Vivado从此开始（To Learn Vivado From Here）

Vivado
从此开始
To Learn Vivado From Here
高亚军 编著
· 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
中国工信出版集团
电子工业出版社

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 1

Lauren Gao

Basic Functionality

- Blocking statements vs. Non-blocking statements
- Incomplete sensitivity list
- Latch inference
  - An if statement without an else clause
  - An intended register without a rising edge or falling edge construct
  - *WHY*: more difficult timing analyses
- Incomplete reset specification
  - the reset signal will get hooked to the CE pin, thereby creating another unique control set

all_latches

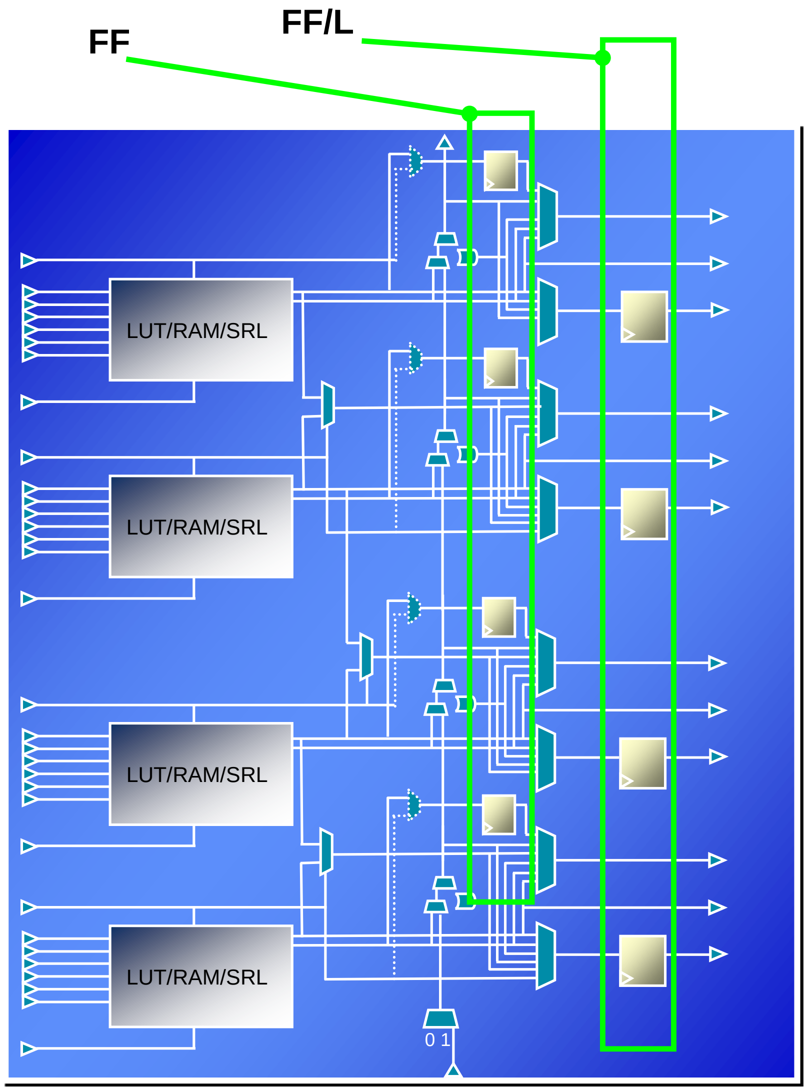
```
process (G, D)
begin
if (G='1') then
Q <= D;
end if;
end process;
```

```
always @(G or D)
if (G)
Q = D;
```

```
always @(posedge clk)
if (rst)
  reg1<= 1'b0;
else
begin
  reg1 <= din1;
  reg2 <= din2;
end
```

Slice Flip-Flops and Flip-Flop/Latches

- **Each slice has four flip-flop/latches (FF/L)**
  - Can be configured as either flip-flops or latches
  - The D input can come from the O6 LUT output, the carry chain, the wide multiplexer, or the AX/BX/CX/DX slice input
- **Each slice also has four flip-flops (FF)**
  - D input can come from O5 output or the AX/BX/CX/DX input
    - These don't have access to the carry chain, wide multiplexers, or the slice inputs
- **If any of the FF/L are configured as latches, the four FFs are not available**



Use of Loops in Code

- Pros and Cons
  - Minimize coding effort
  - May lead to inefficient structures thereby degrading performance
- **Xilinx recommends representing the same functionality using constructs that are easier for the tool to interpret**
- TIP
  - It is acceptable to infer loops for basic connectivity
  - when the code infers hardware resources (other than just wires/interconnects), it is better to avoid loops

```
reg [3:0] dout;
integer i;
always@(posedge clk)
begin
  for(i=0;i<=3;i=i+1)
    dout[3-i] <= din[i];
end
```

```
always@(posedge clk)
begin
  for(i=0;i<=3;i=i+1)
  begin
    if(en[i])
      dout[i] <= i;
  end
end
```

