Hands on with FPGA's: Module 5

Venkat Rangan

Questions on Module 4

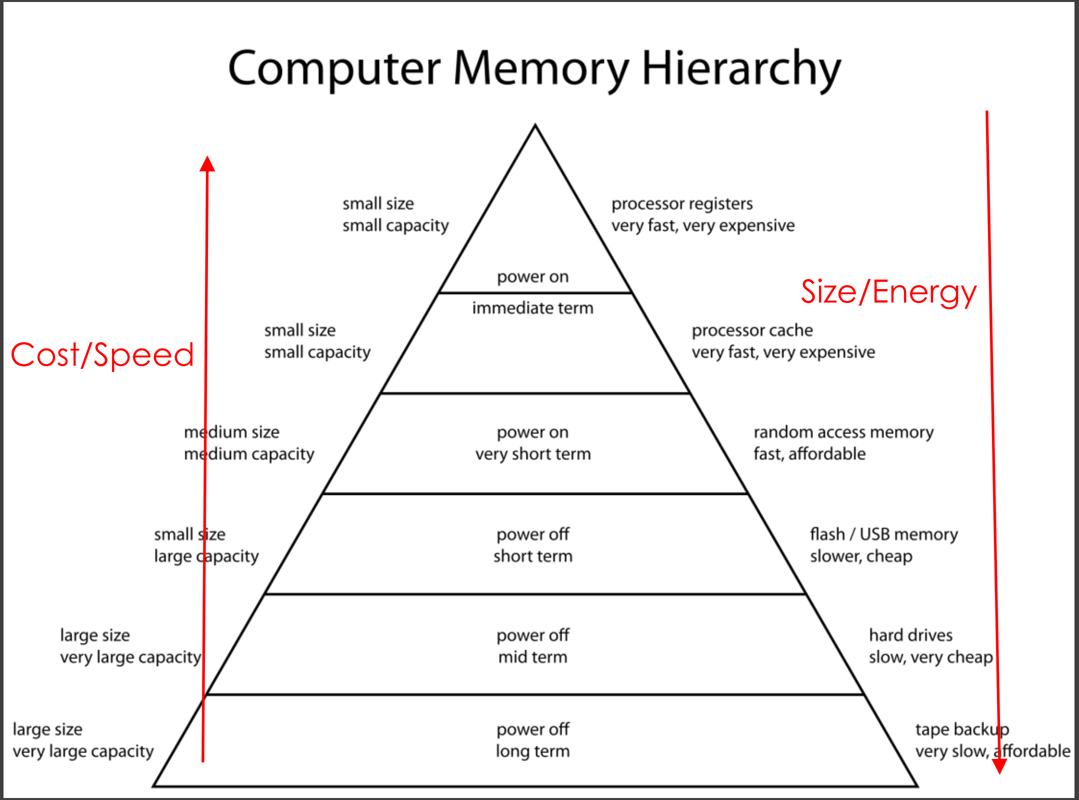
- State Machines
- Traffic light problem

Topics

- Pre-class: Open floor for questions
  - Module 4
- Memories: ROM, RAM
  - Memory hierarchies
- Playing with Memories
- Structural vs Behavioral coding styles
- Open discussion

Memory Hierarchy

- Non-volatile: ROM
- Volatile: RAM
  - Registers: very fast, small
  - Compact bank of registers (register arrays)
    - Found in very few FPGA's
  - Look Up Table RAM
    - Very small, fast RAM
  - Block RAM
    - Usually ~10's of Kb
  - Large RAM
    - Usually ~100's of kB
  - Off Chip: SRAM/DRAM
    - Large, slow, high energy (1000x!)

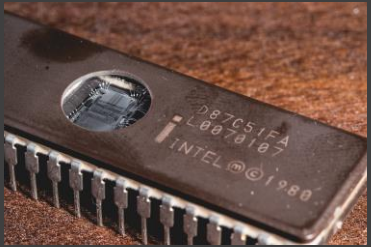


By ComputerMemoryHierarchy.png: User:Danlash at en.wikipedia.org - ComputerMemoryHierarchy.png, Public Domain, https://commons.wikimedia.org/w/index.php?curid=9439275

Memories: Read Only Memory

- Store programs, data
- Can range from a small look up table through a full program

Game cartridges



UV EEPROM

RAM Access

- Byte access: more flexibility
- Variable access widths
- Single Port RAM
  - Single clock domain
  - Can do a single write/read per clock cycle
- Pseudo Dual Port RAM
  - Independent read/write ports
  - Can have independent clock domains
- True Dual Port RAM
  - Each port and read/write independently

