FishTail: The Automated Generation of Golden Timing Constraints

IC design is not getting any easier. With increased gate counts, higher clock speeds, smaller chip sizes and reduced power requirements, designers have a very difficult task. Today's virtual prototyping and chip-implementation tools are powerful and address several key deep-sub micron issues, but there remains a fundamental conundrum. Precise constraints on chip timing, upon which the design ultimately succeeds or fails, remain in a state of flux throughout the design cycle. False paths and multi-cycle paths are typically entered only in response to timing problems. As timing problems seriously manifest themselves only during place & route, this is late in the design cycle to be tweaking your fundamental timing goals. All of this results in extra timing closure iterations, chips that consume more area and power than they should, chips that do not run as fast they could, and a messy handoff from chip design to implementation teams.

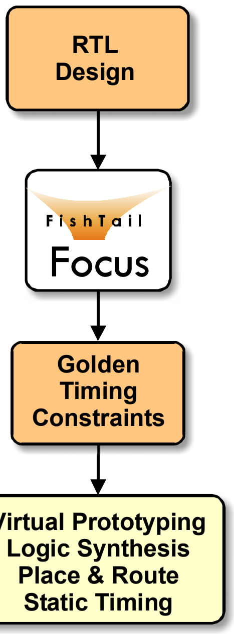


Figure 1: The Automated Generation of Golden Timing Constraints.

FishTail Design Automation has developed ground-breaking, patented technology to solve this problem, reducing risk and improving design quality. FishTail's Focus product, as shown in Figure 1, starts with the RTL description for a design. Focus generates a template clock definition file that points out the nets on the design on which clocks and generated clocks should be defined. This template clock definition file is then massaged by users to generate the final clock constraints for the chip. Next, the clock constraints and RTL are fed into Focus and used to identify the clock-modes on the chip. Clock modes result from multiplexing in the clock generation circuitry. Finally, for each clock mode, Focus analyzes the intended functionality of the chip in the context of how it will be clocked and establishes timing relaxations (false and multi-cycle paths). These automatically identified false and multi-cycle paths, along with the clock specifications for the design and case-analysis statements for each mode on the design are written out in standard Synopsys Design Constraint (SDC) format. Focus generated constraint files are then used to drive virtual prototyping, logic synthesis, and place & route tools.

Customers derive substantial value by deploying FishTail in their design flow. FishTail allows them to:

1) Improve the quality of results (QoR) of the final chip implementation. Timing relaxations focus the attention of synthesis tools on the real timing challenges on the design, and stop their distraction with the optimization of paths that are false or multi-cycle. As a result, the overall timing of the chip is significantly improved, with modest improvements in the area and power consumption of the chip.

2) Reduce the time spent in the back-end design flow to close timing. The identification of timing exceptions early in the design flow reduces the number of timing problems that need to be manually addressed during timing closure. This, in turn, reduces the time taken to close timing because there is less back-and-forth between implementation and design engineers in establishing whether timing problems are real or not. The time taken to resolve back-end timing problems is particularly significant when chip-design and chip-implementation teams span geographical and business boundaries.

3) Eliminate the risk of silicon failure that results from the application of incorrect timing exceptions. When engineers are under the gun to tape-out a chip they make mistakes and enter timing exceptions that are either incorrect, or broader in their scope than is legitimate because of the use of wildcards. The automated generation and verification of timing exceptions is a formal approach to timing closure that ensures that timing constraints go through similar levels of scrutiny that other aspects of chip design and implementation already do.

Applying FishTail to the Front-End Design Flow

When FishTail is deployed in a front-end design flow, shown in Figure 2, the timing exceptions generated by Focus are integrated with synthesis tools. The flow commences by using Focus to generate complete timing exceptions (false paths and multi-cycle paths) for a design. Synthesizable RTL for a design (Verilog, VHDL, or a mix) is provided as input to Focus. For hard macros, memories, and library cells that are instantiated in the RTL (for which synthesizable descriptions do not exist) simulation models are provided. If simulation models do not exist, then .lib files are read in. Clock and boundary constraints are provided in SDC format.

