Memories

- Introduction
  - Why do we need memory in an FPGA Device?
- Topics
  - Types of FPGA Memories
  - Available Resources
  - Realizing memories in an HDL design
  - Applications
  - Examples
  - Extra: Memory Controller Block

Memories

- Types of FPGA memories
  - Registers (Flip-Flops)
    - single bit Random Access Memory (RAM)
  - Look-Up-Tables
    - Read Only Memory (ROM)
  - Distributed RAM
    - Added LUT functionality
  - Shift Registers
    - Added LUT functionality
  - Block RAM
    - Hard IP

Resources: *where find more info*

- Xilinx User Guides
  - Memory Capabilities
    - UG384: SP6 Configurable Logic Block
      - Flip-Flops and Look-Up-Table
    - UG383: SP6 Block RAM
    - UG388: SP6 Memory Controller Block
  - Port Descriptions and Templates
    - UG615: SP6 Libraries Guide
- Example designs and tutorials
  - Eval Board Tutorial

Placing Memories in HDL

- Two Basic Methods
  - Instantiation Template from a Port Description
    - Libraries Guide
      - What are wrappers?
    - Sub-Module
    - IP Cores
    - Tools:
      - Core Generator
      - Memory Interface Generator (MIG)
  - Inference for Functional Code
- CORE or CODE?

THE Verilog MEMORY STATEMENT

- Remember the Register

  reg [7:0] mem_1x8bit;

  - this statement alone represents a 1x8 bit declaration of memory
  - how do you read and write to this memory?

```
always@(posedge clock) begin
  ...
  mem_1x8bit <= data_in;
  ...
  data_out <= mem_1x8bit;
  ...
end
```

  - can you write to individual bits?

THE Verilog MEMORY STATEMENT

- Implementing multiple address lines

  reg [7:0] mem_8x8bit [7:0];

  - this statement alone represents a 8x8 bit declaration of a memory array
  - how do you read and write to this memory?

```
always@(posedge clock) begin
  ...
  mem_8x8bit[addr] <= data_in;
  ...
  data_out <= mem_8x8bit[addr];
  ...
end
```

  - what is the width of addr?
  - can you write to individual bits?

THE Verilog MEMORY STATEMENT

- If you write your code correctly, XST will automatically use block memory for large arrays.
  - How large? Test, Simulation Report
  - Code Investigation

```
reg [15:0] memory [1023:0];
always@(posedge clock) begin
  ...
  memory[1023] <= data_in1;
  memory[1022] <= data_in2;
  memory[1021] <= data_in3;
  ...
end
```

  - If the functionality of the code is not contrained to the capabilities of the Hard Memory Block, then it cannot be *directly* synthesized.

Memory Applications

- Arbitrary Waveform Generation
  - Store a signal in consecutive address and playback waveform
  - Block Diagram
  - Direct Digital Synthesis

Memory Applications

- Processor
  - Instruction and Data
  - Dual-Port
  - Block Diagram

Memory Applications

- FIFOs
  - Data Buffering
  - Transitioning between clock domains
  - Block Diagram