APPLICATION NOTE

Mentor Synthesis Group

Using Register Retiming in Precision

Dan Devries – Technical Marketing Engineer

Mentor Graphics®

Introduction

Precision Synthesis includes a powerful optimization algorithm called register retiming for improving performance in FPGA designs. Retiming allows the optimizer to move registers across combinatorial logic to improve circuit performance. Improvements of up to 50% are not uncommon when using this algorithm.

Performing register retiming on a design will not change the functionality at the primary ports but may effect the observability of internal registers during post synthesis simulation. For this reason register retiming is not enabled by default – users must select retiming through the optimization "options" form. If observing internal registers during gate level simulation is not an issue users should feel comfortable enabling register retiming to solve timing issues.

The retiming process will add registers to a design. These additional registers do not add pipeline stages to a design and therefore do not add clock latency to a designs performance. The FPGA architectures from Xilinx and Altera are register rich and easily accommodate the additional registers inserted through retiming

How retiming works

A circuit with very critical timing will contain many non-critical paths that easily meet timing. This excess time available for data propagation, called slack, will be unevenly distributed, with some circuit paths having negative slacks, and some having positive slacks. Retiming will be carried out if proper slack budget is obtained through the proprietary budgeting algorithms. If the structure of the circuit is appropriate, it is possible to move the register forward or backwards, effectively moving some of the delay through the register, without affecting the functionality of the design at the primary output ports. Ideally, the result will be positive slack on both sides of the register.

Supported Technologies

Vendor	Device Family
Xilinx	Virtex
	Virtex E
	Virtex II
	Virtex II Pro
	Spartan II
	Spartan IIE
	Spartan III
Altera	APEX 20K
	APEX 20KC
	APEX 20KE
	APEX II
	Cyclone
	Excalibur-ARM
	Stratix
	Stratix GX

Note: When retiming is enabled for a non-supported technology a message will appear in the transcript indicating that it has been turned off.

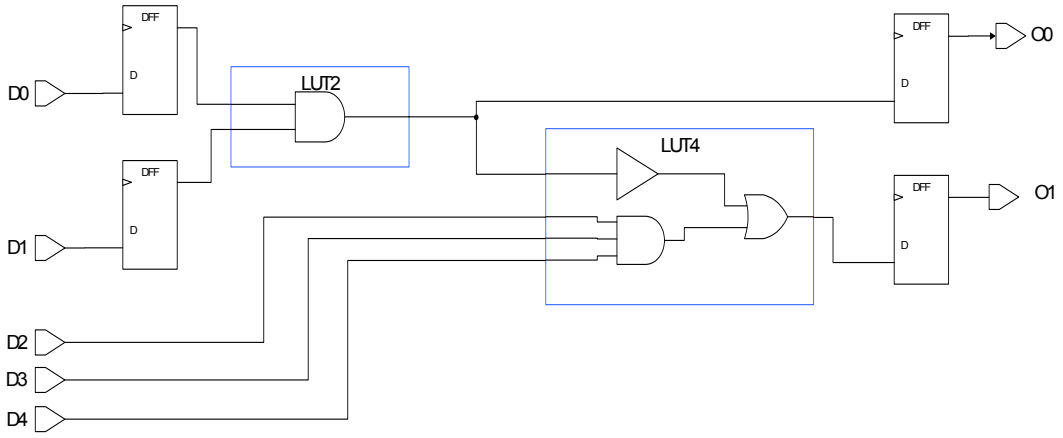

Figure 1 - Simple Circuit before Retiming. Slack = -0.44 ns.

Figure 1 shows a circuit as it might be implemented after normal optimization. The combinatorial logic between the 2 register banks was coded in such a manner that 2 serial LUTs were required to implement the logic. The critical path for this circuit goes through these 2 LUTs. As implemented, this design does not meet timing. Contrast this with the circuit shown in figure 2. Note that the 2 registers at the input have been merged into one register. The LUT2 has been moved behind the merged register. Retiming this circuit has resulted in one less LUT on the critical path, which causes this design to meet timing.

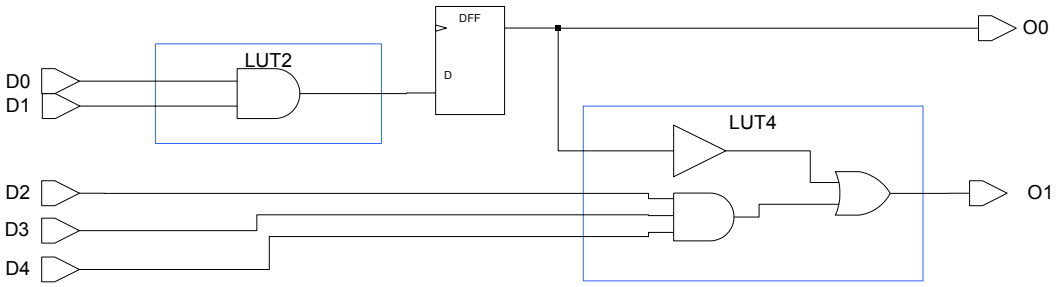

Figure 2. Simple Circuit after Retiming. Slack = 0.55 ns.