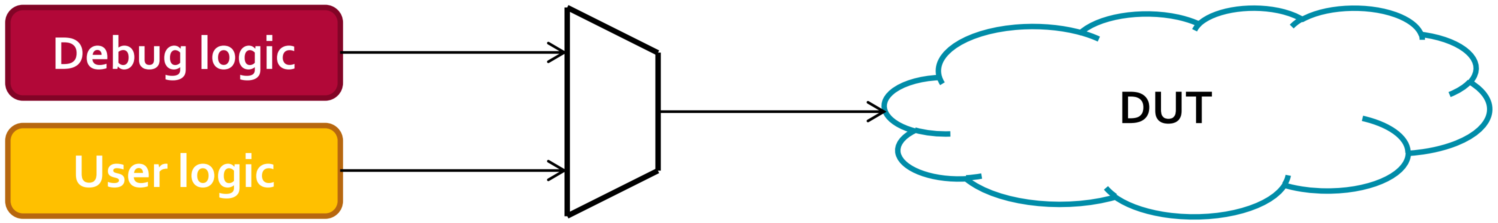
State-Machine Guidance

- Mealy vs. Moore Styles
  - Main difference:
    - Mealy: Current state + Input => output
    - Moore: current state => output
  - In general, Moore state machines implement best in FPGA devices
    - Most often one-hot state machines is the chosen encoding method, and there is little decode logic necessary for output values
- One-Hot vs. Binary Encoding
  - The two most popular for FPGA designs are binary and one-hot
  - **Vivado: FSM_ENCODING**
    - "one_hot", "sequential", "johnson", "gray", "auto" and "none", default: "auto"

```
(* fsm_encoding = "one_hot" *) reg [7:0] my_state;
--------------------------VHDL----------------------------
type count_state is (zero, one, two, three, four, five, six);
signal my_state : count_state;
attribute fsm_encoding : string;
attribute fsm_encoding of my_state : signal is "sequential";
```

Use of Debug Logic

- **Debug logic**
  - The logic that is not necessary for the design function, but which is useful in the design analysis
- **Several methods can assist in this objective**
  - Guard the logic with a `ifdef, parameter, or generic that can be set to disable or enable these sections of code
  - Code the logic in a way to more easily facilitate commenting it out for the future
  - Have a separate debug version of a module or entity to interchange for this purpose
- **Target**
  - Have a good methodology for debugging the design code
  - Have a good way to remove that logic

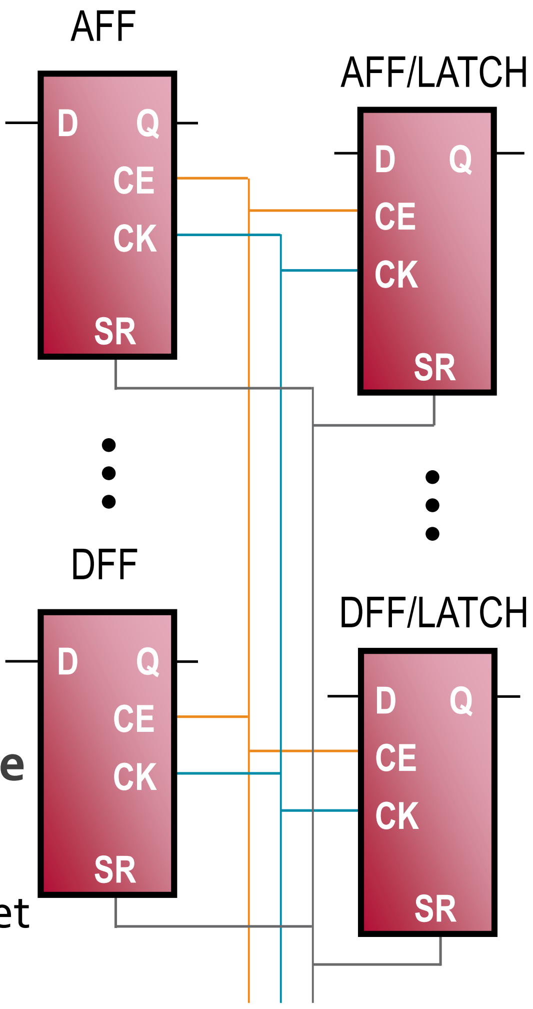


Control Signals and Control Sets

- **A control set is the grouping of control signals**
  - set/reset
  - clock enable
  - clock
- **Registers within a slice all share common control signals**
  - only registers with a common control set may be packed into the same slice
- **Designs with several unique control sets**
  - Have a lot of wasted resources
  - Fewer options for placement resulting in higher power and lower performance
- **Designs with fewer control sets**
  - Have more options and flexibility in terms of placement, generally resulting in improved results

