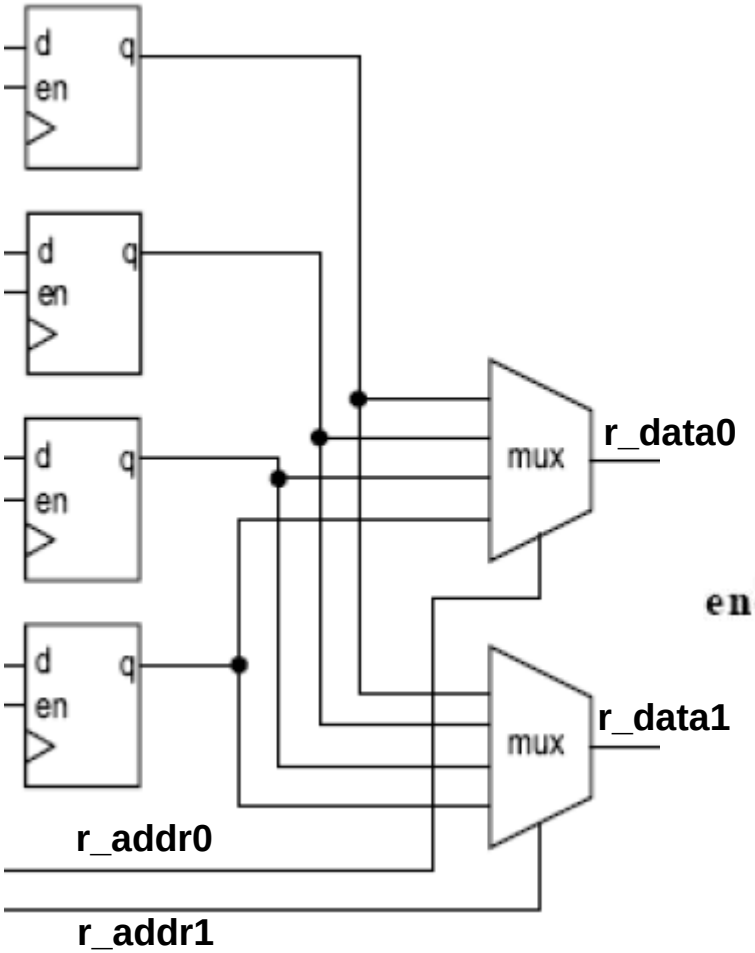
Enable logic for register (Cont.)




```
process(wr_en,w_addr)
begin
    if (wr_en='0') then
        en <= (others=>'0');
    else
        case w_addr  is
            when "00" =>    en <= "0001";
            when "01" =>    en <= "0010";
            when "10" =>    en <= "0100";
            when others => en <= "1000";
        end case;
    end if;
end process;
```

VHDL: a parameterized 2^W-by-B register file

Read Multiplexing

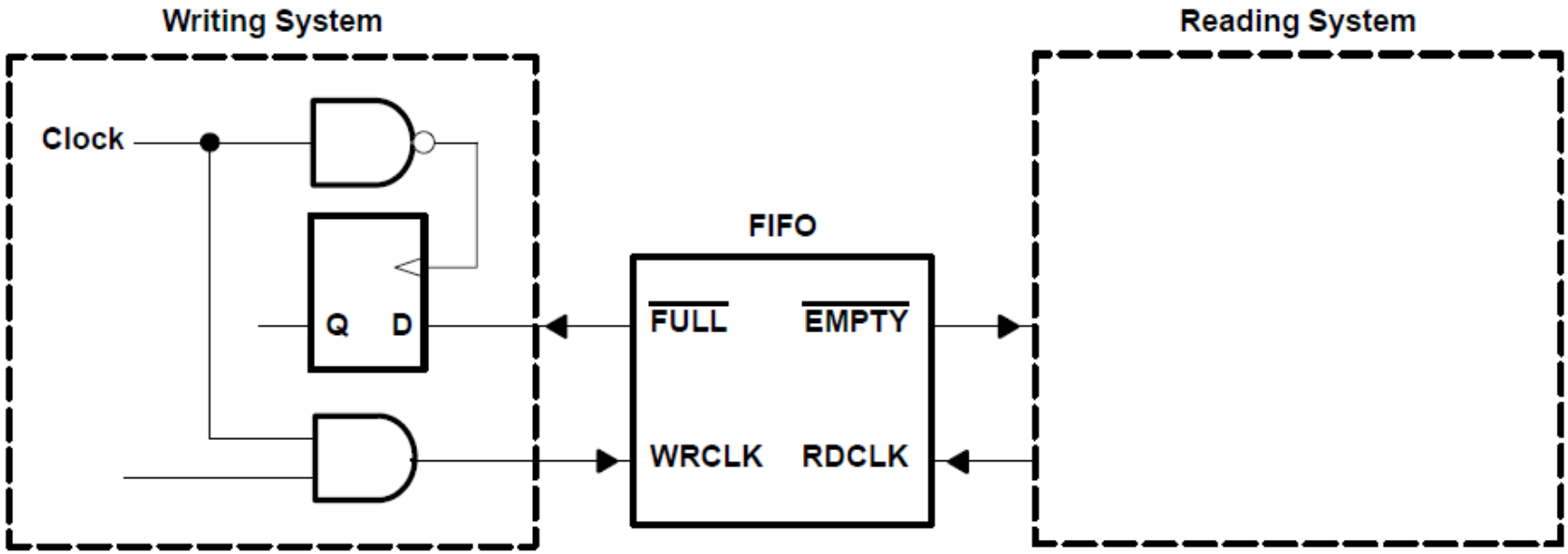


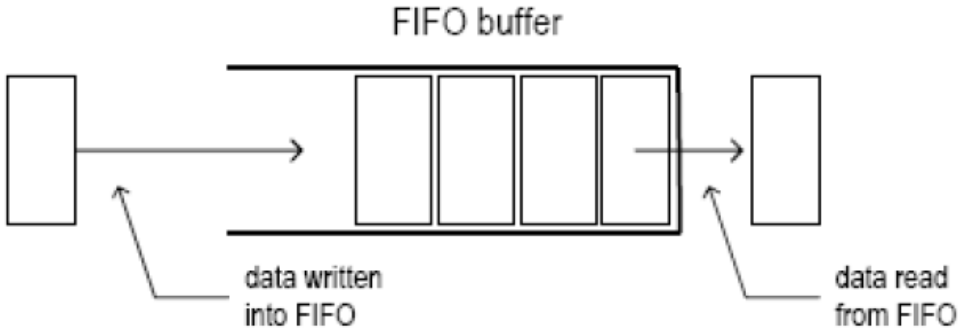

```
with r_addr0 select
   r_data0 <=  array_reg(0) when "00",
               array_reg(1) when "01",
               array_reg(2) when "10",
               array_reg(3) when others;
with r_addr1 select
   r_data1 <=  array_reg(0) when "00",
               array_reg(1) when "01",
               array_reg(2) when "10",
               array_reg(3) when others;
end no_loop_arch;
```

