Test data register access to mission mode DC common-mode voltage settings enables testing and debugging SERDES lanes without the PCB needing a full mission mode ecosystem. Programmable DC common-mode voltages enables more efficient testing of SERDES lanes and enables diagnosis of potential faults with the lane AC coupling capacitors. A programmable DC common-mode voltage can be made from either using a mission mode function or in the case where the PHY supports multiple SERDES protocols, changing the protocol settings via a test data register. Some SERDES protocols have a far end receiver detect capability. This detect capability is performed by the transmitter using a charge pump to change the common-mode voltage from 0 volts to 500mv, for example. A detector is present at the transmitter and detects the rate of charge by setting a timer and sampling the voltage after a fixed time. The rate of charge is changed by the presence of the receiver termination. The impedance of the circuit outside of the transmitter can also affect the detection, for instance, AC coupling cap values that are too small will prevent detection of the receiver. Access to the transmitter's charge pump via a test data register would allow direct control using the TAP of the Vcm. Receivers may have programmable termination resistors which can be used to change the receiver's DC common mode voltage. With a little planning with the IP provider, an IC can be designed to leverage these mission mode common-mode voltages to be accessible via a test data register enabling lower cost test setup on the IC tester or enabling better test generation and diagnostics at the PCB level.

Formatted: Numbering: Continuous

IP Provider PDL routines: set_VCM, get_VCM, getAll_VCM

IC Provider PDL routines: set_VCM_top, get_VCM_top, getAll_VCM_top

Common Grammar:

<voltage> :== <integer> | <integer>.<integer>

<rvoltage> :== <voltage> | "ERROR" | ""

<Tcl Voltage List> :== <rvoltage> { <comma> <rvoltage> }

The grammar specified below uses a "C" style notation to indicate the cases where an iProc returns a value. The return type is indicated in the first position before the iProc keyword if there is a return value for the procedure. Unlike "C", this return value is not used in the instantiation of the iProc hence it is not to be interpreted as a value to parse before the keyword iProc.

IP level procedure grammar

iProc set_VCM { <voltage> }

<rvoltage> iProc get_VCM { }

<Tcl Voltage List> iProc getALL_VCM { }

IC level procedure grammar

iProc set_VCM_top { <representative port> <voltage> }

<rvoltage> iProc get_VCM_top { <representative port> }

<Tcl Voltage List> iProc getALL_VCM_top { <representative port> }

Recommendations

a) It is recommended that the common mode voltage be programmable where possible
b) If Vcm is not programmable then it is recommended that Vcm be set to zero volts for receivers and for transmitters a non-zero voltage compatible with the mission mode operation.

Rules

a) When at least one <twin group> has a programmable common-mode voltage, the set_VCM_top, get_VCM_top and getAll_VCM_top PDL procedures shall be provided.

b) The set_VCM_top, get_VCM_top and getAll_VCM_top PDL procedures shall be associated with the <component name> specified by the <iProcGroup_cmd>.

c) A procedure with the name getAll_VCM_top" shall be a Level-1 PDL procedure that returns a <Tcl Voltage List> representing all the common-mode voltages in millivolts that are available for a <twin group> by specifying the <representative port> as the argument to the procedure.

d) A procedure with the name "get_VCM_top" shall be a Level-1 PDL procedure that returns the <rvoltage> representing the current common-mode voltage in millivolts for a <twin group> by specifying the <representative port> as the argument to the procedure.

e) The procedure with the name "set_VCM_top" shall be a PDL procedure that sets the common-mode voltage for a <twin group> by specifying the <representative port> and the <voltage> respectively as the arguments to the procedure.

f) Any procedure with the name "getAll_VCM_top" and "get_VCM_top" shall return the string "" when a <representative port> does not have a programmable common-mode voltage.

g) An iProcGroup representing procedures for a <twin group> with a programmable common-mode voltage, shall contain set_VCM, get_VCM and getAll_VCM PDL procedures.

h) The procedure with the name "getAll_VCM" shall be a Level-1 PDL procedure that returns <Tcl Voltage List> representing all the common-mode voltage in millivolts that are available for the <object_or_instance> specified by the <iProcGroup_cmd>.

