

Inter-Standard Issues with VITAL

Ravi Kumar S V
Texas Instruments India (Pvt) Ltd.
150/1 Infantry Road
BANGALORE.INDIA -560 001

Abstract

This paper presents some of the Inter-standard related issues impeding VITAL standardization efforts for fast and accurate simulations and ease of modeling. The issues described in this paper are as follows:

1. Skew checks for Differential and Phase detector cells
2. PATHPULSE implementation issue related to VHDL'87/'93
3. Annotation definition when a "generate" statement is used in VHDL
4. Labels for big string names as part of conditional path annotation
5. Logic value system for representing strengths

All the issues are covered with some realistic examples and the current work arounds of implementation. The paper finally gives the extensions and features required to SDF constructs and VHDL language to address these issues in an elegant way.

1 Introduction

VITAL based flows follow the following standards:

- OVI SDF2.1 for SDF annotation and timing exchange mechanism
- IEEE 1076 VHDL'87 language for timing/primitive package implementation and for library/netlist build.
- IEEE 1164 MVL9 logic value system for library build/netlist resolution

1.1 Why VITAL based on OVI SDF2.1 for timing interface?

Most of the EDA Vendor tools are yet to catch up to OVI SDF2.1. The intention of VITAL is for the EDA tools to catch up with SDF trend/extensions because of ASIC vendor needs for timing accurate simulations with simulation acceleration addressed. The new SDF 3.0 specification addresses many of critical needs of ASIC.

1.2 Why VITAL based on VHDL'87 and not on VHDL'93?

VHDL'93 is not been supported yet by many EDA vendors though it has got very good features of shared variable/delayed process and additional construct support like reject. VHDL'87 is supported by many EDA and ASIC vendors and VITAL needs to catch up with VHDL'93 once the EDA vendors support full VHDL'93 features to resolve some of the unresolved issues.

2 Inter-standard Issues

2.1 Skew checks for Differential and Phase detector cells

The following is the SDF Syntax for Skew Checks between clocks:

SKEW (posedge CLKI) (posedge CLKR)
(0.05)

SKEWCONSTRAINT (posedge CLKR)
(0.05)

SKEWCONSTRAINT is intended for use in forward annotation eg. Synthesis constraints. This is not supported for back-annotation of VITAL generic timing parameters. Also the signal in

SKEWCONSTRAINT specifies the reference signal against which all associated port signals are constrained. Hence it's not suitable for usage when we want to specify a skew between signal pair only. SKEW is intended for use for the skew between clocks due to interconnect in post layout simulation.

As can be seen above we have a single place holder as these constructs are meant to be for different application annotation. But in Phase detector and Differential cells we need to have triple place holder to take care of (min:nom:max) conditions. One can think of using a HOLD TIMINGCHECK for the skew as follows:

```
(TIMINGCHECK
(HOLD CLKI (posedge CLKR)
(1.50:2.32:3.45))
```

and annotate to as hold generic. This can be used as long as there is no hold constraint check between CLKI and CLKR apart from skew check.

2.1.1 Phase detector model

A schematic of phase detector model is shown in Fig 1. Fig 2 depicts the phase alignment process. It essentially involves two steps:

1. When incoming clock(CLKR) and the feedback clock(CLKI) are not in the variable skew limit QZ assumes one value so that it can give an increment signal to the following shift register stage(RL input of delay element)
2. When CLKR and CLKI are within the variable skew limit i.e phase aligned then QZ assumes the opposite value so that it can give a decrement signal to the following shift register stage(RL input of delay element)

2.1.2 VITAL Modeling Aspects

The variable skew between the clocks can be modeled as follows:

1. Declare a generic as part of entity as follows for annotation:

```
thold_CLKR_CLKI_posedge : DelayTypeXX
:= 1.25 ns;
```
2. Delay CLKR as follows(Level-0):

```
-- CLKR_del is the delayed CLKR and
-- declared as internal signal
```

```
CLKR_del <= CLKR_ipd after
thold_CLKR_CLKI_posedge;
```