Storage Examples 2

FIFO (first in first out) Buffer

- "Elastic" storage between two subsystems



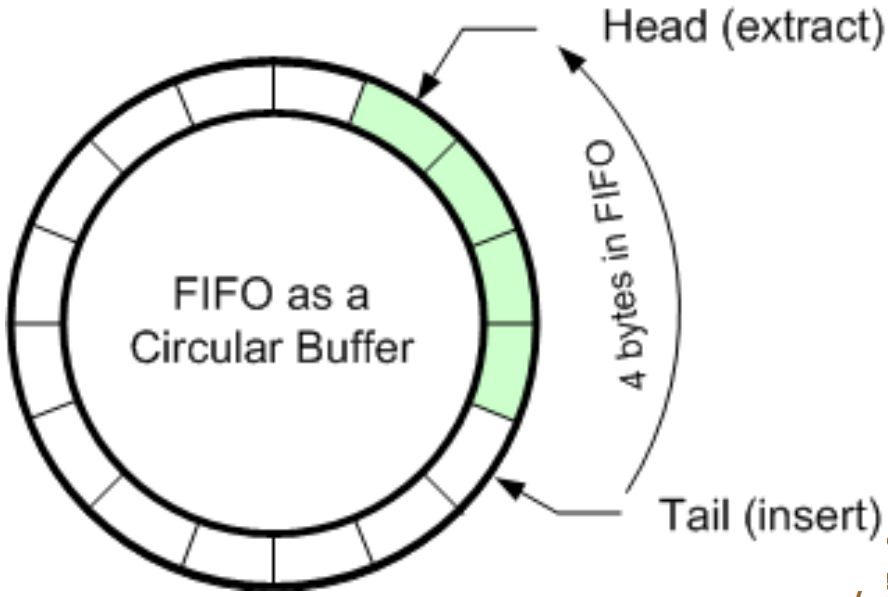


Circular FIFO

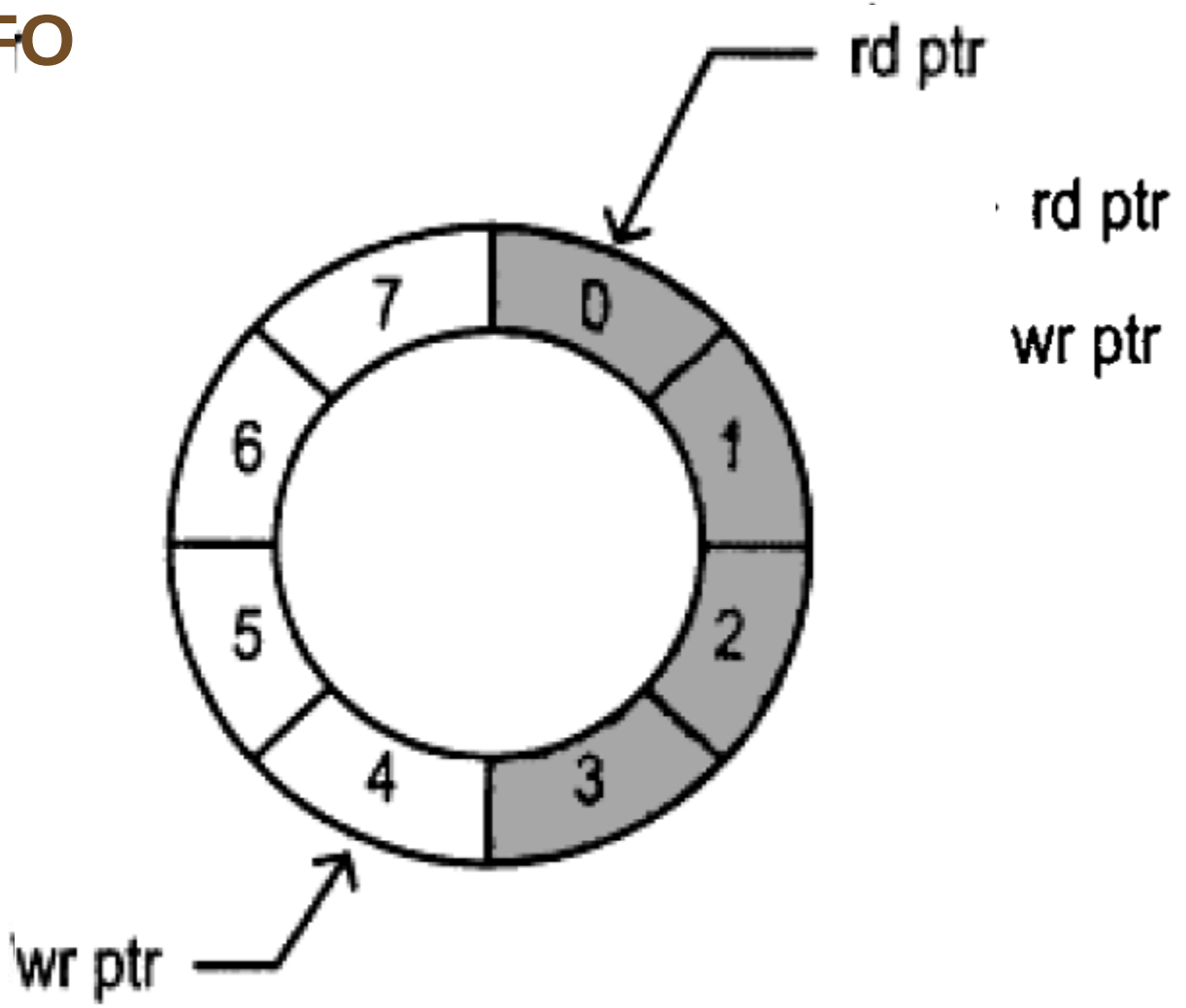
How to Implement a FIFO?