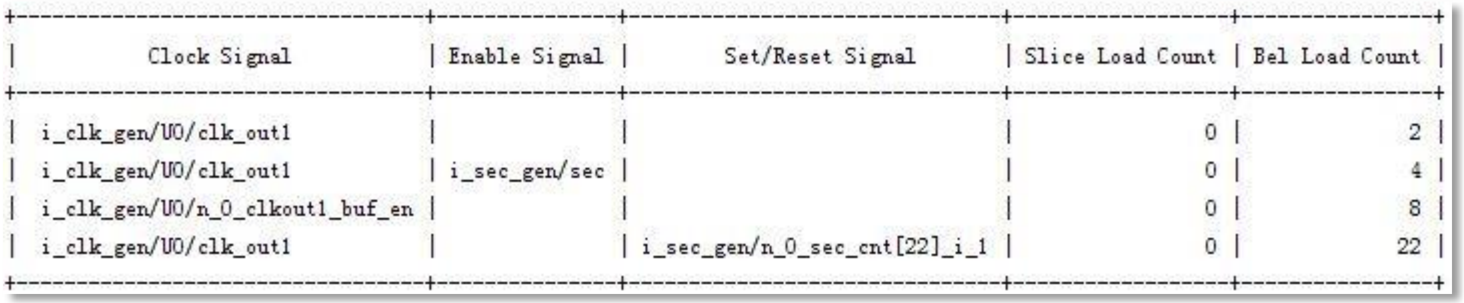
Control Sets

- **All flip-flops and flip-flop/latches share the same CLK, SR, and CE signals**
  - This is referred to as the "control set" of the flip-flops
  - CE and SR are active high
  - CLK can be inverted at the slice boundary
- **If any one flip-flop uses a CE, all others must use the same CE**
  - CE gates the clock at the slice boundary
  - Saves power
- **If any one flip-flop uses the SR, all others must use the same SR**
  - The reset value used for each flip-flop is individually set by the SRVAL attribute

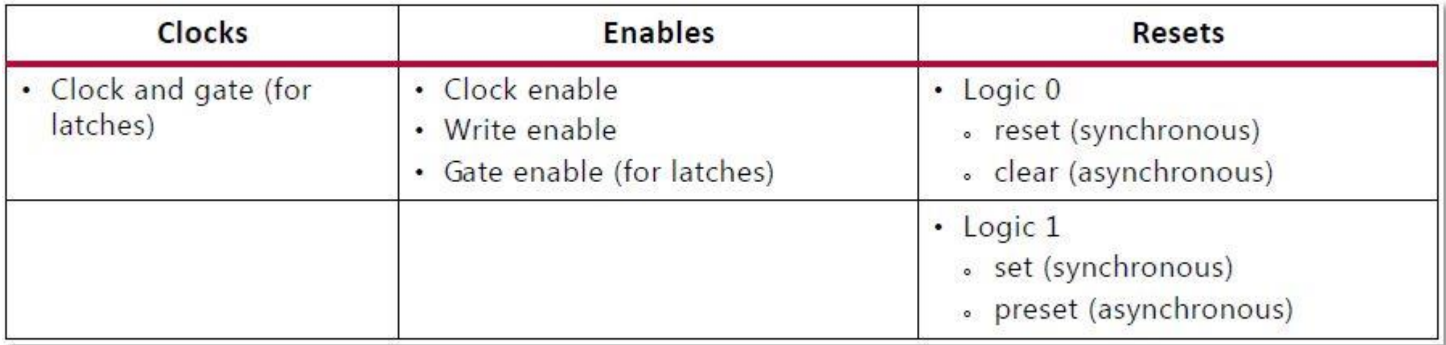


Control Set

- `report_control_sets`
  - Indicator of possible packing fragmentation and fitting issues
  - Run the `–verbose` option to generate a full list



```
+-----------------------------------+---------------+------------------------------+-----------------+---------------+
|            Clock Signal           | Enable Signal |       Set/Reset Signal       | Slice Load Count | Bel Load Count |
+-----------------------------------+---------------+------------------------------+-----------------+---------------+
|  i_clk_gen/U0/clk_out1            |               |                              |               0 |             2 |
|  i_clk_gen/U0/clk_out1            | i_sec_gen/sec |                              |               0 |             4 |
|  i_clk_gen/U0/n_0_clkout1_buf_en  |               |                              |               0 |             8 |
|  i_clk_gen/U0/clk_out1            |               | i_sec_gen/n_0_sec_cnt[22]_i_1 |               0 |            22 |
+-----------------------------------+---------------+------------------------------+-----------------+---------------+
```