3. A relevant section of multiprocess style alike model now can look as follows:

```
-- skew detection mechanism between clocks
```

```
skew_det_process : process (CLKR_ipd)
variable QZ_GlitchData : GlitchDataType;
variable tpd_CLKR_QZ_NEW :
TransitionArrayType(tr01 to tr10);
```

```
-----
-- Functionality Section
-----
```

```
CLKR_del <=VitalBUF(CLKR_ipd'delayed
(1.25 ns));
```

```
-----
-- Path Delay Section
-----
```

```
-- As CLKR is delayed we need to adjust the
-- CLKR to QZ pin to pin delay as follows:
```

```
tpd_CLKR_QZ_NEW(tr01) := tpd_CLKR_Q(tr01)
- thold_CLKR_CLKI_posedge;
tpd_CLKR_QZ_NEW(tr10) := tpd_CLKR_Q(tr10)
- thold_CLKR_CLKI_posedge;
```

```
VitalPropagatePathDelay (
OutSignal => QZ,
OutSignalName => "QZ",
OutTemp => QZ_zd,
Paths => (0 => (CLKR_del'last_event,
VitalExtendToFillDelay(tpd_CLKR_QZ_NEW)
TRUE)))
```

```
-- A different way of representation to
-- achieve current level 1 compliance! is
-- as follows:
```

```
-- Paths => (0 => (CLKR_del'last_event,
VitalExtendToFillDelay(DelayType01'(
(tpd_CLKR_QZ(tr01) -
thold_CLKR_CLKI_posedge),
(tpd_CLKR_QZ(tr10) -
thold_CLKR_CLKI_posedge))),
TRUE)),
GlitchData => QZ_GlitchData,
GlitchMode => MessagePlusX,
GlitchKind => OnEvent);
```

```
end process;
```