- Circular queue implementation
- Use two pointers and a “generic storage”
  - ***Write pointer*: point to the empty slot before the *head* of the queue**
  - ***Read pointer*: point to the *tail* of the queue**

"First in? First out!"



Circular FIFO

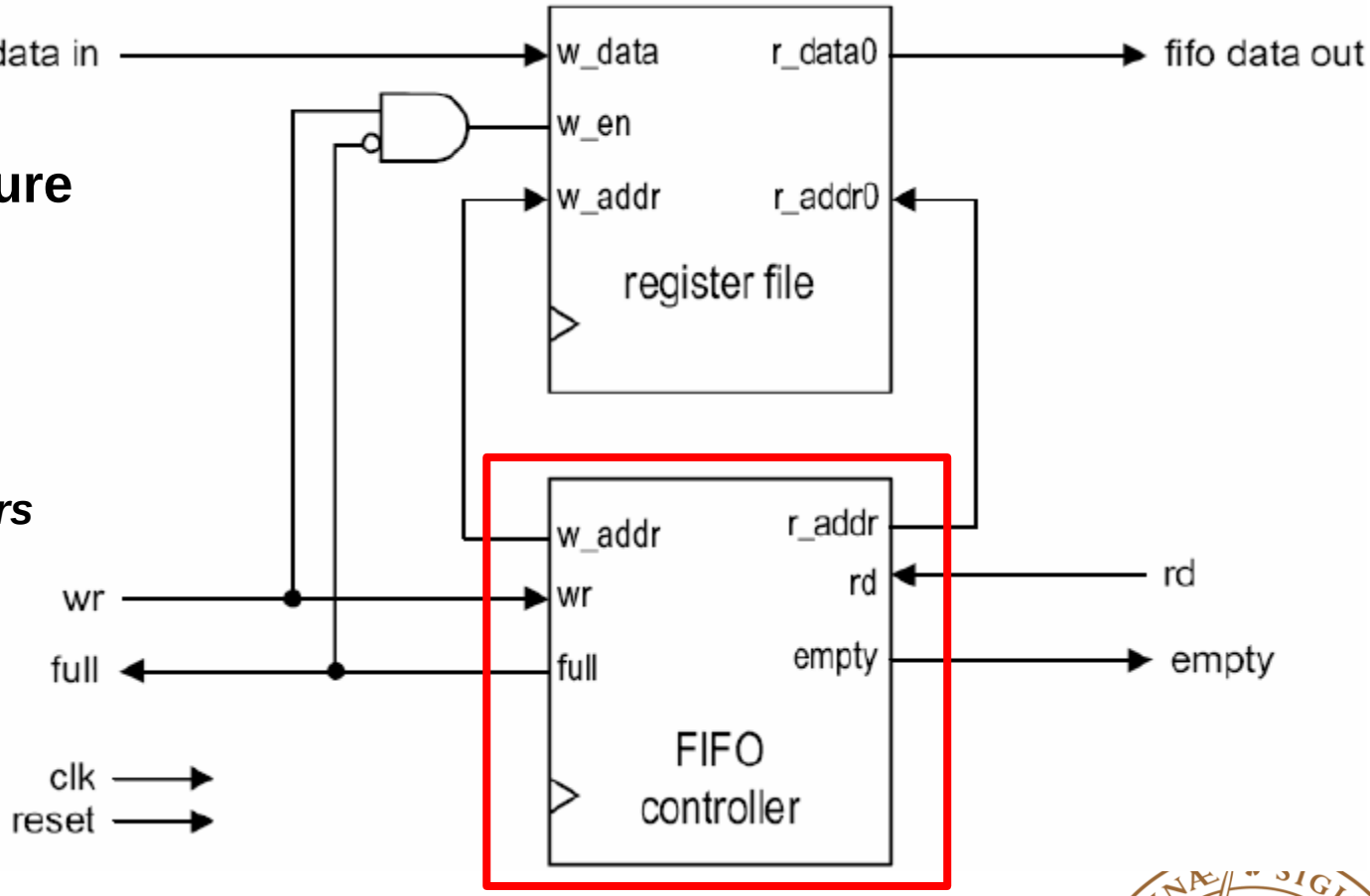


(c). 4 more writes

FIFO Implementation

- **Overall Architecture**
  - Storage Elements
    - ***Reg. file***
  - **FIFO Controller**
    - ***Read and write pointers: 2 counters***
    - ***Status circuit:***

      full, empty



FIFO Implementation: Controller

- **Augmented binary counter:**
  - Increase the counter by 1 bits
  - Use LSBs for as register address
  - Use **MSB** to distinguish full or empty

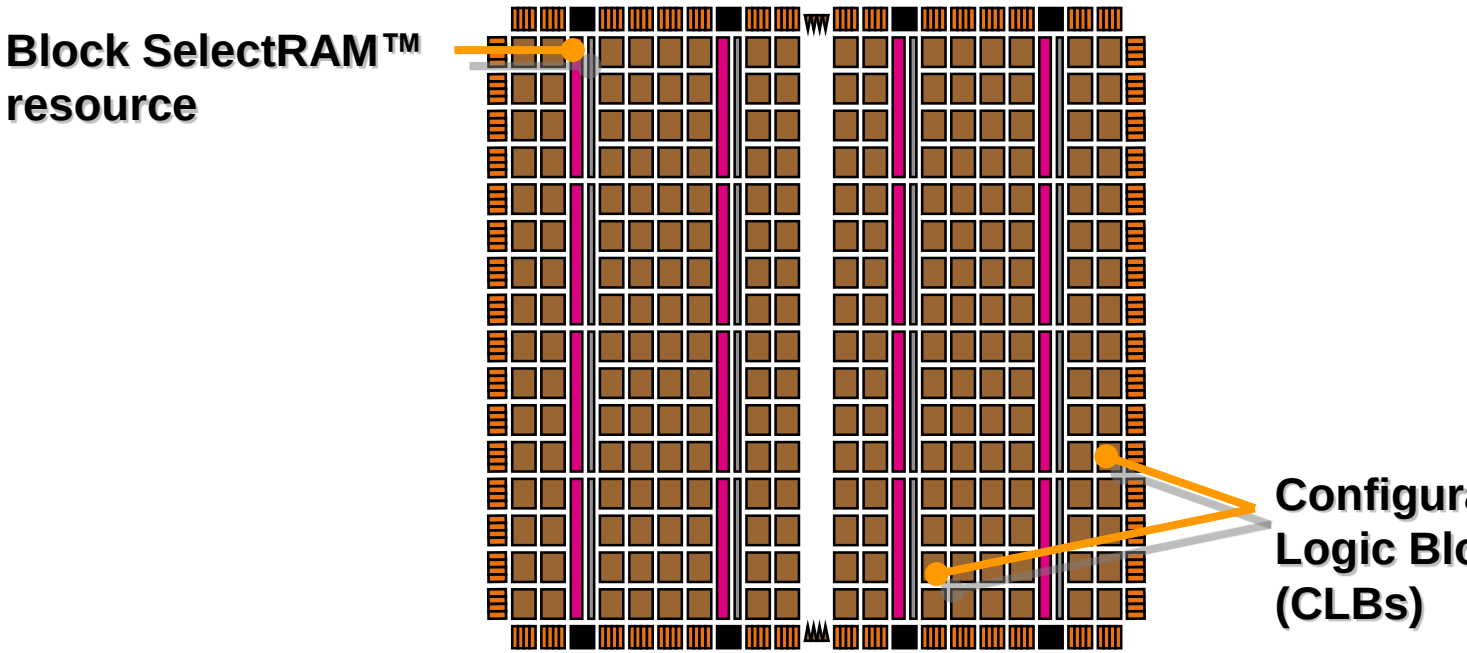
Write pointer	Read pointer	Operation	Status
0 000	0 000	initialization	empty
0 111	0 000	after 7 writes	
1 000	0 000	after 1 write	full
1 000	0 100	after 4 reads	
1 100	0 100	after 4 writes	full
1 100	1 011	after 7 reads	
1 100	1 100	after 1 read	empty
0 011	1 100	after 7 writes	
0 100	1 100	after 1 write	full
0 100	0 100	after 8 reads	empty