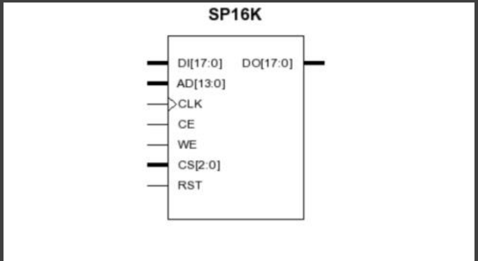


Figure 4.2. Single Port RAM Primitive for Nexus Platform Devices

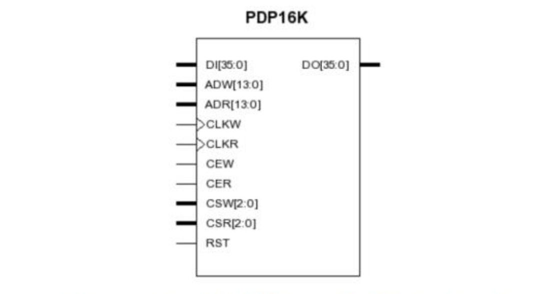
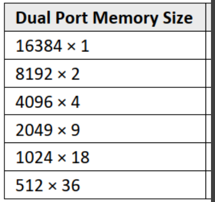


Figure 4.10. Pseudo-Dual Port RAM Primitive for Nexus Platform Devices



Dual Port Memory Size
16384 × 1
8192 × 2
4096 × 4
2049 × 9
1024 × 18
512 × 36

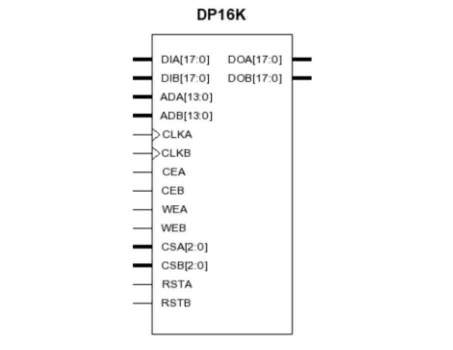


Figure 4.6. True Dual Port RAM Primitive for Nexus Platform Devices

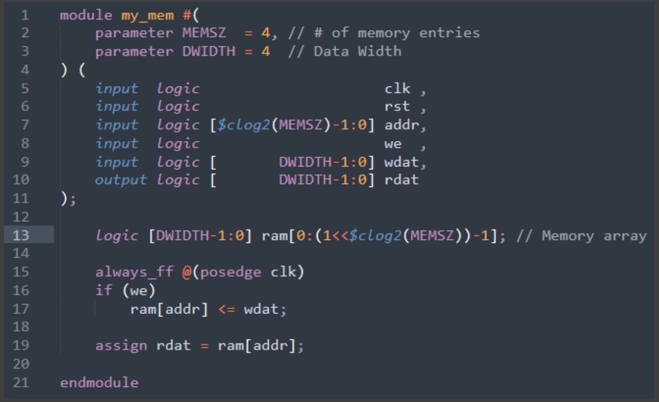
Memories in Digital HW: System Level decisions

- What kind of memory to use?
  - How fast?
  - How much data?

- How wide should it be?
  - How much data to process?

Playing with memories

- Lattice Radiant/Synplicity for Synthesis
- What happens as we increase memory size?



```
module my_mem #(
    parameter MEMSZ  = 4, // # of memory entries
    parameter DWIDTH = 4  // Data Width
) (
    input  logic                       clk ,
    input  logic                       rst ,
    input  logic [$clog2(MEMSZ)-1:0] addr,
    input  logic                       we   ,
    input  logic [        DWIDTH-1:0] wdat,
    output logic [        DWIDTH-1:0] rdat
);

    logic [DWIDTH-1:0] ram[0:(1<<$clog2(MEMSZ))-1]; // Memory array

    always_ff @(posedge clk)
    if (we)
        ram[addr] <= wdat;

    assign rdat = ram[addr];

endmodule
```

Experiment #	MEMSZ	DWIDTH	Number and Type of Memory
1	4	4	
2	4	8	
3	1024	8	
4	1024	32	

Understanding an FPGA: Lattice iCE40UP)

