SystemVerilog Assertions for Clock-Domain-Crossing Data Paths

Don Mills
Microchip Technology Inc.
Outline

• Brief Review of CDC Concepts and Issues
• Basics of SystemVerilog Assertions
• Modeling Techniques and Issues using SVA for CDC
• SVA Model for both RTL and GLS
What is Clock Domain Crossing?

- What is a Clock Domain?
  - flip-flops with same clock (clock tree)

- Clock Domain Crossing
  - Data from one clock domain is captured (sampled) in another clock domain
Clock Domain Crossing Issue - Metastability

Violating “setup” or “hold” time of flip-flop B1 can cause metastability

- A signal settling from metastability settles to
  - the old state
  - or the new state

Screen shot from Philip Freidin – www.fpga-faq.com
http://www.fpga-faq.com/FAQ_Pages/0017_Tell_me_about_metastables.htm
Asynchronous Clocks & Metastability Scenario 1

Original Slide from Cliff Cummings Boston SNUG 2008 (Used by permission) – Modifications made to match this presentation.

A_in sampled by A_clk

B_in sampled by B_clk

A_in

B_in

B_out

Clk_A

Clk_B

A1

B1

Clk_A

Clk_B

B_clk

A_clk

B_out

All outputs are known in this cycle

Clocked B_in signal is initially metastable – settles to either 0 or 1

Original Slide from Cliff Cummings Boston SNUG 2008 (Used by permission) – Modifications made to match this presentation.
Asynchronous Clocks & Metastability Scenario 2

B2 filters the metastable state of B1 from propagating – assumes B1 settles within one clock cycle.

Original Slide from Cliff Cummings Boston SNUG 2008 (Used by permission) – Modifications made to match this presentation.
An Assertion Property for CDC

property CDC_prop1;
    logic v_temp;
    @(posedge Clk_A) ($changed(A_in), v_temp = A_in) |=> ($changed(B1_in) && (B1_in === v_temp));
    ##1 @ (posedge Clk_B) ($changed(B1_in) && (B1_in === v_temp)) [2:3] ($changed(B_out) && (B_out === v_temp));
endproperty: CDC_prop1
Map Assertion to Wave – view 1 (passes)

@(posedge Clk_A)
($\text{changed}(A\_in), \ v\_temp = A\_in)$

$\implies$

($\text{changed}(B1\_in) \land (B1\_in == v\_temp))$

###1 //switch clk domains

@(posedge Clk_B)
($\text{changed}(B1\_in) \land (B1\_in == v\_temp))$

###[2:3]

($\text{changed}(B1\_out) \land (B1\_out == v\_temp))$

Test B1_in at output of A1 in A_clk domain
Map Assertion to Wave – view 2 (fails)

@ (posedge Clk_A)
($changed(A_in),
 v_temp = A_in)
|=>
((($changed(B1_in) &&
   (B1_in === v_temp)))

##1 //switch clk domains

@ (posedge Clk_B)
($changed(B1_in) &&
 (B1_in === v_temp))

##[2:3] ($changed(B1_out) &&
   (B1_out === v_temp));

Test B1_in at output of A1 in A_clk domain

Testing $changed of input to B1 in B_clk domain is a cycle late
Why does this Property fail for CDC?

Cannot remain testing in Clk_A domain while actual data has moved onto the next clock domain

```
logic v_temp;
 @(posedge Clk_A) ($changed(A_in), v_temp = A_in) |=> ($changed(B1_in) && (B1_in == v_temp)) ##1
 @(posedge Clk_B) ($changed(B1_in) && (B1_in == v_temp)) ##[2:3] ($changed(B_out) && (B_out == v_temp));
endproperty: CDC_prop1
```
A correct (RTL) Assertion Property for CDC

Changes to Property
1. overlapping implication operator
2. no test after implication operator
3. use $|$ for clock transition

```verilog
property CDC_prop1;
    logic v_temp;
    @(posedge Clk_A) ($changed(A_in), v_temp = A_in) $|$;
    @(posedge Clk_B) ($changed(B1_in) && (B1_in === v_temp))
    ##[2:3] ($changed(B_out) && (B_out === v_temp));
endproperty: CDC_prop1
```
Correctly working CDC Assertion

@ (posedge Clk_A)
  ($changed(A_in),
   v_temp = A_in)
|->

@ (posedge Clk_B)
  ($changed(B1_in) &&
   (B1_in === v_temp))
##[2:3]

($changed(B1_out) &&
  (B1_out === v_temp));
Full RTL test results – rising and falling, 2 and 3 clk delay

NOTE: Final $changed is tested one cycle after last transition

All conditions passed
Gate Level Sim (GLS) can fail with RTL assertion

Assertion sampling does not account for GLS propagation

This passes in RTL because B1_in changes immediately on Clk_A

Exaggerated Clk to Q GLS delay

@ (posedge Clk_B) ($changed(B1_in) && (B1_in === v_temp))
A Property that works for both RTL and GLS

```
property CDC_prop1;
    logic v_temp;
    @(posedge Clk_A) ($changed(A_in), v_temp = A_in) |->
    @(posedge Clk_B)
        ##[0:1] ($changed(B1_in) && (B1_in === v_temp))
        ##[2:3] ($changed(B_out) && (B_out === v_temp));
endproperty: CDC_prop1
```

The ##[0:1] allows for immediate testing in Clk_B domain for RTL or for a Clk_B to occur during GLS propagation of flip-flop A1
Passing GLS test

\[
\begin{align*}
\@ (\text{posedge Clk\_A}) \\
\quad (\$\text{changed}(A\_in), \ v\_temp = A\_in) \\
\rightarrow
\end{align*}
\]

\[
\begin{align*}
\@ (\text{posedge Clk\_B}) \\
\quad \#[0:1] \\
\quad \quad (\$\text{changed}(B1\_in) \ \&\& \ (B1\_in === v\_temp)) \\
\quad \#[2:3] \\
\quad \quad (\$\text{changed}(B1\_out) \ \&\& \ (B1\_out === v\_temp));
\end{align*}
\]
Conclusions

• Lots of papers on CDC in Google land

• Not so many papers on SV Assertions

• This paper presents an SV Assertion that works for both RTL and GLS

• This SV Assertion can be extended to support loop back to original clock domain