EE Times Connecting the Global Electronics Community

Blue Pearl Software

3/20/2014

Shakeel Jeeawoody
Vice President, Marketing

The truth about knowing your False Paths

Timing Closure Challenges

Life would be simpler for a lot of engineers if their designs met their timing requirements the first time around. Unfortunately, it is not as simple as that. Time and effort spent to meet timing is increasing to unheard of levels. Designers resort to constraining their synthesis and place & route tools to achieve the required performance. However, generating the proper constraints is a hit-and-miss game for designers, while verifying them becomes someone else's nightmare.

Designers typically load the implementation tools with the design constraints and hope the tools give the results they need, i.e. zero timing violations. However in most cases after first pass the timing reports give 100s to 1000s of violations. When diving into the timing report looking at specific paths, they ask the question: Is it a real path that can occur? Static timing engines in the synthesis and P&R tools just trace through all paths in the design to find the longest path. They do not evaluate to see if the path can functionally occur. If the implementation tool can meet the timing, designers do not worry about this question. But if they can know which critical timing paths are false they could be eliminated from the timing report or, even better, eliminated from the optimization run. If the optimization tool has fewer paths to work on, the better chance it has in meeting timing on other critical paths.

Creating constraints to meet timing

Before deciding to make significant changes to their RTL descriptions, designers try to meet timing by constraining their synthesis and place &route tools. The initial constraints are somewhat easy and based on the designers specific knowledge of the design. These include specifying those paths which most probably will not be exercised during circuit operation. False paths and multi-cycle paths are the most effective constraints in enabling the designers to converge to the optimal implementation solution. If it takes multiple iterations to close timing, designers start guessing which constraints to use. In the end, it becomes very difficult to determine which constraints were necessary and which ones are the most effective.

Chip designers try different tactics to attempt to solve their timing problems. They may change tool switches or over constrain the tools. They may evaluate the constraints to determine what errors or incorrect assumptions may exist. However, this may result in runtime increase and there may be limited improvement. The next phase is determining if the critical path is false. Some false paths are dependent on the functionality of the input pins; this determination may require functional knowledge of the design and may be validated via a simulation assertion. This can be a timing consuming task and if the design changes, paths may have to be re-evaluated. The amount of engineering effort could be greatly improved if there was a way to automatically find structural false paths and generate an assertion to validate. If RTL changes are required the false path generation can just be re-run.

Focus on fixed obstacles first

For a particular RTL description, false paths and multi-cycle paths are like fixed barriers on the obstacle course to meet timing. In effect, the critical paths of a design are moving targets that change as the designers modify the setup options for their synthesis and place and route tools. Focusing on the critical paths from the start may not be the right thing to do. It is advisable to focus on the fixed obstacles, a.k.a. the false paths and multi-cycle paths, first. Using its proprietary technology, Blue Pearl Software has built intelligence into its tools to automatically detect and report false paths and multi-cycle paths in an RTL description. Obviously, there are some paths that the designers with proprietary knowledge can easily specify. So the question is, what types of paths can the Blue Pearl software tool find automatically? In the next sections, we will discuss some of the different types of false paths that Blue Pearl can automatically detect.

Blue Pearl's "SDC" Timing Constraint Generation tool employs symbolic simulation to analyze and automatically generate functional false paths. Blue Pearl recognizes false paths with control logic from finite state machines (FSMs), sequentially controlled data paths and counters.

By rapidly analyzing sequential control behavior, Blue Pearl generates complex functional false paths that would be very time-consuming or even impossible to identify manually. The tool also generates False Path schematics and assertions (PSL & SV) to aid and improve confidence in the generated SDC.

Blue Pearl False Path Detection

A path is a sequence of logic elements through which data can propagate, bounded by either ports or registers. A path is false if no sequence of input vectors can result in an event propagating along it. Such paths may be ignored for timing analysis purposes.

Now, let's take a look at some false path examples that Blue Pearl recognizes:

1. ***Simple false Paths***
   A simple false path is shown in **Figure 1** below. The path from Reg B to Reg C can never be sensitized. This occurs because whenever the output of Reg B is selected by Mux1, the output of the cloud of logic is de-selected at Mux2.

