Enhanced SDC Support for Relative Timing Designs

University of Southern California

Eric Quist (equist@usc.edu)

Peter Beerel (pabeerel@usc.edu)

University of Utah

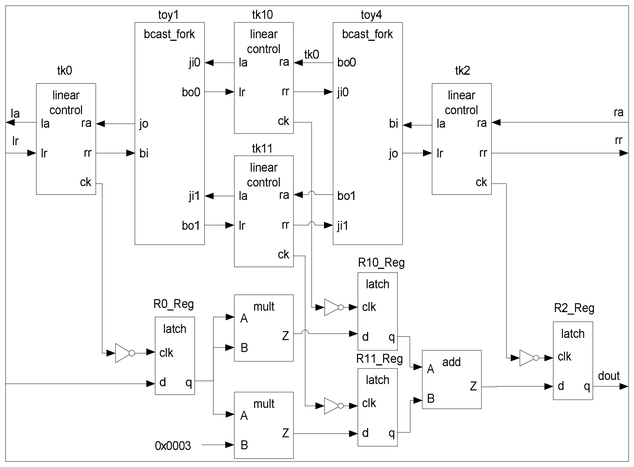
Kenneth Stevens (kstevens@ece.utah.edu)

Introduction and Motivation

The context – Asynchronous Circuits

- Synchronous designs are faced with increasing complexity
  - Problems include clock tree synthesis, hold time fixing, clock skew management and increasing power consumption
- Asynchronous designs offer several advantages
  - Low power consumption
  - Low latency and high throughput
- Relative Timing Designs
  - Asynchronous circuits rely on non-standard timing constraints
  - These constraints have been captured in the newly proposed, more general relative timing paradigm

Sample Relative Timing Circuit



- This example consists of both a forks and joins
- Since the fork and join cells are the same, simple naming conventions cannot fully be generalized using current tools

Motivation

- Current state of the art
  - Constraints such as set_max_delay, set_mid_delay, set_data_check, and set_disable_timing allow for relative timing circuitry
  - There is no mechanism to take advantage of hierarchal circuit design
  - Such constraints must be specified instance-by-instance for the entire design
- The problem
  - Asynchronous Timing cells rely on precise timing constraints
  - Current tools were designed for mainly synchronous designs
  - With current tools timing constraint generation is cumbersome and error-prone
  - This can lead to timing escapes and failing circuits

The Solution – An SDC Language Enhancements and Pre-processor

Current sdc constraints file

```
create_clock -period $clk_period -waveform "0 $clk_phase" tk0/lr
create_clock -period $clk_period -waveform "0 $clk_phase" tk10/lr
create_clock -period $clk_period -waveform "0 $clk_phase" tk11/lr
create_clock -period $clk_period -waveform "0 $clk_phase" tk2/lr

set_disable_timing -from A2 -to Y  [find -hier cell *lc1]
set_disable_timing -from B1 -to Y  [find -hier cell *lc1]
set_disable_timing -from A2 -to Y  [find -hier cell *lc3]
set_disable_timing -from B1 -to Y  [find -hier cell *lc3]
set_disable_timing -from B -to Y  [find -hier cell *c_element1]
set_disable_timing -from B -to Y  [find -hier cell *c_element2]
set_disable_timing -from A0 -to Y  [find -hier cell *lc3]
set_disable_timing -from B0 -to Y  [find -hier cell *lc3]
set_disable_timing -from A1 -to Y  [find -hier cell *lc1]

set_data_check -clock     tk0/lr    -fall_from  tk0/lc1/A2  \
          -rise_to   tk0/lc1/B1    -setup $race_margin
set_data_check -clock     tk10/lr   -fall_from  tk10/lc1/A2 \
          -rise_to   tk10/lc1/B1   -setup $race_margin
set_data_check -clock     tk11/lr   -fall_from  tk11/lc1/A2 \
          -rise_to   tk11/lc1/B1   -setup $race_margin
set_data_check -clock     tk2/lr    -fall_from  tk2/lc1/A2  \
          -rise_to   tk2/lc1/B1    -setup $race_margin

set_max_delay $clk_period -from R0_reg/q -to { R10_reg/d R11_reg/d }
set_max_delay $clk_period -from R10_reg/q -to R2_reg/d
set_max_delay $clk_period -from R11_reg/q -to R2_reg/d

set_min_delay $req_del_min -rise_from  tk0/lr -rise_to  tk0/rr
set_min_delay $req_del_min -rise_from tk10/lr -rise_to toy4/jo
set_min_delay $req_del_min -rise_from tk11/lr -rise_to toy4/jo

set_max_delay $req_del_max -rise_from tk0/lr -rise_to tk0/rr
set_max_delay $req_del_max -rise_from tk10/lr -rise_to toy4/jo
set_max_delay $req_del_max -rise_from tk11/lr -rise_to toy4/jo
```

Note that because an exhaustive list is necessary, there are a total of 26 lines.