Outline

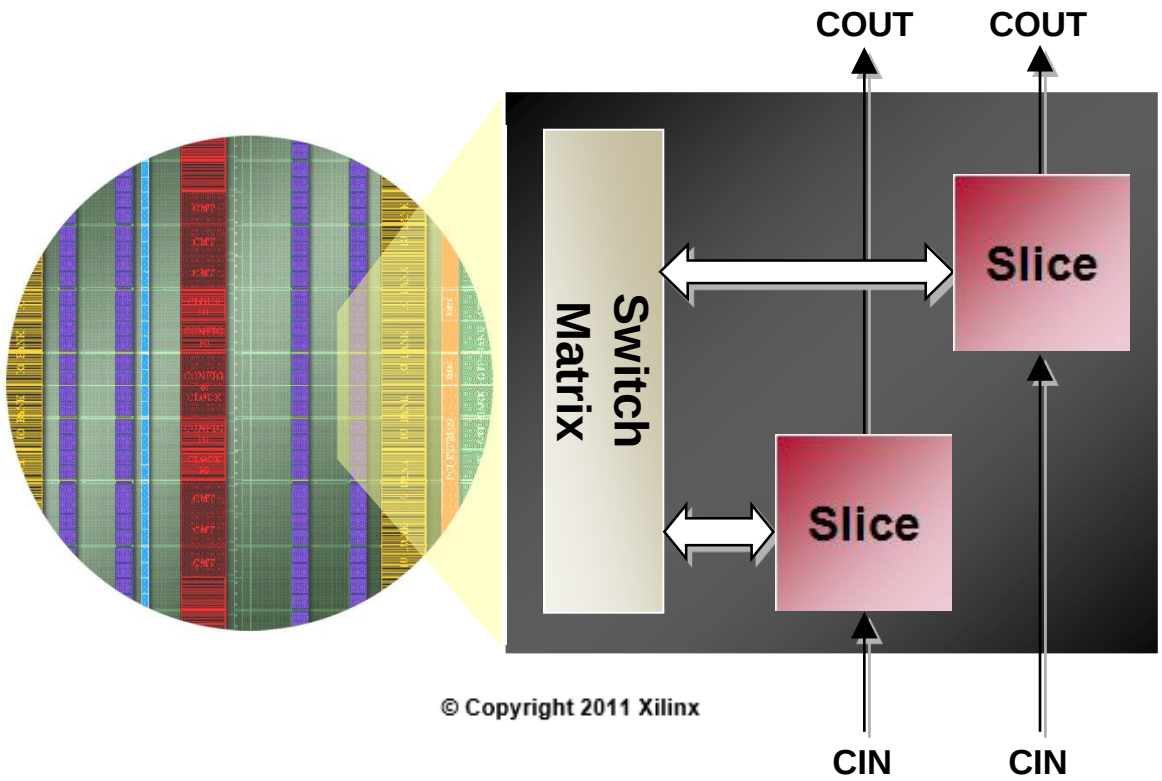
- Overview of Memory
  - Application, history, trend
  - Different memory type
  - Overall architecture
- Registers as Storage Element
  - Register File
  - FIFO
- **Xilinx Storage Elements**
- Memory Generator

Storage Components in a Xilinx Device

- **Distributed RAM**
  - Fast, localized
  - ideal for small data buffers, FIFOs, or register files
- **Block RAM**
  - For applications requiring large, on-chip memories