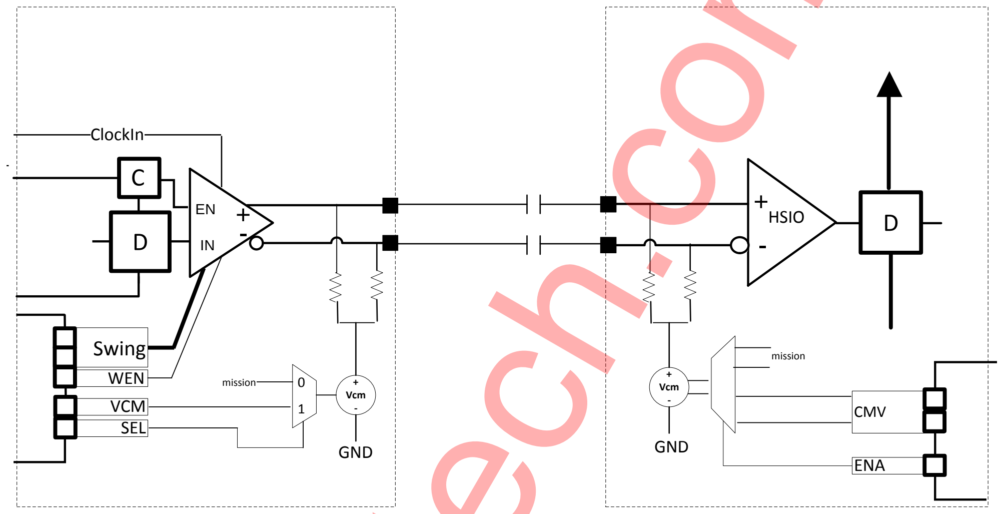
i) The procedure with the name "get_VCM" shall be a Level-1 PDL procedure that returns <rvoltage> representing the current common-mode voltage in millivolts for the <object_or_instance> specified by the <iProcGroup_cmd>.

j) The procedure with the name "set VCM" shall be a PDL procedure that sets the common-mode voltage for a <object_or_instance> by specifying the <voltage> as the argument to the procedure.

k) The set_VCM, get_VCM, ,getAll_VCM, set_VCM_top, get_VCM_top and getAll_VCM_top procedures shall not contain an **iTRST** or **iTMSReset** command at any level of the procedure hierarchy and the procedure definitions shall not have the **-TRSTreset** or **-TMSreset** keywords.

l) <voltage> shall be expressed in millivolts.

m) The get_VCM, getAll_VCM, get_VCM_top and getAll_VCM_top procedures shall return "ERROR" when an error occurs in returning the current common mode voltage.



Vendor A Vendor B

IP Provider A:

BSDL Package

```
-- Excerpts from BSDL Package Body
attribute REGISTER_MNEMONICS of SerdesA : package is
"ONOFF (ON (1) < enable >, " &
"       OFF (0) < off >),  " &
"CMV   ( 0V (1) <  Low power mode >, " &
"      500mV (0) <  Mission mode common-mode voltage>), " &
"SWING (1200mV (0b11) <Boundary Scan Output Swing, mVdfpp>, " &
" 1000mV (0b10), " &
" 800mV (0b01), " &
" 700mV (0b00))" ;


attribute REGISTER_FIELDS of SerdesA : package is
"init_data[5] ( "&
-- TDI
"(SEL [1] IS (4) DEFAULT(ONOFF(OFF) RESETVAL(ONOFF(OFF)) ), "&
"(VCM [1] IS (3) DEFAULT(CMV(0V)) NOPI NOUPD ), "&
"(WEN [1] IS (2) DEFAULT(ONOFF(OFF) RESETVAL(ONOFF(OFF)) ), "&
"(SWING [2] IS (1 DOWNTO 0) DEFAULT(SWING(800mV)) NOPI NOUPD) "&
" )";
```

