GFIR-55: WP275 Summary

Get Your Priorities Right - Make Your Design Up to 50% Smaller

By writing HDL code that respects the priority rules of flip-flop controls (reset, set, enable), you can significantly reduce the size and improve the performance of your FPGA design, potentially achieving up to 50% size savings.
(*"This white paper describes a rarely noticed design technique that can make a difference in the size and the performance of your FPGA design."*

"The ideas contained in this white paper might not yield the full 50% savings potential in your design, but should save a significant amount." ref: Page 1)

- Optimized code can reduce design size by up to 50%.

- Single-level logic is the most efficient approach. (*"The LUT can implement any function of four inputs regardless of how many gates are needed to describe it. The output of the LUT then feeds directly into the D input of the flip-flop (see Figure 1). Designs that pack together this well will be small and high-performance"* ref: Page 2)

VHDL Example

```
process (clk)
begin
  if clk'event and clk='1' then
    data_out <= a and b and c and d;
  end if;
end process;
```

Verilog Example

```
always @(posedge clk) begin
  data_out <= a & b & c & d;
end
```

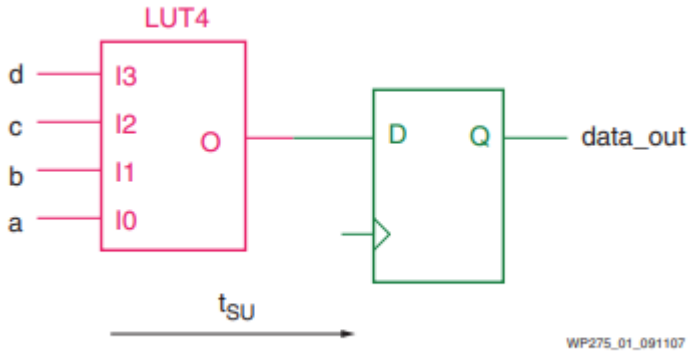

Figure 1: **Simple Logic Using a LUT and a Flip-Flop**

- More than four inputs force two-level logic, which doubles cost and decreases performance. (*"As soon as a function has more than four inputs, the synthesis tools have no choice but to split the logic between two or more LUTs in some way. Figure 2 shows the most likely way a six-input AND gate is implemented regardless of language or synthesis tool."* ref: Page 3)

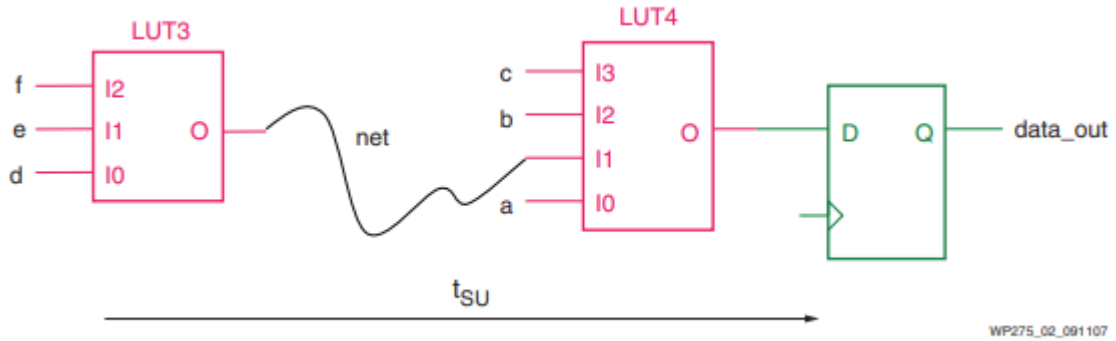

Figure 2: **Two-Level Logic**

- Global reset is usually unnecessary due to FPGA's defined power-on state. (*"Because a Xilinx FPGA already starts up in a known state following configuration, the global reset is really not necessary."* ref: Page 3)

- Avoid mixing asynchronous and synchronous controls on the same flip-flop. (*"However, they cannot support a mixture of asynchronous and synchronous controls on the same flip-flop. The synthesis tool must therefore choose between a synchronous flip-flop with SET and RST controls or an asynchronous flip-flop with PRE and CLR controls (see Figure 5). An asynchronous reset always takes priority and forces the selection."* ref: Page 5)

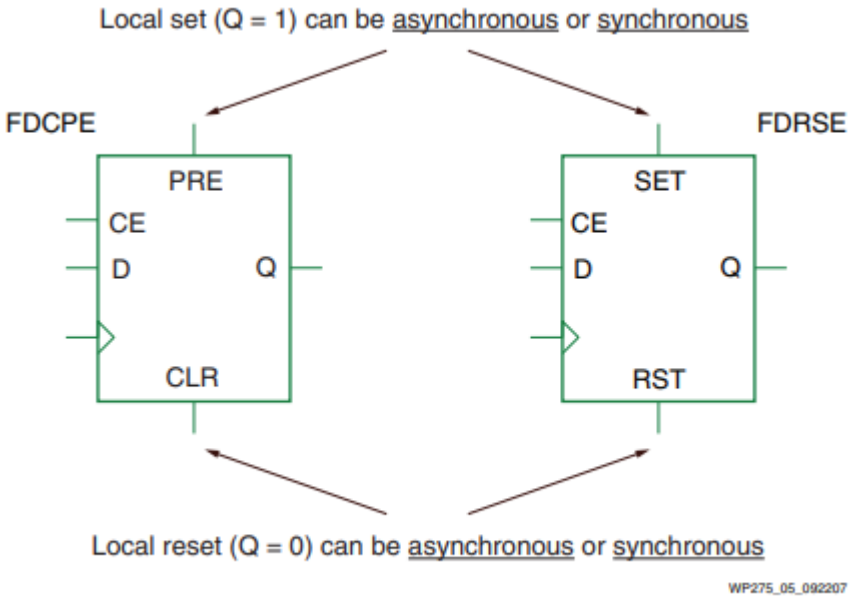

Figure 5: **Asynchronous versus Synchronous Controls**

- Asynchronous resets increase logic levels and reduce performance. ("*If the global reset is asynchronous, the local synchronous reset must be emulated using the LUT, with the potential to force two levels of logic at twice the cost and at lower performance.*" ref: Page 6)

- FPGA flip-flops have a fixed control priority: Reset > Set > Enable > D input.

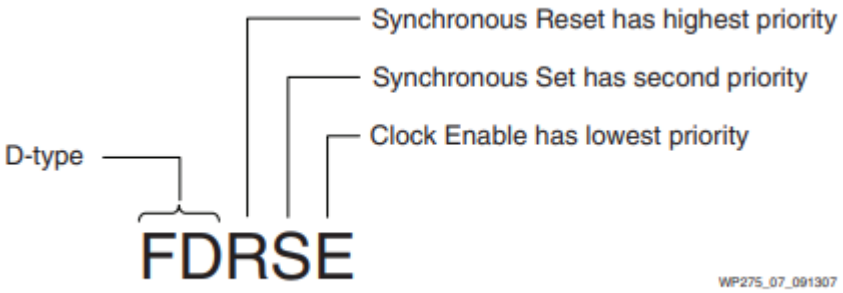

Figure 7: **FDRSE Definition**