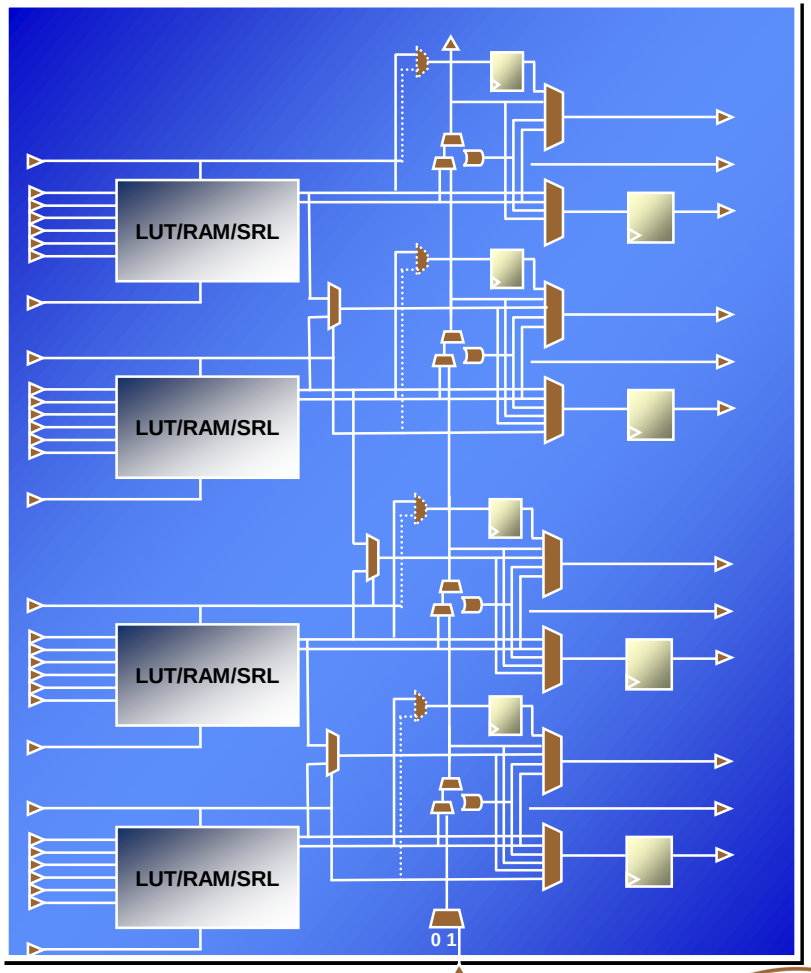
Xilinx CLB



- CLB contains two slices
- Connected to switch matrix for routing to other FPGA resources

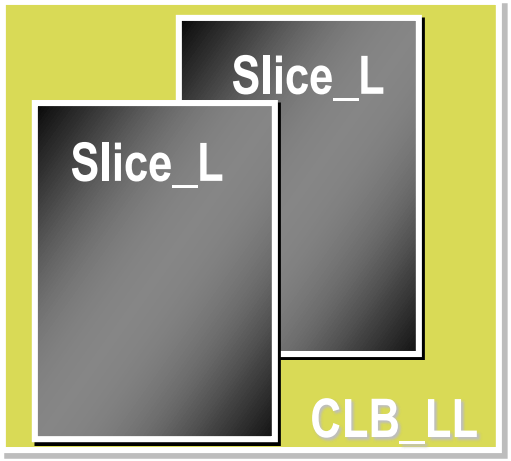
Xilinx Slice

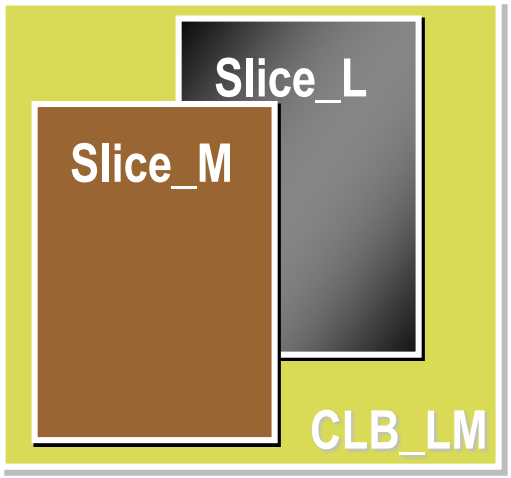
- Four six-input Look Up Tables (LUT)
- Wide multiplexers
- Carry chain
- Four flip-flop/latches
- Four additional flip-flops



Two type of slice

- **Two types of slices**
- **SLICEM: Full slice**
  - LUT can be used for logic and memory/SRL
  - Has wide multiplexers and carry chain
- **SLICEL: Logic and arithmetic only**
  - LUT can only be used for logic (not memory)
  - Has wide multiplexers and carry chain





SLICEM Used as Distributed Memory

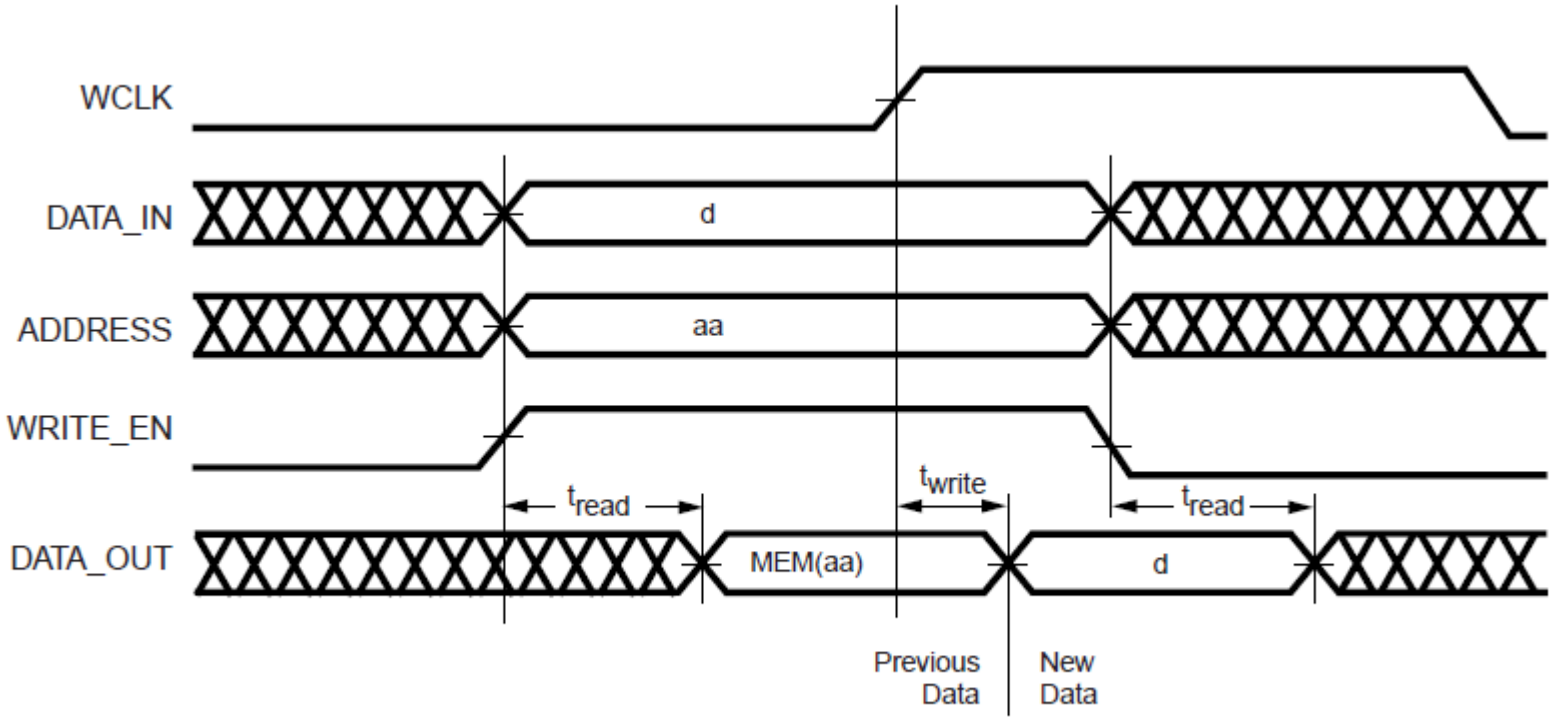
- **Various configurations**
  - Single port
    - One LUT6 = 64x1 or 32x2 RAM
    - Cascadable up to 256x1 RAM
  - Dual port (D)
    - 1 read / write port + 1 read-only port
  - Simple dual port (SDP)
    - 1 write-only port + 1 read-only port
  - Quad-port (Q)
    - 1 read / write port + 3 read-only ports
- **Synchronous write**
- **Asynchronous read**
  - Accompanying flip-flops can be used to create synchronous read