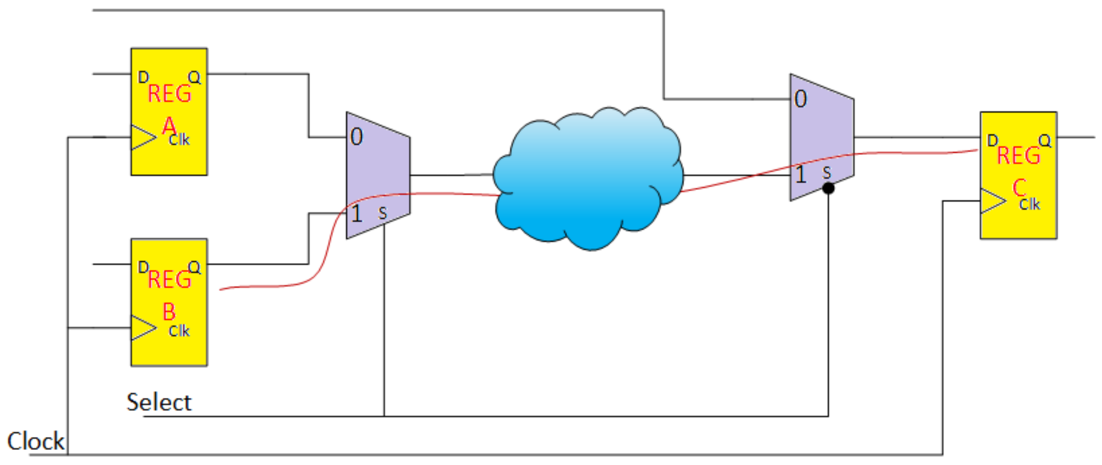

Figure 1

2. *Finite state machines (FSMs)*

The Blue Pearl Software Suite will also recognize False-cycle paths controlled by FSMs, see **Figure 2** below. The simple state machine below is implemented by a case statement.

```
case (state)
s0: begin
nextstate <= s1;
end
s1: begin
nextstate <= s2
end
...
s7: begin
nextstate <= s0
end

assign Select1 = (state == s3) ? 1'b1 : 1'b0;
assign Select2 = (state == s6) ? 1'b1 : 1'b0;
```

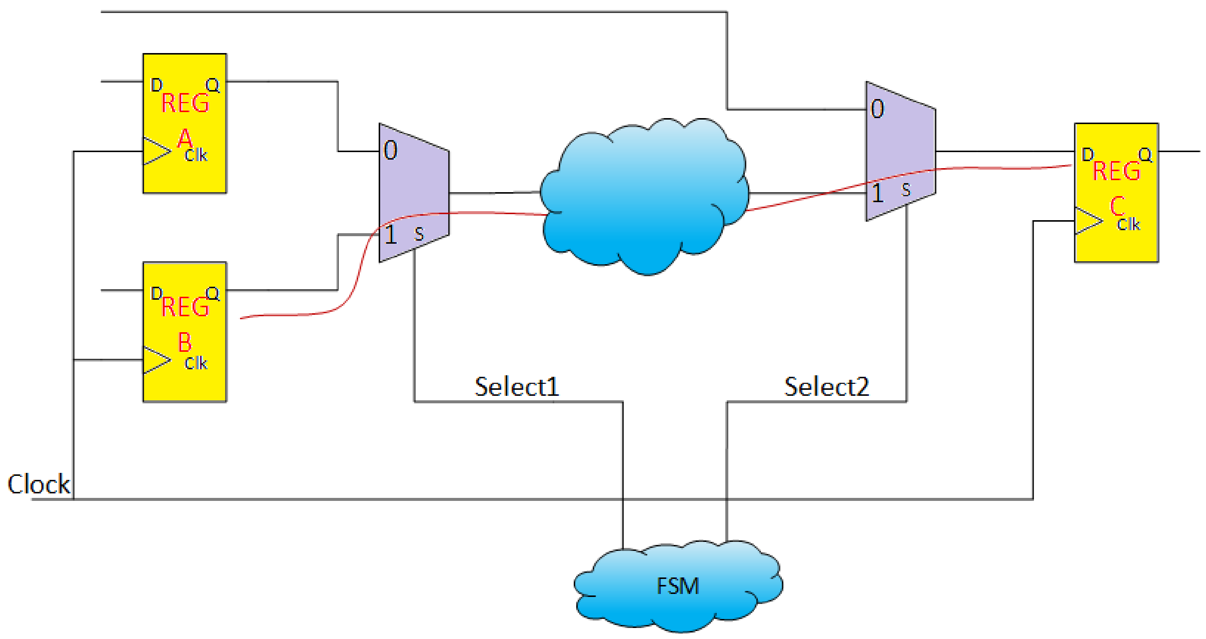

Figure 2

3. *False path with sequential control signals*

By default, timing constraint generation for false paths involves only a single cycle analysis. Multi-cycle path analysis, on the other hand, involves an analysis of the design over multiple cycles. For designs containing decoding logic (such as one-hot decoders, pulse generators, pipelined structures), a multiple cycle sequential analysis is required.