Register retiming will change the function of a local register. It can add new registers and merge existing registers. In all of these cases, debugging of the circuit can become more difficult. To aid the user, all register moves are reported in the transcript window, and in the log file. Retiming is performed iteratively. Circuit constraints are initially loosened to focus the first iteration on the most critical paths. Once this is complete constraints are slightly tightened. In this manner, it is possible to fix timing violations that are exposed after some retiming moves have occurred.

Up to 10 iterations can occur on a circuit, but the process will terminate when timing is met, or improvements are no longer possible. Registers that are retimed are renamed by adding "_FRT", as shown in the retiming report. The signal that is driven will also be renamed in the same manner. This is to remind the user that the register will not have the same function as it would in the RTL design.

Controlling Register Retiming

By default, register retiming is not performed. To enable retiming from the GUI, be sure that "Retiming" is checked in the Optimization form for Synthesis options. For batch operations, retiming can be controlled as follows:

Synthesis Setting	Precision Command
Enable Retiming	Setup_design –retiming
Disable Retiming	Setup_design –retiming=false

Table 1 – Enabling retiming commands

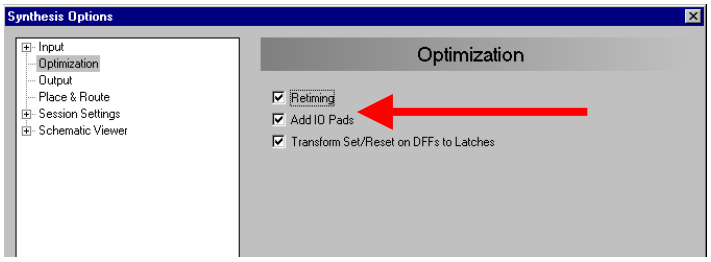

Figure 3. Enabling Retiming.

Retiming can also be disabled on a register-by-register basis or on a module basis. If the generic (RTL) register has the "dont_retime" attribute set, the register will not be retimed. This attribute can be set in the Design Browser or in the Schematic by using the popup menu "set attributes" on the Right-Mouse button. From a script, use the set_attribute command to selectively disable retiming. For example:

```
Set_attribute reg_dat25 –instance –name DONT_RETIME –value TRUE
```

Attributes can also be set in the source VHDL:

```
attribute DONT_RETIME : boolean;
attribute DONT_RETIME of dat25 : signal is true;
```

Or in Verilog:

```
Wire dat25;  // synthesis attribute dat25 DONT_RETIME true
```

Preventing Retiming on a hierarchical instance

The set_attribute command can also be used to prevent retiming on an entire block of a design. This can be done using wildcards as shown below. This would prevent retiming on all the registers in the instance I3

```
set_attribute -name DONT_RETIME -value "TRUE" -instance I3/*
```

Retiming Rules

For retiming to occur it must be possible to perform the transformation without modifying the function of the circuit at the boundary pins. An important consideration here is that the initial state (reset state) of the register be maintained. Additionally, it is important that design latency not change – the same number of register stages must exist before and after retiming. For retiming to occur, the following must be true:

- Registers must have proper timing budgets to be retimed. One example is having positive slack in on one side and negative slack on the other.
- All inputs of a LUT must have registers for a move to be possible (to maintain latency).
- Registers with both set and reset cannot be retimed.

While control signals must be consistent to allow retiming, control signals may change based on the function of the combinatorial logic. In figure 4 we see an example of retiming a register through inverting logic. Note that the original reset signal must be implemented as preset logic to maintain the same initial state at the outputs.

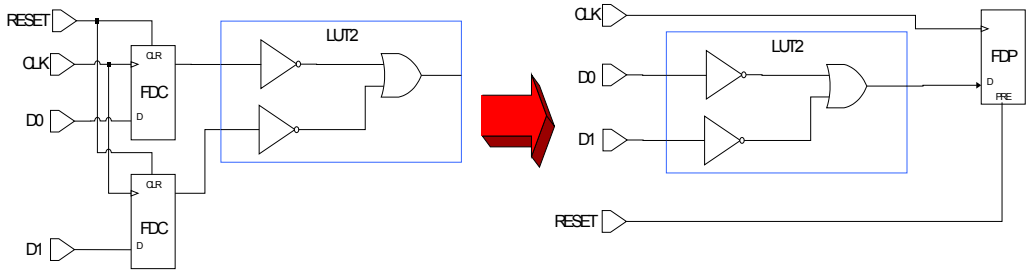

Figure 4. Reset Signal Changes to Preset.

.