PDL

# remember that the voltage passed in is in millivolts without units.   Mnemonics should not be passed in here but they can be used within the procedure for ease of use.

```
iProc set_VCM { voltage  } {

if  {$voltage == 0} {
iWrite VCM 0V
}
else {$voltage == 500} {
iWrite VCM 500mV
}
else {
puts "The common-mode voltage $voltage is not supported by this IP\n"
return
}

iWrite SEL ON
iApply
}

# get_VCM returns a voltage.
iProc get_VCM {} {

# Note: the true value of the current VCM is only present
# if SEL is set

set common [iGet -si -mnem VCM]
iApply

# match the strings here with the case of the mnemonics
if  {$common  == "0V"} {
return "0"
}
else if {$common == "500mV" } {
return "500"
}
else {
puts "The common-mode voltage has never been set\n"
return "ERROR"
}

}

iProc getAll_VCM {} {

return "0, 500"
}
```

IP Provider B:

BSDL Package

```
-- Excerpts from BSDL Package Body
attribute REGISTER_MNEMONICS of SerdesB : package is
"EnaDis (Enable   (1) < enable >, " &
"        Disable  (0) < off >), " &
"CV   ( 0V  (0) <  Low power mode >, " &
"     500mV (1)  )," &
"     750mV  (2)  )," &
"       1V   (3)  )";


attribute REGISTER_FIELDS of SerdesB : package is
"init_data[3] ( "&
-- TDI
"(ENA [1] IS (2) DEFAULT(EnaDis(Disable)) RESETVAL(EnaDis(Disable))), "&
"(CMV [2] IS (1 downto 0) DEFAULT(CV(0V)) NOPI NOUPD ) "&
" )";
```

PDL

# remember that the voltage passed in is in millivolts without units. Mnemonics should not be passed in here but they can be used within the procedure for ease of use.

```
iProc set_VCM { voltage } {

if  {$voltage == 0} {
iWrite CMV 0V
}
else {$voltage == 500} {
iWrite CMV 500mV
}
else {$voltage == 750} {
iWrite CMV 750mV
}
else {$voltage == 1000} {
iWrite CMV 1V
}
else {
puts "The common-mode voltage $voltage is not supported by this IP\n"
return
}

iWrite ENA Enable
iApply
}

# get_VCM returns a voltage.
iProc get_VCM {} {

set currVoltage [iGet -si -mnem CMV]
~~iApply~~

# match the strings here with the case of the mnemonics
if  {$currVoltage  == "0V"} {
return "0"
}
else if {$currVoltage == "500mV" } {
```

```
return "500"
}
else if {$currVoltage == "750mV" } {
return "750"
}
else if {$currVoltage == "1V" } {
return "1000"
}
else {
puts "The common-mode voltage has never been set\n"
return "ERROR"
}

}

iProc getAll_VCM {} {

return "0, 500, 750, 1000"
}
```

IC Level

```
attribute REGISTER_ASSEMBLY of MyChip : entity IS
"init_data (" &
"(i1 IS SerdesA), "&
"(i2 IS SerdesB) "&
")";


iProc set_VCM_top { port voltage  } {

if  {$port == "TX_P" } {
iCall i1.set_VCM( $voltage)
}
else {$port == "RX_P"} {
iCall i2.set_VCM( $voltage)
}
else {
puts "Setting the common-mode voltage is not supported on $port\n"
}

}


# get_VCM returns a voltage.
iProc get_VCM_top { port } {

set debug 0

if  {$port == "TX_P" } {
set ret [iCall i1.get_VCM()]
}
else {$port == "RX_P"} {
set ret [iCall i2.get_VCM()]
}
else {
# tools call this procedure to generate tests hence we only desire
# user output information when in some type of debug mode
if { $debug = 1 } {
puts "getting the common-mode voltage is not supported on $port\n"
}
~~return ""~~
set ret ""
}

return $ret
}


iProc getAll_VCM_top { port } {

set debug 0

if  {$port == "TX_P" } {
set ret [iCall i1.getALL_VCM()]
}
else {$port == "RX_P"} {
set ret [iCall i2.getALL_VCM()]
}
else {
# tools call this procedure to generate tests hence we only desire
# user output information when in some type of debug mode
if { $debug = 1 } {
puts "getting the common-mode voltage is not supported on $port\n"
}

return ""
}

return $ret
}
```