Single Port	Dual Port	Simple Dual Port	Quad Port
32x2	32x2**D**	32x6**SDP**	32x2**Q**
32x4	32x4**D**	64x3**SDP**	64x1**Q**
32x6	64x1**D**		
32x8	64x2**D**		
64x1	128x1**D**		
64x2			
64x3			
64x4			
128x1			
128x2			
256x1			

Spartan-3 Distributed Memory

Timing

- Synchronous write
- Asynchronous read



x464_02_070303

Artix-7 Block Memory

Logic Fabric
LUT-6 CLB

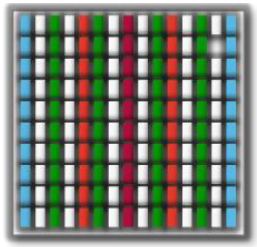
Precise, Low Jitter Clocking
MMCMs

On-Chip Memory
36Kbit/18Kbit Block RAM

Enhanced Connectivity
PCIe® Interface Blocks

DSP Engines
DSP48E1 Slices

Hi-perf. Parallel I/O Connectivity
SelectIO™ Technology

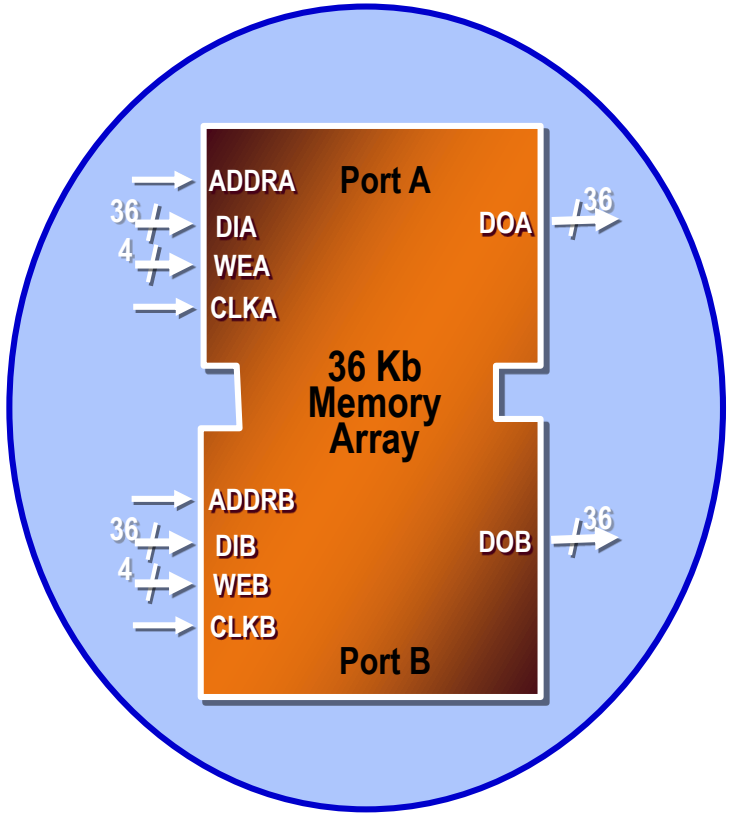


Artix™-7 FPGA

- **Most efficient memory implementation**
  - Dedicated blocks of memory
  - 4,860 Kbits of fast block RAM for Artix-7 100T
- **Builds both single and true dual-port RAMs**
- **Synchronous write and read (different from distributed RAM)**

Artix-7 Block Memory

- **36K/18K block RAM**
- **Configurations**
  - 32k x 1 to 512 x 72 in one 36K block
  - Simple dual-port and true dual-port configurations
  - Built-in FIFO logic
  - 64-bit error correction coding per 36K block
  - Adjacent blocks combine to 64K x 1 without extra logic



