Vivado从此开始（To Learn Vivado From Here）

本书围绕Vivado四大主题

- 设计流程
- 时序约束
- 时序分析
- Tcl脚本的使用

作者：高亚军（Xilinx战略应用高级工程师）

- 2012年2月，出版《基于FPGA的数字信号处理（第1版）》
- 2012年9月，发布网络视频课程《Vivado入门与提高》
- 2015年7月，出版《基于FPGA的数字信号处理（第2版）》
- 2016年7月，发布网络视频课程《跟Xilinx SAE学HLS》

- 内容翔实全面：涵盖Vivado所有基本功能
- 讲解深入浅出：结合大量案例，帮助读者加强对基本概念的理解
- 描述图文并茂：给出具体操作步骤，易于快速动手实践

RTL Coding Style
Part 2

Lauren Gao

Know What You Infer

- **For larger than 4-bit addition, subtraction and add-sub**
  - Carry chain + one LUT per 2-bit addition
  - 8-bit + 8-bit adder: 8 LUTs + associated carry chain
- **Ternary addition and without the use of a register in between**
  - One LUT per 3-bit addition
  - 8-bit + 8-bit + 8-bit adder: 8 LUTs + associated carry chain
- **In general, multiplication is targeted to DSP blocks**
  - Three levels of pipelining around it generates best setup, clock-to-out, and power characteristics

Know What You Infer

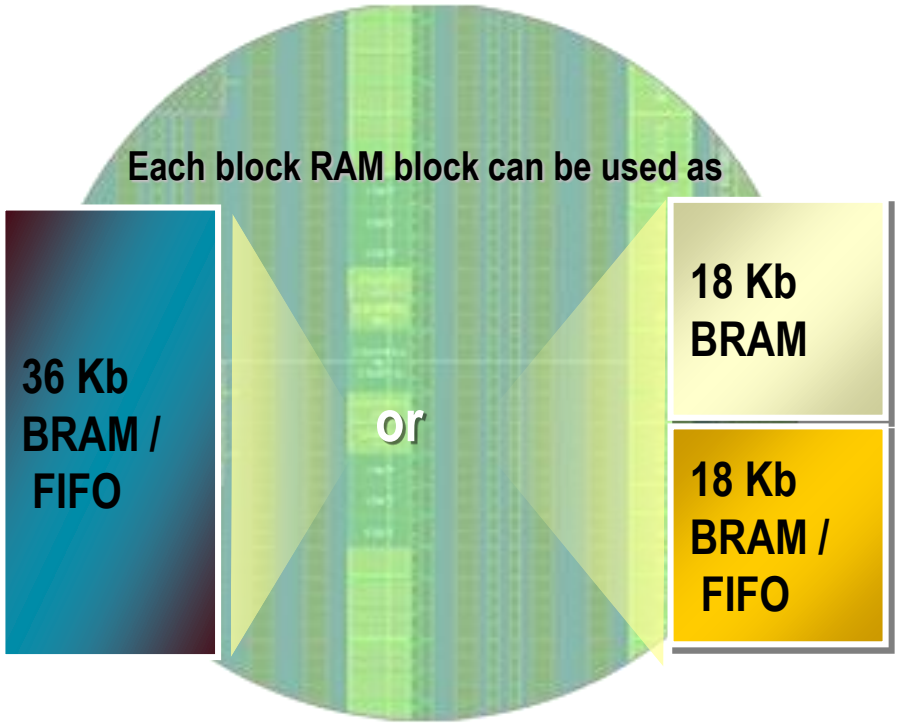
- **Shift registers or delay lines that do not require reset or multiple tap points are generally mapped into Shift Register LUTs or SRLs**
  - To best utilize SRLs, avoid using reset for those blocks
  - In 7-series FPGA, each LUT can delay serial data from 1 to 32 clock cycles
- **For conditional code resulting in standard MUX components**
  - 4-to-1 MUX: 1 LUT, one logic level
  - 8-to-1 MUX: 2 LUTs + 1 MUXF7, one logic level
  - 16-to-1 MUX: 4 LUTS + 1 MUXF7 + 1 MUXF8, one logic level