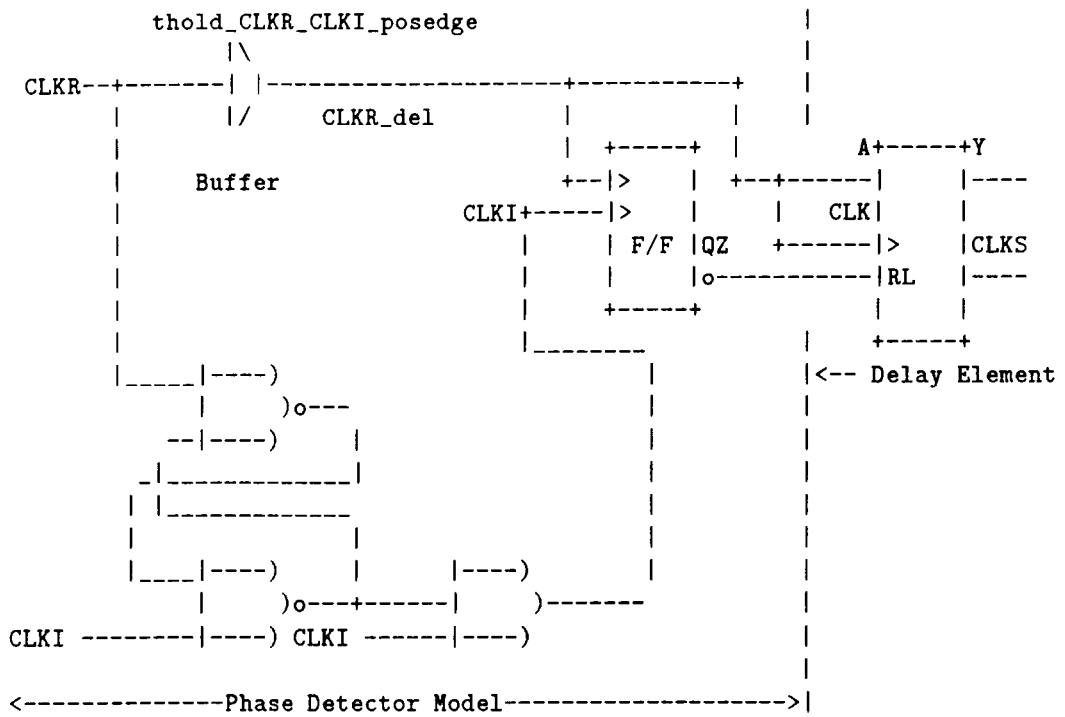


Fig 1 : Phase detector model

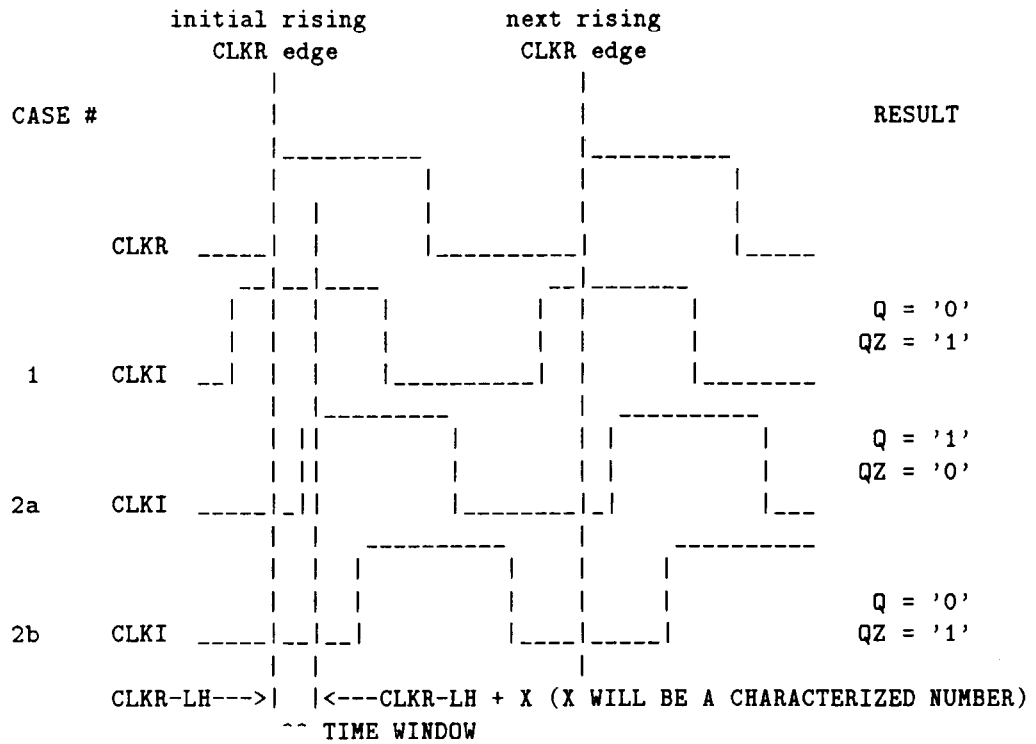


Fig 2 : Phase Alignment in Phase Detector

2.1.3 Proposed solution

The author feels a pin-to-pin hierarchical delay model introduction in VITAL will solve the problem of delay corrections internal to the model to be VITAL Level 1 compliant. But again an hierarchical model support demands for additional generics and an algorithm for hierarchical references and hence an optimization issue.

2.2 PATHPULSE implementation issue related to VHDL'87/'93:

Currently, PATHPULSE/GLOBAL PATHPULSE SDF constructs are not supported by VITAL. In VHDL'87, "transport" prefix to a signal takes care of passage of input pulse to output without any pass/rejection limits to input pulse specified. The implicit inertial prefix to a signal takes care of the pulse suppress mode.

In GLOBAL PATHPULSE % limits are expressed. Also there is a need to use shared variable of VHDL'93. The current implementation of VITAL(Ref 1) has a "GlitchKind" enumeration type for Pulse handling with 4 options viz.

OnEvent, OnDetect, VitalTransport, VitalInertial

The pulse handling options defined by VITAL take care without pulse pass/reject limits specified and handles completely the Spike,Glitch,Pass and Suppress pulse options in a stand alone model without PATHPULSE annotation option. Currently, VITAL deferred support of pulse rejection limits as VITAL is based on VHDL'87 and only in VHDL'93 we have a explicit reject prefix.

The issues here are as follows:

- SDF construct specify pulse rejection w.r.t individual inputs without addressing the interaction of inputs
- Current VITAL glitch detection is performed w.r.t output signal queue.

For example, if we consider PATHPULSE construct, SDF syntax specifies on input pulse limits whereas VHDL'93 syntax is not very clear for X-limit. Hence forth package implementation problem for the future. The author feels extension to VHDL'93 syntax required in this aspect.

SDF Semantics : (PATHPULSE A Y (2 8))

Here 2 and 8 are say in ns time unit.