Block RAM Ports

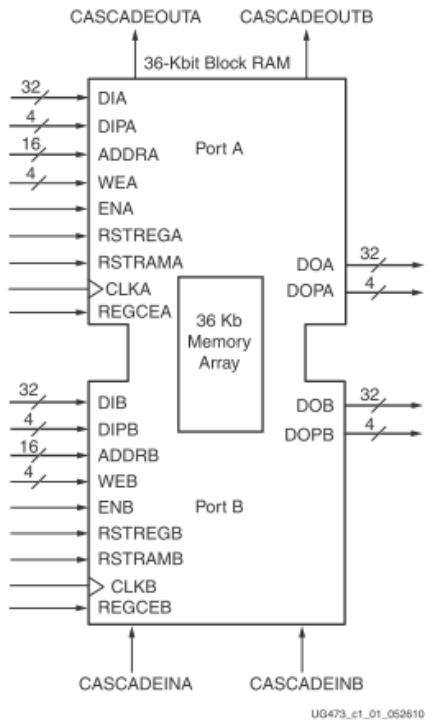


Table 1-3: **True Dual-Port Functions and Descriptions**

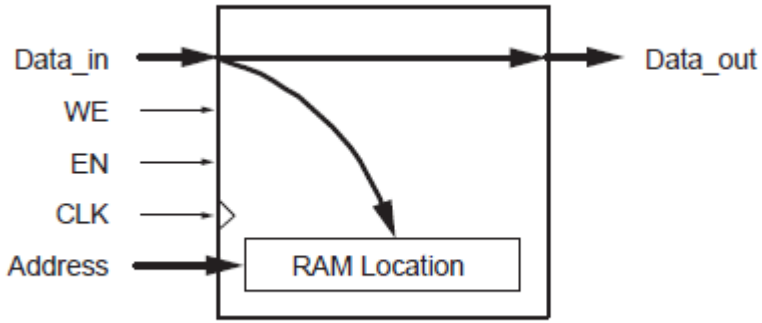
Port Function	Description
DI[A | B]	Data input bus.
DIP[A | B](1)	Data input parity bus. Can be used for additional data inputs.
ADDR[A | B]	Address bus.
WE[A | B]	Byte-wide write enable.
EN[A | B]	When inactive no data is written to the block RAM and the output bus remains in its previous state.
RSTREG[A | B]	Synchronous Set/Reset the output registers (DO_REG = 1). The RSTREG_PRIORITY attribute determines the priority over REGCE.
RSTRAM[A | B]	Synchronous Set/Reset the output data latches.
CLK[A | B]	Clock input.
DO[A | B]	Data output bus.
DOP[A | B](1)	Data output parity bus. Can be used for additional data outputs.
REGCE[A | B]	Output Register clock enable.
CASCADEIN[A | B]	Cascade input for 64K x 1 mode.
CASCADEOUT[A | B]	Cascade output for 64K x 1 mode.

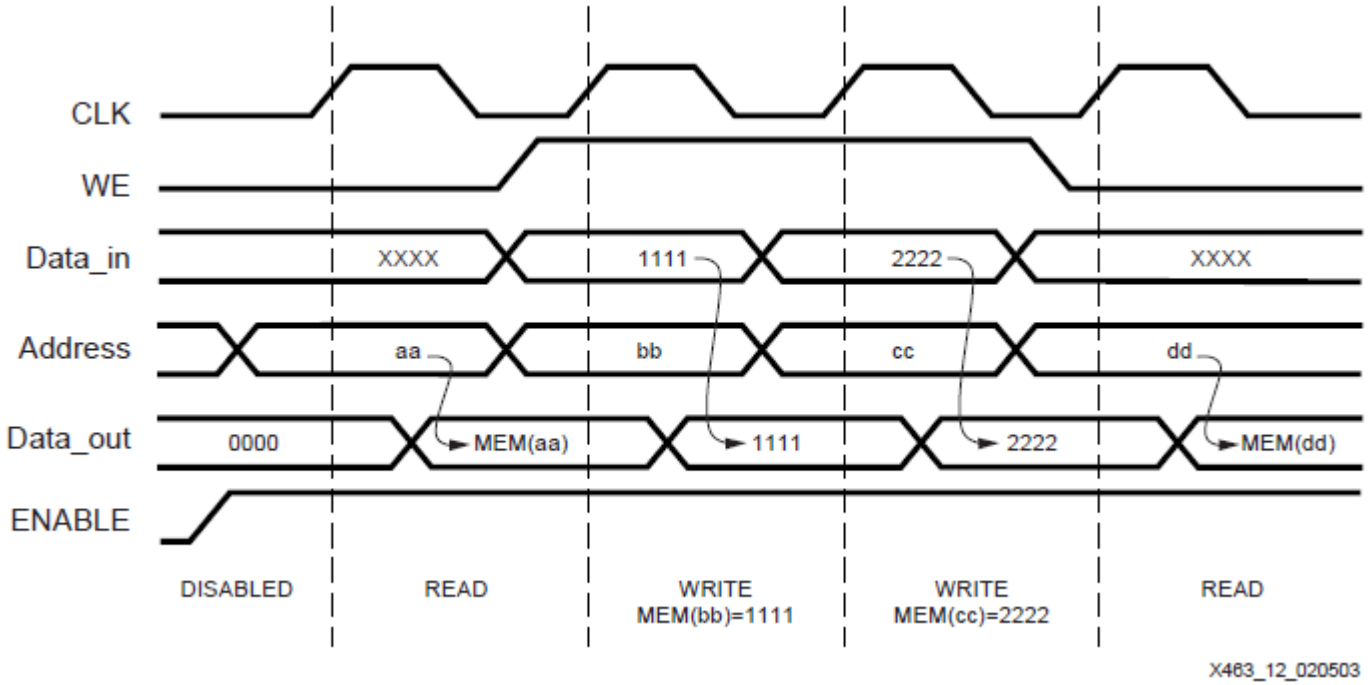
- **$DI_{A,B}$: the data path width at ports A,B.**
- **$ADDR_{A,B}$: the address bus width at ports A, B**
- **The control signals CLK, WE, EN**
- **Reset signal does NOT affect memory cells**

Block RAM: Operation Modes

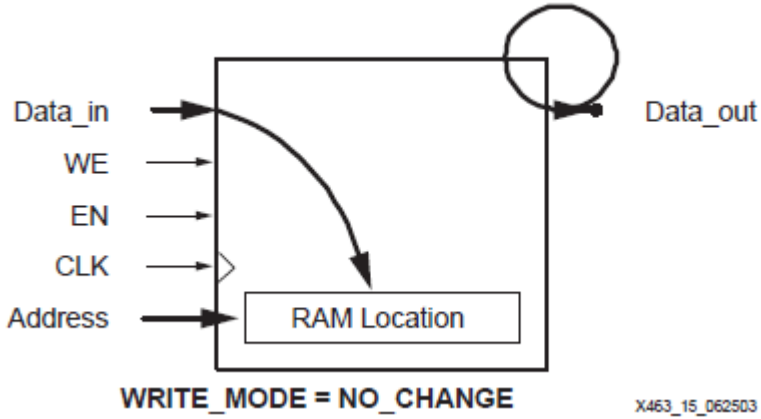
Write Mode	Effect on Same Port	Effect on Opposite Port (dual-port mode only, same address)
WRITE_FIRST Read After Write (Default)	Data on DI, DIP inputs written into specified RAM location and simultaneously appears on DO, DOP outputs.	Invalidates data on DO, DOP outputs.
READ_FIRST Read Before Write (Recommended)	Data from specified RAM location appears on DO, DOP outputs. Data on DI, DIP inputs written into specified location.	Data from specified RAM location appears on DO, DOP outputs.
NO_CHANGE No Read on Write	Data on DO, DOP outputs remains unchanged. Data on DI, DIP inputs written into specified location.	Invalidates data on DO, DOP outputs.