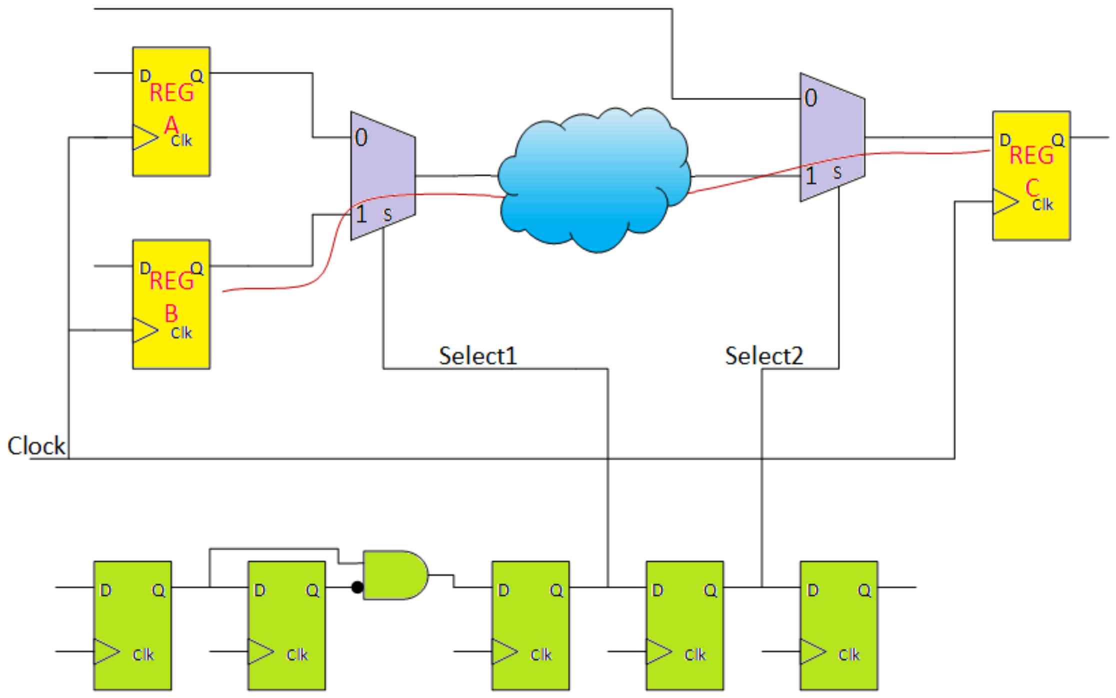

Figure 3

In the example depicted in **Figure 3** above, select 1 and select 2 are delayed versions of the output from a pulse generator. Hence select1 and select 2 cannot be high at the same time resulting in false path.

Likewise, in the example shown in **Figure 4** below the control logic is such that it is controlled by pipeline. The conflicting logic can be multiple levels down the pipeline. Blue Pearl can identify False Paths based on such control logic.