The first defines the pulse rejection limit i.e the minimum pulse width required for the pulse to pass through to the output; anything smaller doesn't affect the output. The second value denotes the X-limit i.e minimum pulse width necessary to drive the output to a known state; anything smaller causes the output to enter unknown state or is rejected if smaller than than the pulse rejection limit.

```
VHDL Semantics :  
Y <= reject 0.002 ns A transport  
      after .008 ns;
```

This passes pulse with width minimum of 0.002 ns. Here 0.008 ns is propagation delay.

```
Y <= reject 0.002 ns inertial  
      after .008 ns;
```

This suppresses the pulse in the range 0.002 ns and 0.008 ns.

Both semantics doesn't specify how to handle the X-limit?

PATHPULSE support by VITAL is deferred because of the above aspects.

2.3 Annotation definition when a "generate" statement is used in VHDL

For example, how can unique delay values be annotated to each instance port in the following VHDL extract:

```
for I in 1 to 3 generate  
  U1 : IV110 port map (A => A(I), Y => Y(I));  
end generate;
```

Although annotating to arrayed instances seems esoteric today, synthesis vendors have indicated that they will be improving their algorithms to create generate statements where applicable. If this becomes the case, VITAL needs to define annotation definition for such arrayed instances.

2.4 Labels for big string names as part of Conditional Path Annotation

This issue is in addressed to OVI SDF committee by TAG and others and issue resolution

is expected in OVI SDF3.0. The author currently sees the implicit reference of labels being specified by VITAL VPPD call as an issue for simulator vendor independence. To simplify the case the author presents with a simple example. The conditional path strings of characterised cells may run into a number of lines!

Ex. SDF Syntax for CK020 clock distribution macro:

```
( COND CAPDR && !CLK && !SHIFTDR &&
  TCKZ ( IOPATH ISCAN DCLK
    (1.13:1.72:3.59) (0.33:0.52:1.16)))
```

The generic parameter declaration as part of entity is as follows:

```
tpd_ISCAN_DCLK_CAPDR_AN_NT_CLK_
AN_NT_SHIFTDR_AN_TCKZ :
  DelayType01 := (1.371 ns, 1.392 ns);
```

VitalPropagatePathDelay(VPPD) procedural call currently handles as follows:

```
VitalPropagatePathDelay (
  OutSignal => DCLK,
  OutSignalName => "DCLK",
  OutTemp => DCLK_zd,
  Paths => (0 => (CAPDR_ipd'last_event,
    VitalExtendToFillDelay(tpd_ISCAN_DCLK_
      CAPDR_AN_NT_CLK_AN_NT_SHIFTDR_AN_TCKZ),
      (CAPDR_ipd AND (NOT CLK_ipd) AND
        (NOT SHIFTDR_ipd) AND TCKZ_ipd) = '1')),
  GlitchData => DCLK_GlitchData,
  GlitchMode => MessagePlusX,
  GlitchKind => OnEvent);
```

As can be seen above the "PathCondition" in the Paths of VPPD can extend to miles! depending on the conditions we impose on complex cells. Henceforth apart from annotation aspect of putting short label reference VPPD also should be able to handle such short labels or there should be a way of pre-declaration of expressions and pass on to VPPD.

2.5 Logic value system for representing strengths

IEEE 1164 MVL9 logic value system is supported for many of VHDL based simulator design flows.

As part of ieee_std_logic package we can represent only 9 logic values. For open drain/open emitter(ECL) and pull up and down this value system is a limitation. For example, we need to represent pull down and open emitter strengths as weak low 'L' in VSS/Leapfrog VHDL simulator compatible models.

3 Conclusions

Some of the VITAL standard inter-standard issues are highlighted and possible work arounds for some of the issues are given in this paper. The inter-related VITAL standard issues must get resolved for a true accuracy/ performance achievement from modeling/simulation perspective.

Acronyms:

VITAL	VHDL Initiative Toward ASIC Libraries
SDF	Standard Delay Format
VPPD	VitalPropagatePathDelay
ASIC	Application Specific Integrated Circuit
TAG	Technical Action Group(VITAL)

References

- (1) VITAL Model Development Specification version 2.2b
- (2) VHDL IEEE 1987/1992A Language Reference Manual(LRM)
- (3) OVI SDF2.1 Manual/OVI SDF3.0 extensions
- (4) VITAL TAG Issue Resolution Reports
- (5) Phase Detector Model Specification