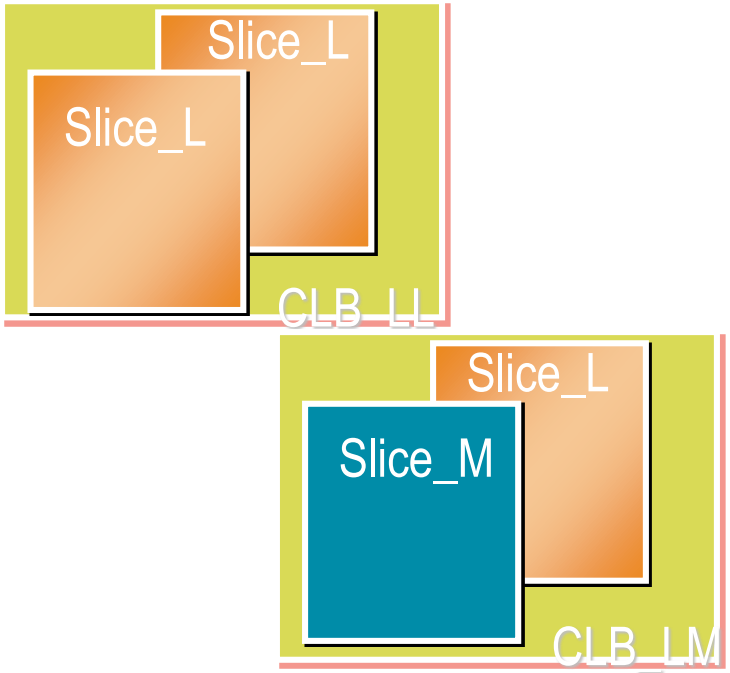
Performance Considerations When Implementing RAM

- Using Dedicated Blocks or Distributed RAMs
- Using the Output Pipeline Register
- Selecting the Proper Block RAM Write Mode

Using Dedicated Blocks or Distributed RAMs

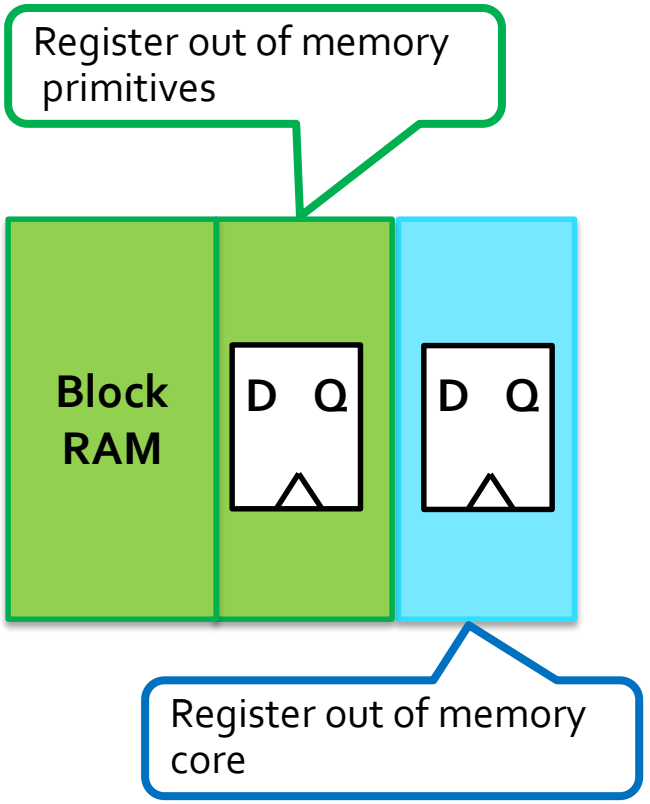
- RAMs may be implemented in either
  - the dedicated block RAM
  - Within LUTs using distributed RAM
- The First Choice Criterion: Required Depth
  - Memory arrays deeper than 256 are generally implemented in Block memory