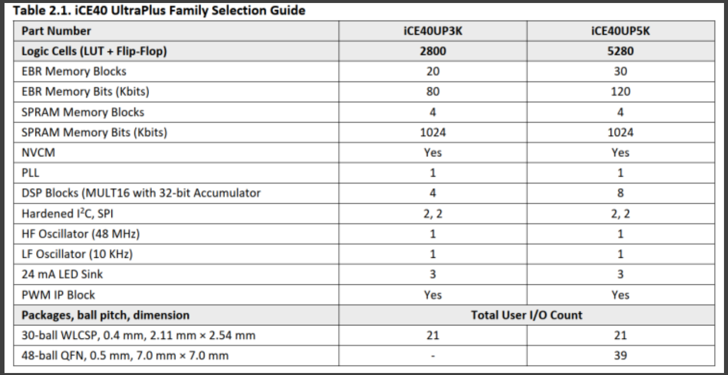


Table 2.1. iCE40 UltraPlus Family Selection Guide

Part Number	iCE40UP3K	iCE40UP5K
Logic Cells (LUT + Flip-Flop)	**2800**	**5280**
EBR Memory Blocks	20	30
EBR Memory Bits (Kbits)	80	120
SPRAM Memory Blocks	4	4
SPRAM Memory Bits (Kbits)	1024	1024
NVCM	Yes	Yes
PLL	1	1
DSP Blocks (MULT16 with 32-bit Accumulator	4	8
Hardened I²C, SPI	2, 2	2, 2
HF Oscillator (48 MHz)	1	1
LF Oscillator (10 KHz)	1	1
24 mA LED Sink	3	3
PWM IP Block	Yes	Yes
Packages, ball pitch, dimension	**Total User I/O Count**	
30-ball WLCSP, 0.4 mm, 2.11 mm × 2.54 mm	21	21
48-ball QFN, 0.5 mm, 7.0 mm × 7.0 mm	-	39

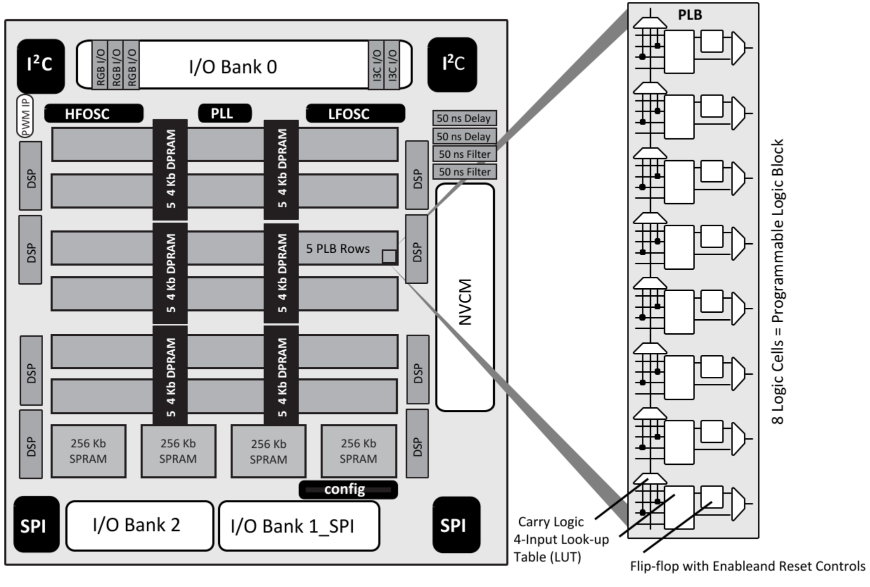


Figure 3.1. iCE40UP5K Device, Top View

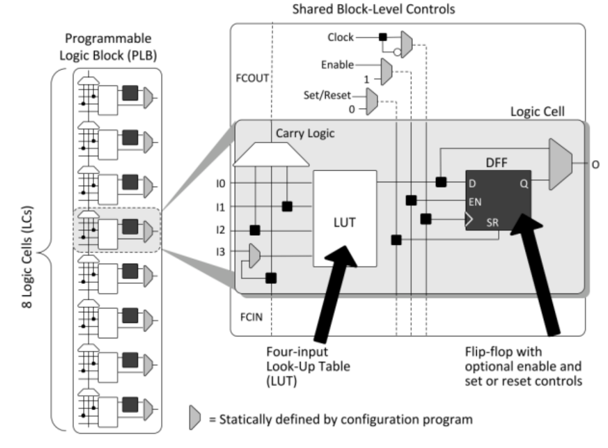

Shared Block-Level Controls
Programmable Logic Block (PLB)
Clock
Enable
1
Set/Reset
0
FCOUT
Logic Cell
Carry Logic
DFF
I0
I1
I2
I3
LUT
D
Q
EN
SR
O
8 Logic Cells (LCs)
FCIN
Four-input Look-Up Table (LUT)
Flip-flop with optional enable and set or reset controls
= Statically defined by configuration program

Figure 3.2. PLB Block Diagram

Coding Style

- Structural coding
    - Specific FPGA features
    - Found in low level netlists
    - Excellent for timing
    - Brittle code

```
module structural (
             input  logic a,
             input  logic b,
             output logic y
);

             and a1 (y,a,b); // AND gate instantiation

endmodule
```

- Behavioral code
    - Easy to understand
    - Requires logic synthesis
    - Sometimes have to use for IO cells, specialized blocks

```
module behavioral (
             input  logic a,
             input  logic b,
             output logic y
);

             assign y = a & b; // Behavioral description

endmodule
```

Module 5: Memories

- Challenge
    - Code up a digital clock

Open Discussion