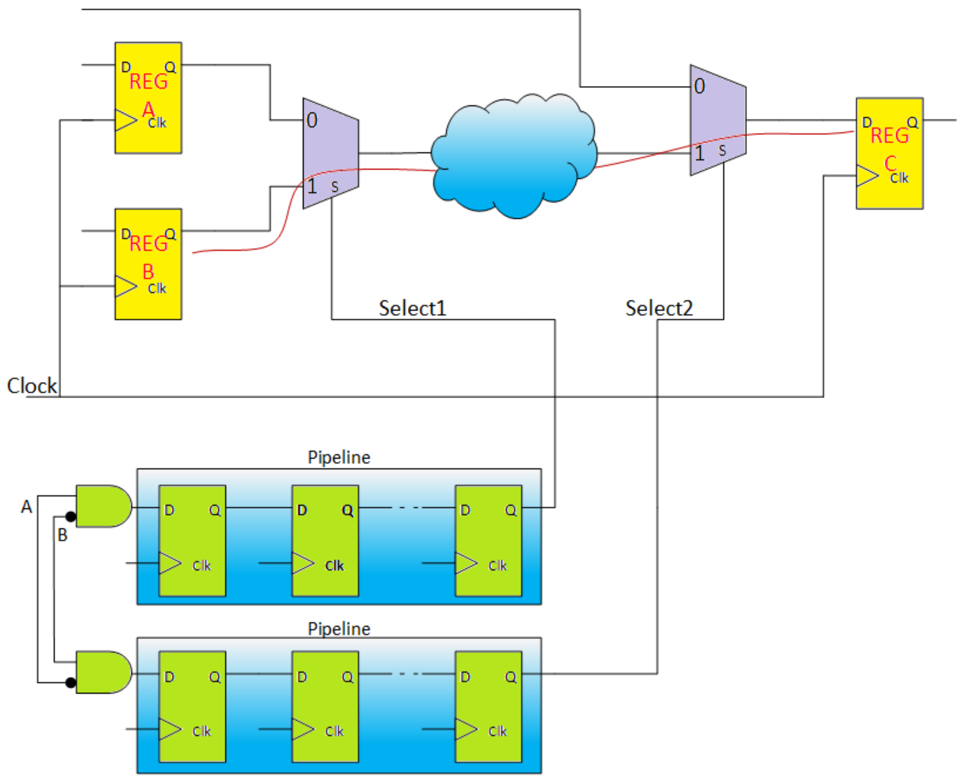

Figure 4

What's the take away?

Using Blue Pearl Software, designers can automatically find false paths hidden in their designs instead of going through the error-prone and time consuming manual method. **Figure 5** below illustrates a sample design where a false path is automatically found across multiple levels of hierarchy. In this example, the data paths from source registers to destination registers are indicated by the green, blue and orange lines. The pink line is the signal controlling the dataflow though the AND gate.

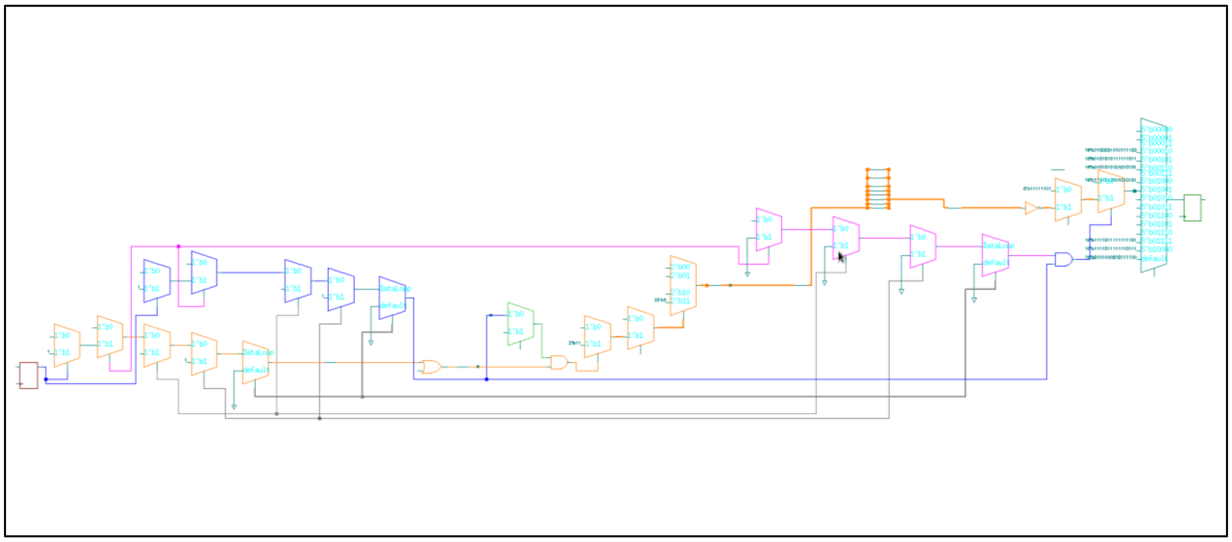

Figure 5

As these false paths remain the same for a particular RTL code, designers can eliminate them from consideration and focus their effort on the real critical paths. So, knowing your false paths early helps guide the implementation tools to the optimum solution faster.