For each timing exception generated by Focus, assertions are also generated. Assertions provide the rationale for why a false or multi-cycle path definition is correct. These assertions are verified using an assertion-based verification methodology that requires the use of functional simulation or property checking to establish the correctness of Focus generated timing exceptions. The assertions are generated in a variety of formats (`PSL`, `SVA`, `OVA`, `OVL`) to facilitate their integration into commonly used functional simulation and property-checking tools.

While every effort is made to generate false paths that only refer to clocks, registers or hierarchical pins on a design, sometimes a path is false only when a specific internal net is traversed. It is possible that the internal nets referred to by Focus are not preserved by a synthesis tool. Also, attaching a timing exception to an internal net limits the optimization flexibility of a synthesis tool. The net cannot be optimized away because a timing exception is attached to it, and this can hurt synthesis QoR. For these reasons, the SDC file generated by Focus is partitioned into two files: one that contains exceptions that do not refer to any internal nets and another that contains exceptions that refer to internal nets. The exceptions that do not refer to internal nets are sourced into logic synthesis prior to compiling the design. The initial compile is performed without ungrouping the design.

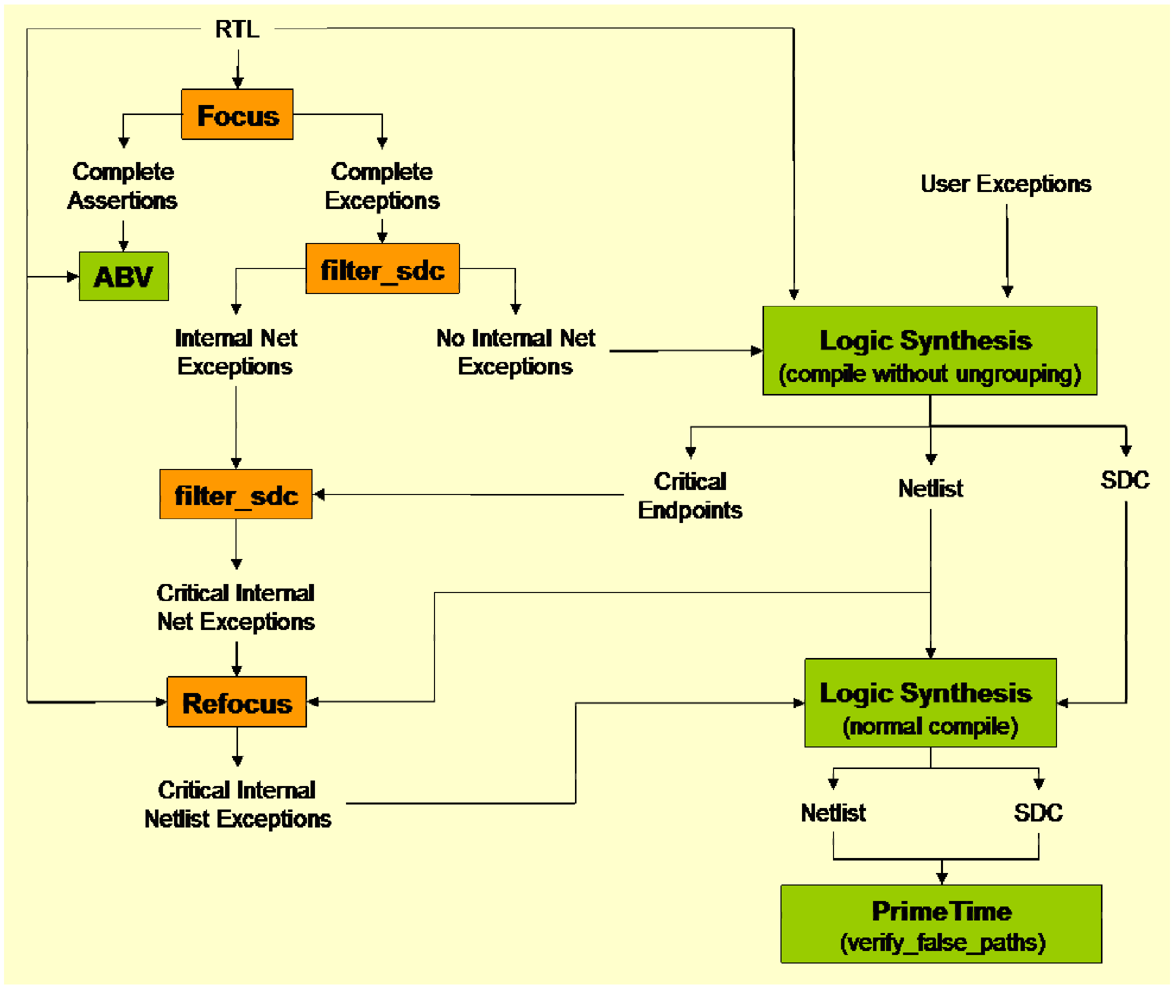

RTL
Focus
Complete Assertions
Complete Exceptions
User Exceptions
ABV
filter_sdc
Internal Net Exceptions
No Internal Net Exceptions
Logic Synthesis (compile without ungrouping)
Critical Endpoints
Netlist
SDC
filter_sdc
Critical Internal Net Exceptions
Refocus
Logic Synthesis (normal compile)
Critical Internal Netlist Exceptions
Netlist
SDC
PrimeTime (verify_false_paths)

Figure 2: Application of FishTail to the Front-End Design Flow.

Once the netlist from the initial compile is obtained a procedure is run within the synthesis tool to list all timing endpoints that do not meet timing. The SDC file generated by Focus that refers to internal RTL nets is now filtered so that timing exceptions that do not apply to timing-critical endpoints are filtered out. This yields a set of critical timing exceptions that refer to internal RTL nets. These exceptions cannot be directly applied to the synthesized gate-level netlist, because internal RTL nets are not preserved during logic synthesis.