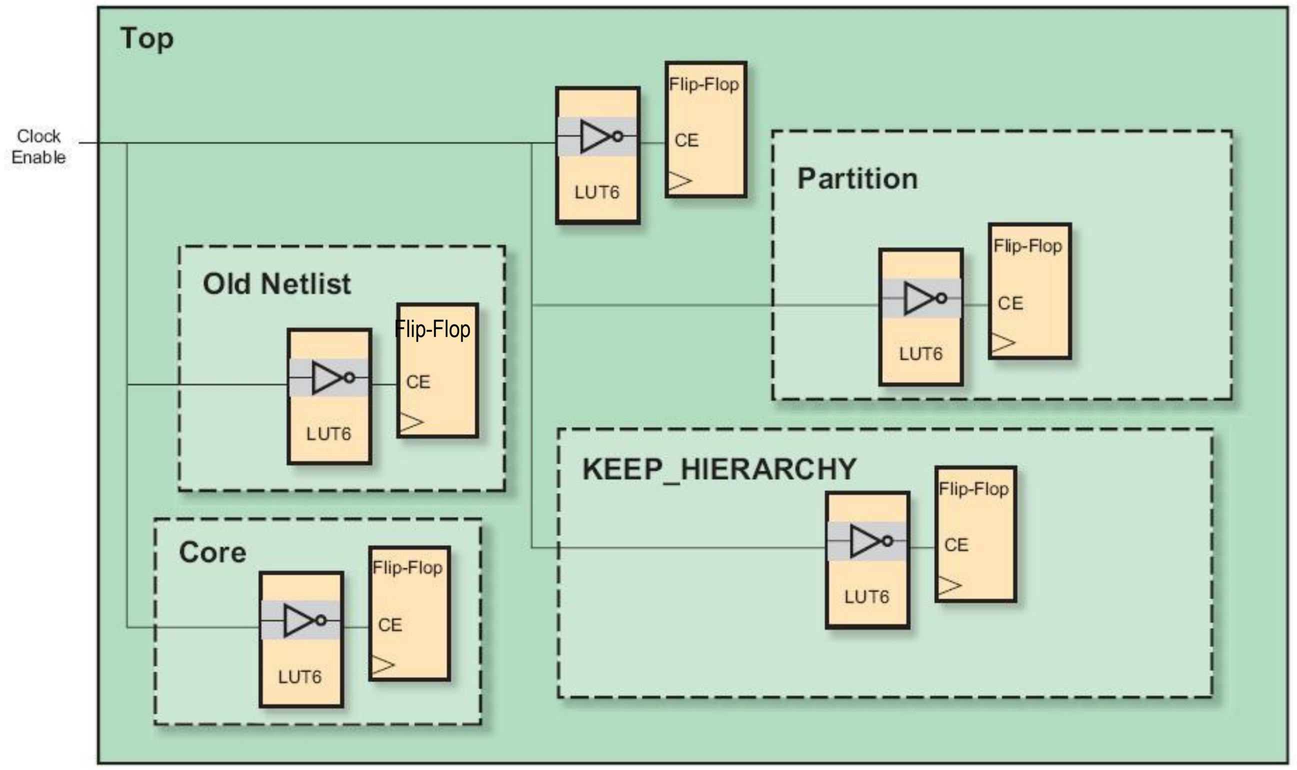
Clocks	Enables	Resets
• Clock and gate (for latches)	• Clock enable • Write enable • Gate enable (for latches)	• Logic 0 ∘ reset (synchronous) ∘ clear (asynchronous)
		• Logic 1 ∘ set (synchronous) ∘ preset (asynchronous)

When and Where to Use a Reset

- If an initial state is not specified, it defaults to a logic zero
- It is not necessary to code a global reset for the sole purpose of initializing the device
- Limits the overall fanout of the reset net
- Simplifies the timing of the reset paths
- Functional simulation should easily identify whether a reset is needed or not
- No reset brings much greater flexibility in selecting the FPGA resources to map the logic

With reset	Without reset
• registers with a common reset	• SRL • SRL + Registers • All registers • LUT or Block memory