Enhanced SDC Constraints

- Module-type specification
  - Extends instance-based specifications to module-types
    - Support top-level module-types, e.g, “linear_control”
    - Support an instance of a module-type, e.g., “linear_control/lc3”
    - Support a module-type in an instance, e.g., “tk10/NAND32”
- “$n” specifier
  - This parameter would allow the nth module type to be repeated across multiple parameters as above right
  - This would result in the following timing constraints shown to the below right

```
set_data_check -clock     linear_control/lr    -fall_from  \
    $1/lc1/A2  -rise_to   $1/lc1/B1    -setup $race_margin
```

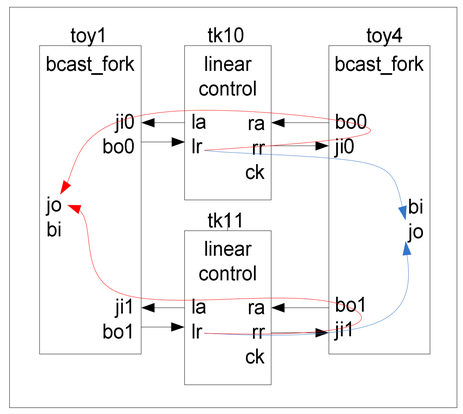
```
set_data_check -clock     tk0/lr    -fall_from  tk0/lc1/A2  \
          -rise_to   tk0/lc1/B1    -setup $race_margin
set_data_check -clock     tk10/lr   -fall_from  tk10/lc1/A2  \
          -rise_to   tk10/lc1/B1    -setup $race_margin
set_data_check -clock     tk11/lr   -fall_from  tk11/lc1/A2  \
          -rise_to   tk11/lc1/B1    -setup $race_margin
set_data_check -clock     tk2/lr    -fall_from  tk2/lc1/A2  \
          -rise_to   tk2/lc1/B1    -setup $race_margin
```

Enhanced SDC Constraints

```
set_max_delay $clk_period –from  \
    linear_control/lr -to bcast_fork/jo
```

- “-not_through” parameter
  - By using the above constraint, both the red and blue paths would be activated, while only the blue was desired
  - “-not_through” parameter disables timing paths through specified pins
  - If the below constraint was specified, only the blue path would be constrained

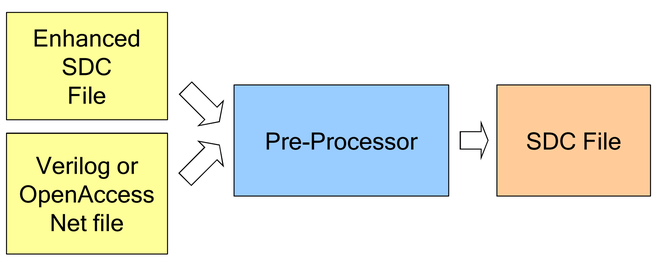
```
set_max_delay $clk_period –from  \
    linear_control/lr -to bcast_fork/jo
```



Conclusions

An Enhancement Processor

- A simple enhancement to the SDC constraints allows for template-based constraints to be created
  - These enhanced constraints are fed into a preprocessor along with the verilog file of the design.
  - The preprocessor generates the desired sdc constraints
  - The enhanced sdc method does not interfere with current SDC flow



Enhanced Constraints

```
create_clock -period $clk_period -waveform "0 $clk_phase"  \
      linear_control/lr

set_disable_timing -from A2 -to Y  linear_control
set_disable_timing -from B1 -to Y  linear_control
set_disable_timing -from B -to Y  [find -hier cell *c_element1]
set_disable_timing -from B -to Y  [find -hier cell *c_element2]
set_disable_timing -from A0 -to Y  linear_control/lc3
set_disable_timing -from B0 -to Y  linear_control/lc3
set_disable_timing -from A1 -to Y  linear_control/lc1

set_data_check -clock linear_control/lr -fall_from  $1/lc1/A2  \
      -rise_to   $1/lc1/B1 -setup $race_margin

set_min_delay $req_del_min -rise_from tk0/lr -rise_to tk0/rr
set_min_delay $req_del_min -rise_from linear_control/lr \
      -not_through  linear_control/lr -rise_to bcast_fork/jo

set_max_delay $req_del_max -rise_from tk0/lr -rise_to tk0/rr
set_max_delay $req_del_max -rise_from linear_control/lr \
      -not_through linear_control/la -rise_to bcast_fork/jo
```

- A Comparison of original constraints to the enhanced constraints
  - Instead of 26 constraints, only 13 constraints are needed
  - As designs grow, only a small increase of constraints are needed
  - These constraints, upon being processed, result in the same timing rules as the original constraints

Summary

- Asynchronous circuits cannot be widely used without Static Timing Analysis
- Commercial timing verification tools provide limited support for asynchronous circuits
  - Hand-customized timing constraints are necessary
  - Any generalization is extremely cumbersome
- The Enhanced SDC pre-processor was developed to easily enhance the current process flow
  - The enhanced SDC constraints are fed into a pre-processor
  - The pre-processor outputs the desired sdc constraints