Using the Output Pipeline Register

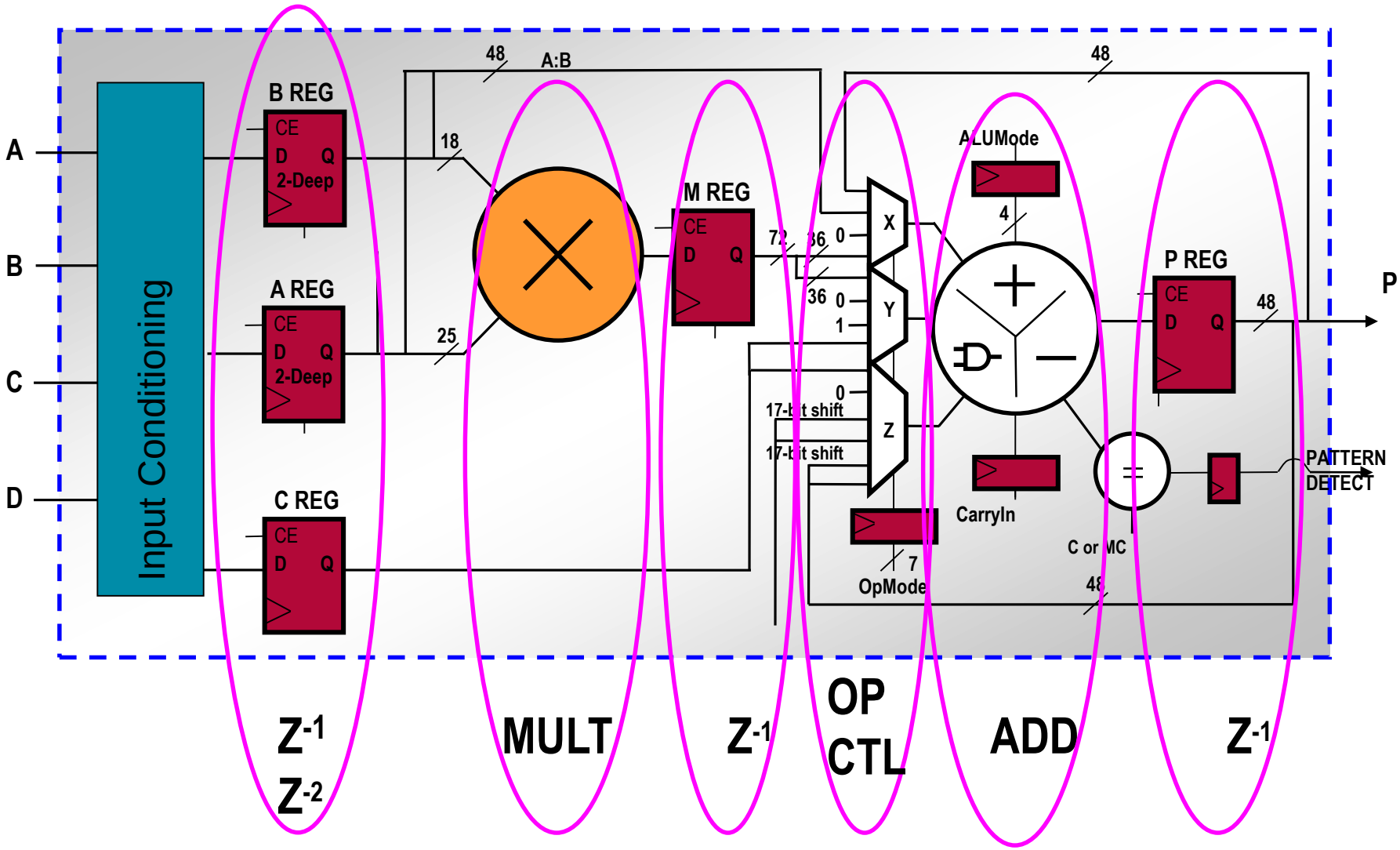
- **Using an output register is required for high performance designs**
  - It is recommended for all designs
  - This improves the clock to output timing of the block RAM
- **Having both registers has a total read latency of 3**
  - Determine early whether an extra clock cycle of latency during reads is tolerable
- **Using asynchronous reset impacts RAM inference, and should be avoided**



Selecting the Proper Block RAM Write Mode

- **Xilinx recommends the following guidelines for selecting the best write mode for a particular operation**
  - Consider Functionality First
    - If you must see the prior value in the block RAM during write, select READ_FIRST
    - If you want to read the new data being written to the block RAM use WRITE_FIRST
    - If you do not care about the data read during writes, then the next selection criteria has to do with memory collisions
  - Use NO_CHANGE Mode
    - In all other cases, Xilinx recommends NO_CHANGE mode. NO_CHANGE has the best power characteristics