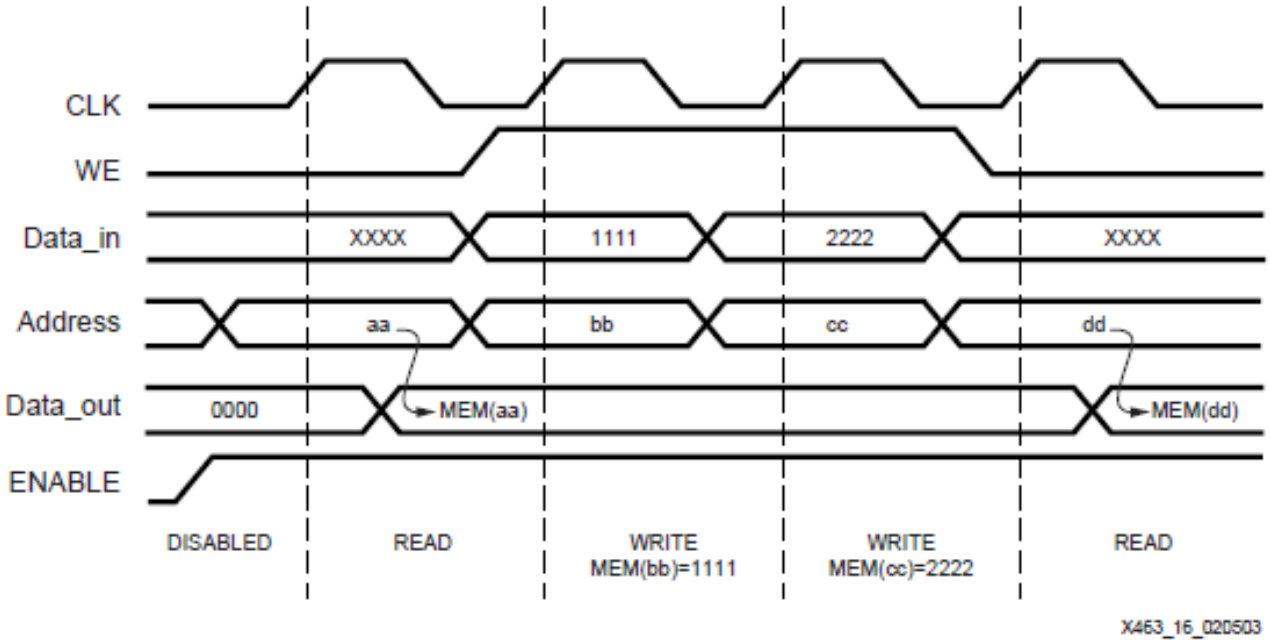
Block RAM: WRITE_FIRST



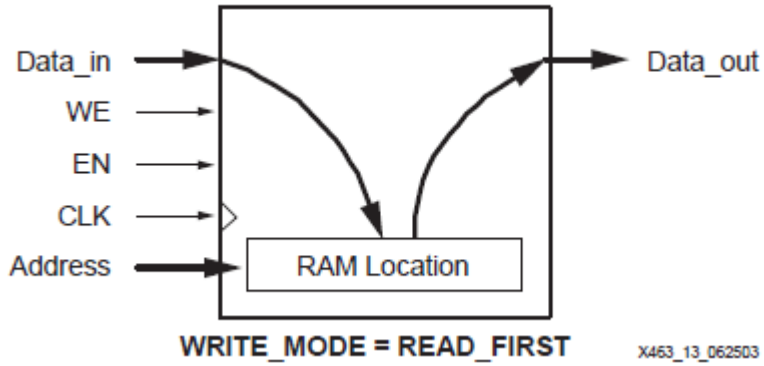


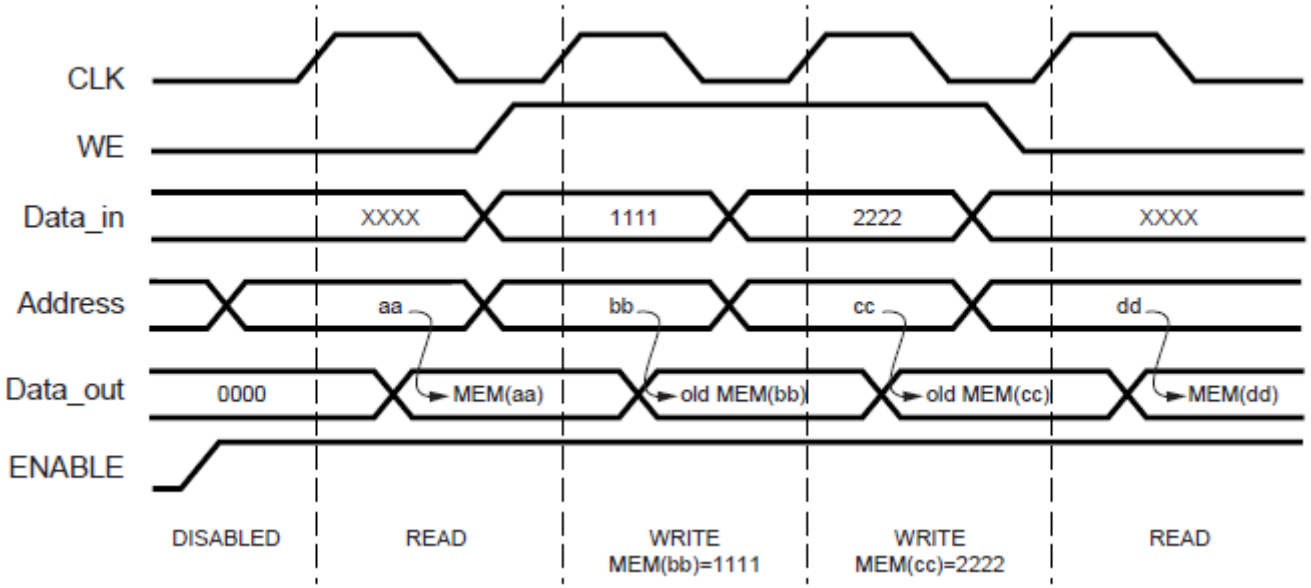
Block RAM: NO_CHANGE





Block RAM: READ_FIRST (Recomm.)





Reading Advice

- RTL Hardware Design Using VHDL: P276-P292
- UG437 7 Series FPGAs Memory Resources
- UG901 Vivado Design Suite User Guide Synthesis