Use Active-High Control Signals

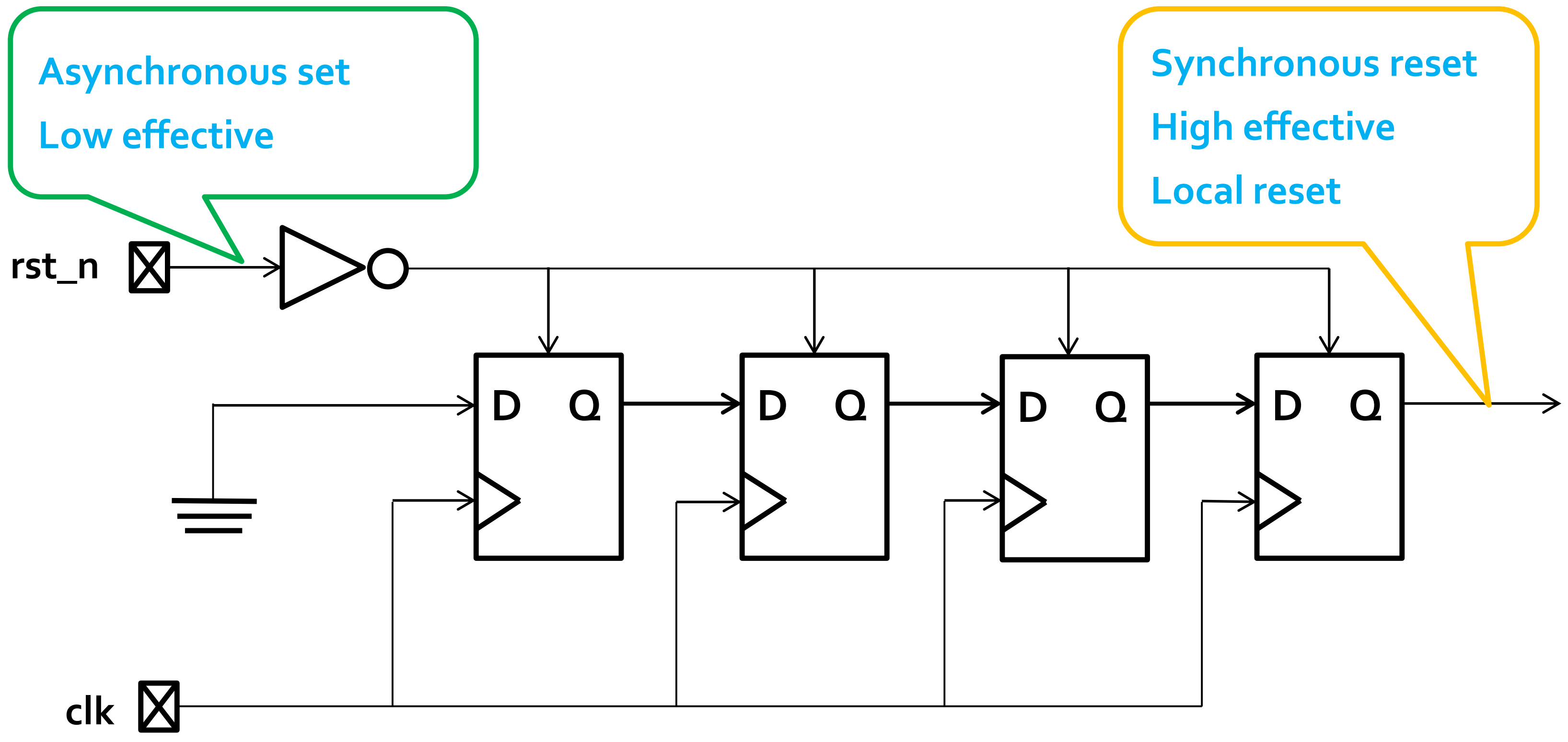


The inverters cannot be combined into the same slice

This consumes more power and makes timing difficult

Hierarchical design methods can proliferate LUT usage on active-low control signals

Control a Localized Reset Network



Synchronous Bridge

The number of flip-flops in the chain determines the minimum duration of the reset pulse issued to the localized network

Control a Localized Reset Network
Verilog

```
always @ (posedge clk or negedge rst_n) //async. Negedge reset
begin
  if (!rst_n)
    synchronizer_ckt <= 4’hf // 4 stage reset syncornization
  else
    synchronizer_ckt <= {synchornizer_ckt[2:0], 1’b0};
  end
assign synchronized_rst_n = ~synchronizer_ckt[3];
// the final reset signal which is used to reset the actual
// flops in the design
```

More Info

- Ug949: UltraFast Design Methodology Guide for the Vivado Design Suite, chapter 4
- Wp272: Get Smart About Reset: Think Local, Not Global