**Summary of WP275**

Getting your prorities rigth is an optimization technique that allows you significantly reduce the size and improve the performance of FPGA designs. Through this way, potential savings can allow a design to fit into a smaller FPGA to get both cost and performance benefits.

FPGAs use Look-Up Tables (LUTs) and flip-flops within Configurable Logic Blocks (CLBs).

Efficient logic design ensures that functions fit within a single LUT, minimizing logic levels and improving speed.

When a logic function exceeds four inputs, synthesis tools split the logic across multiple LUTs.

This increases resource usage and delays due to added interconnections.

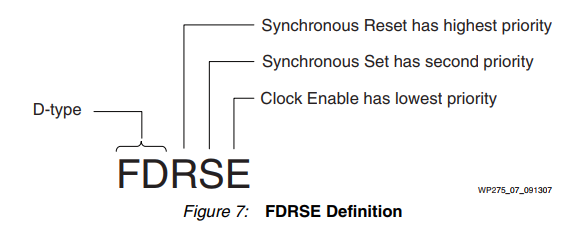
Designers often include a **global reset**, but since Xilinx FPGAs start in a known state, this is unnecessary. Using an unnecessary global reset can introduce extra logic layers, reducing efficiency.

|  |  |
| --- | --- |
| --global reset process (clk,reset) begin  if reset=’1’ then  data\_out <= ’0’;  elsif clk’event and clk=’1’ then  if enable=’1’ then  if force\_high=’1’ then  data\_out <= ’1’;  else  data\_out <= a and b and c and d;  end if;  end if;  end if; end process; | --Synchronous reset process (clk) begin  if clk’event and clk=’1’ then  if reset=’1’ then  data\_out <= ’0’;  else  if enable=’1’ then  if force\_high=’1’ then  data\_out <= ’1’;  else  data\_out <= a and b and c and d;  end if;  end if;  end if; end if; end process; |

Improperly implemented resets and enables can force the use of additional LUTs, increasing design complexity.

Mixing asynchronous and synchronous reset/set signals forces the synthesis tool to prioritize one. Improper priority assignments can lead to inefficient logic and increased resource consumption.

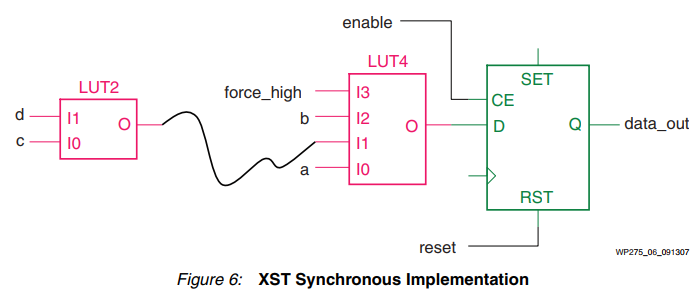
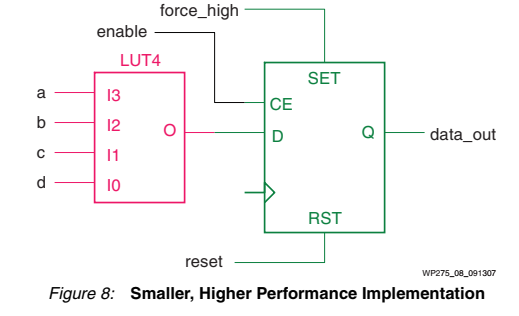
FDRSE priority order for flip-flop controls in Xilinx FPGAs is:

1. **Synchronous Reset** (highest priority)
2. **Synchronous Set**
3. **Clock Enable** (lowest priority)

Synthesis tools attempt to match HDL code to hardware resources but may introduce inefficiencies.Writing **sympathetic** HDL code that aligns with FPGA hardware priorities reduces logic levels and improves speed.

Example: Prioritizing reset, set, and enable in a way that maps naturally to hardware flip-flops.

|  |
| --- |
| --symphatetic code , control signal priorities aligns with the natural FF  process (clk) begin  if clk’event and clk=’1’ then  if reset=’1’ then  data\_out <= ’0’;  else  if force\_high=’1’ then  data\_out <= ’1’;  else  if enable=’1’ then  data\_out <= a and b and c and d;  end if;  end if;  end if;  end if; end process; |

By correctly prioritizing the design while considering FF priorities, the previously inefficient design seen in (Figure 6) is transformed into a faster and less resource-consuming design in (Figure 8).

Key Recommendations

* Avoid unnecessary global asynchronous resets.
* Structure control logic to maximize the use of dedicated flip-flop inputs.
* Reduce multi-level logic by optimizing the number of inputs per LUT.
* Align HDL coding styles with FPGA synthesis priorities for better resource efficiency.