Refocus, a tool that maps RTL objects to netlist objects (and vici versa), is used to map the timing-critical exceptions that refer to internal nets. Refocus reads in the RTL description for a design, the synthesized netlist and the RTL SDC file. It maps references to internal RTL nets to corresponding gate-level pins in the netlist. Sometimes, logic synthesis restructures and optimizes the combinational logic to such an extent that a single unique gate-level pin that corresponds to the original RTL net cannot be found. When this happens, Refocus filters out exceptions that refer to these RTL nets. This is not a problem, because optimal QoR requires giving synthesis maximum flexibility in logic optimization even if it means being unable to apply some timing exceptions.

The output of Refocus is a gate-level SDC file that contains timing critical exceptions. These exceptions are sourced into synthesis and then an incremental compile is performed. This compile takes as input the netlist and SDC generated from the initial compile and the Refocus generated SDC file, as shown in Figure 2. There is no restriction on how this incremental compile is performed and so the design may be ungrouped at this stage.

The netlist and SDC generated by the second compile is handed off to the place and route team. At this point, the gate-level false paths referred to in the SDC file are verified in PrimeTime using a FishTail developed procedure. This procedure uses the false-path sensitization capability in PrimeTime to confirm that all of the Focus generated false paths are correct, even after logic synthesis and the Refocus SDC mapping process.

Table 1 summarizes results from the application of the front-end FishTail flow to a 40K instance, single clock, JPEG decoder. QoR data was obtained using the traditional synthesis flow without timing exceptions and compared against the FishTail flow. The data demonstrates that the application of FishTail to the front-end design flow results in a significant improvement to chip timing accompanied by a modest improvement to chip area.

Logic Synthesis Data	**Without Exceptions**	**With Exceptions**
Worst Negative Slack (WNS)	-1.68 ns	-0.22 ns
No. Violating Endpoints	355	105
Total Negative Slack (TNS)	-251.6 ns	-12.89 ns
Cell Area (units)	68949	68350