DSP Slice Features

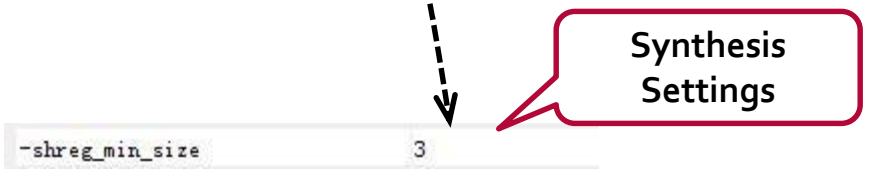


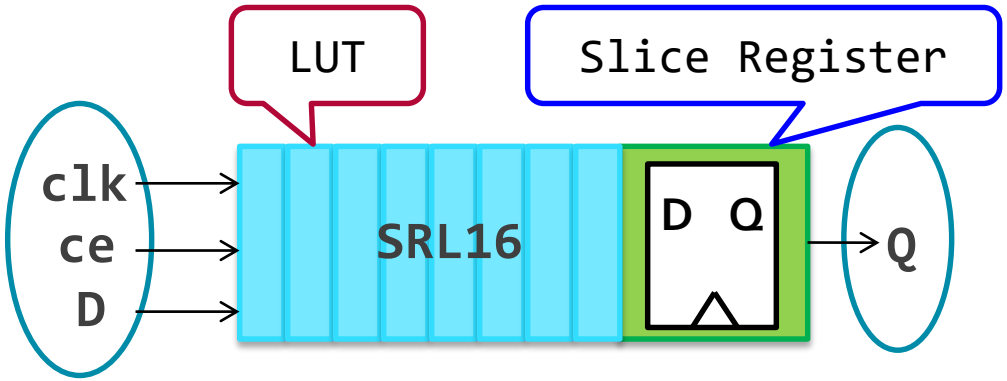
Coding for Proper DSP and Arithmetic Inference

- **The DSP blocks can perform many different function**
  - Multiplication, Addition and subtraction, Comparators, Counters, General logic
  - Fully pipeline the code intended to map into the DSP48
- **DSP48E1 slice registers contain only resets, and not sets**
  - Avoid asynchronous resets, since the DSP slice only supports synchronous reset operations
- **The DSP48E1 blocks use a signed arithmetic implementation**
  - Code using signed values in the HDL source to best match the resource capabilities
  - The bit precision for signed data is 18 bits by 25 bits
  - The bit precision for unsigned data is 17 bits by 24 bits
- **For Verilog code, data is considered unsigned unless otherwise declared in the code**

Coding Shift Registers and Delay Lines

- Xilinx FPGA devices contain dedicated SRL16 and SRL32 resources (integrated in LUTs)
- To obtain the best performance when using SRLs
  - Implement the last stage of the shift register in the dedicated Slice register
    - The Slice registers have a better clock-to-out time than SRLs
    - Synthesis tools often automatically infer this register
  - You should not code set/reset
  - Use the HDL coding styles represented in the Vivado Design Suite HDL Templates





Initialization of All Inferred Registers, SRLs, and Memories

- Xilinx highly recommends that you initialize all synchronous elements accordingly
  - Initialization of registers is completely inferable by all major FPGA synthesis tools
  - This lessens the need to add a reset for the sole purpose of initialization

VHDL

```
signal reg1 : std_logic := '0';
signal reg2 : std_logic := '1';
signal reg3 : std_logic_vector(3 downto 0):="1011";
```

Verilog

```
reg reg1 = 1'b0;
reg reg2 = 1'b1;
reg [3:0] reg3 = 4'b1011;
```

Constraints and Attributes

- **What are constraints and attributes?**
  - ATTRIBUTEs are directives that are provided in the HDL code itself
  - CONSTRAINTs are provided in a constraints file (XDC)
- **Both attributes and constraints provide guidance to specific tools on how to interpret and implement certain signals or instances**
- **Several properties can be provided as an attribute in the HDL or as a constraint in the XDC**
- **Accordingly, in the context of those properties, attributes and constraints are used interchangeably**

Constraints and Attributes

- **Xilinx recommends the following**
  - Embed directives that impact functionality as an attribute in the HDL code
  - Put temporary constraints (such as those required for debugging) in a separate constraints file
  - Remove any LOC, RLOC, or BEL constraints, or other physical constraints, embedded in the code or netlist of an existing design before retargeting to a new design or device

More Info

- Ug949: UltraFast Design Methodology Guide for the Vivado Design Suite, chapter 4
- Ug473: 7 Series FPGAs Memory Resources
- Ug479: 7 Series DSP48E1 Slice
- Ug901: Vivado Design Suite User Guide Synthesis