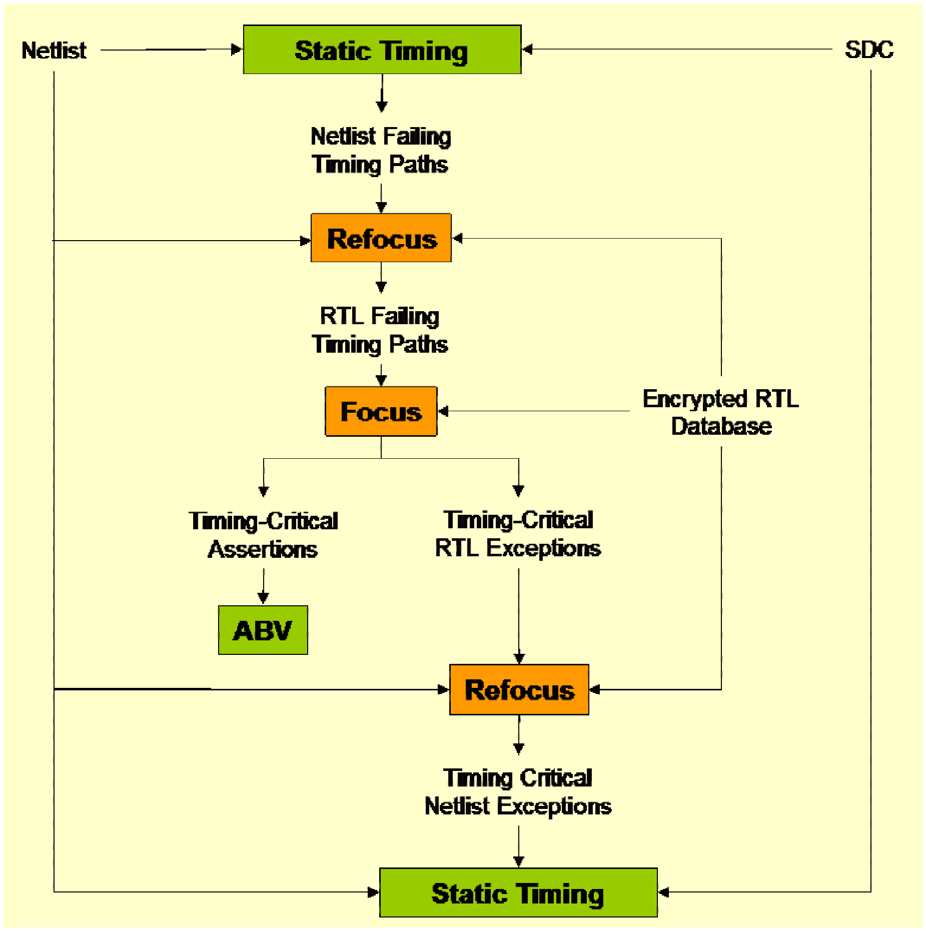
Table 1: Results from the application of FishTail to the Front-End Design Flow.

Applying FishTail to The Back-End Design Flow

When applied to a back-end design flow FishTail is used to eliminate timing problems that are not real, i.e. the failing timing paths are false or multi-cycle. This flow, shown in Figure 3, commences by using a procedure to write out the list of failing timing paths (start and endpoint pairs) from a static timing tool. These are the timing paths that do not meet timing after place and route.

Endpoint names in a back-end netlist are typically different from their RTL names because of hierarchy removal, name changes, etc. Before Focus is able to generate timing exceptions for these timing paths, back-end endpoint names need to be transformed to RTL endpoint names. Refocus is used for this purpose and takes as input a list of failing back-end timing paths, the back-end netlist and an encrypted RTL database. Refocus writes out a list of failing RTL timing paths using RTL names for the endpoints.

The encrypted RTL database used by Refocus is generated using a free utility provided by FishTail that packages up the RTL for a design and writes out a binary file that is then encrypted using AES. The encrypted design database allows back-end place and route teams to run Focus even if they do not have access to the RTL description for a design – it is impossible to reverse engineer the RTL from the encrypted RTL database.



Netlist
Static Timing
SDC
Netlist Failing Timing Paths
Refocus
RTL Failing Timing Paths
Focus
Encrypted RTL Database
Timing-Critical Assertions
Timing-Critical RTL Exceptions
ABV
Refocus
Timing Critical Netlist Exceptions
Static Timing

Figure 3: Application of FishTail to the Back-End Design Flow.

Next, Focus is used to generate exceptions for the failing timing paths. Focus takes as input the list of failing timing paths generated by Refocus, the encrypted RTL database for the design and the clock and boundary constraints for the design. As back-end place and route is done flat for large designs with greater than 500K cell instances, exceptions for the failing timing paths are generated using a transparent hierarchical analysis capability within Focus. Engineers specify the major functional blocks on a design and Focus internally breaks up the analysis of the full design so that exceptions for timing paths that are fully contained within a functional block are generated separately for each functional block. Then, timing-exceptions between functional blocks are generated by analyzing a design representation that only includes the interface logic for each functional block. A single SDC file is generated, and a single Focus run is performed to generate this SDC file, but by breaking up the analysis internally into hierarchical pieces, capacity restrictions are eliminated even on 32-bit operating systems.

The SDC file generated by Focus refers to RTL objects and cannot be applied to the back-end netlist directly. Refocus is used to map references to RTL objects to back-end netlist objects and generate an SDC file that is suitable for consumption by back-end tools. The SDC file generated by Refocus is brought into a static timing tool and timing violations that result from false or multi-cycle paths are eliminated.

Table 2 summarizes results from the application of FishTail to a back-end design flow. As may be observed, a significant improvement in chip timing is made possible by automatically generating timing exceptions for the failing timing paths on a design.

Static Timing Data	Without Exceptions	With Exceptions
Total Negative Slack (TNS)	-8497 ns	-1121 ns
No. of Setup Violations	385	77
Percentage Reduction in TNS	-	87%
Percentage Reduction in Violations	-	80%

Table 2: Results from the Application of FishTail to the Back-End Design Flow.

Completeness of Focus Generated Exceptions

Focus discovers most of the timing exceptions on a design but not all of them. False paths that are not generated by Focus fall into one of three categories:

1) The false path applies to a timing don't care. For example, a false path from reset is often entered by designers because the reset timing is irrelevant. There is nothing functionally false about paths from the reset port on a design; it is just that it isn't necessary for these paths to meet timing.

2) The false path results from a mode register that is software programmed to a constant value. As the value on a mode register does not change, engineers often mark as false all paths that start at such registers. This, again, is not something that can be functionally gleaned from the RTL.

3) The false path only manifests itself after logic synthesis. For example, the implementation of an adder is not visible in the synthesizable description for a design. Once the adder is synthesized there may be false paths within its structure, but as these were not visible at the RT level, Focus will not identify these.

Multi-cycle paths are not generated by Focus when an input port is used to control propagation along timing paths internal to a block. Focus assumes that an input port will assume a new value at each clock cycle. If, in fact, the input port has specific behavior (for example, being asserted every other clock cycle) that is the reason for multi-cycle behavior within a block, then in the absence of an appropriate clock waveform on the input port, Focus will not identify this multi-cycle behavior.

Correctness of Focus Generated Exceptions

FishTail guarantees the results generated by its tools. All false paths generated by Focus are guaranteed to be statically unsensitizable. Further, if a Focus generated false path is dynamically sensitizable it is guaranteed to not be the critical path to a timing endpoint. Bottom line – all Focus generated false paths are guaranteed to be correct. This can be confirmed by using the gate-level false-path verification procedure provided by FishTail for execution within PrimeTime or using assertion-based verification at the RT level.

All multi-cycle paths whose assertions are proven using functional simulation are correct. Multi-cycle path generation requires formally analyzing the sequential behavior of a design. For reasonably complex real designs formal analysis is only possible for a limited number of clock cycles after which the analysis becomes too computationally expensive to continue. As a result, it is necessary to verify the assertions generated for multi-cycle paths and to filter out any that do not hold when a design is simulated for a large number of clock cycles.

Summary

FishTail software allows design engineers to focus the attention of synthesis and place and route tools on the real timing challenges posed by designs. This results in a significant reduction (on the order of 6-8 weeks on average) in the time taken to close chip timing and also provides a modest reduction in chip area and power consumption. Finally, the automated generation and verification of timing exceptions results in a more robust chip-implementation flow, where the risk of silicon failure because of incorrect timing exceptions is completely eliminated. FishTail tools are production ready and have been applied to thousands of complex Verilog, VHDL and mixed-language designs.