

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

16
17
18

19 IP Provider PDL routines: set_VCM, get_VCM, getAll_VCM

20
21 IC Provider PDL routines: set_VCM_top, get_VCM_top, getAll_VCM_top

22
23
24

24 Common Grammar:

25
26
27

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

28
29

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

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

31
32
33

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.

36
37
38

38 IP level procedure grammar

39
40
41

iProc set_VCM { <voltage> }

42 <rvoltage> iProc get_VCM { }

43
44 <Tcl Voltage List> iProc getAll_VCM { }

45
46 IC level procedure grammar

47
48
49

iProc set_VCM_top { <representative port> <voltage> }

50 <rvoltage> iProc get_VCM_top { <representative port> }

51
52 <Tcl Voltage List> iProc getAll_VCM_top { <representative port> }

53
54
55
56

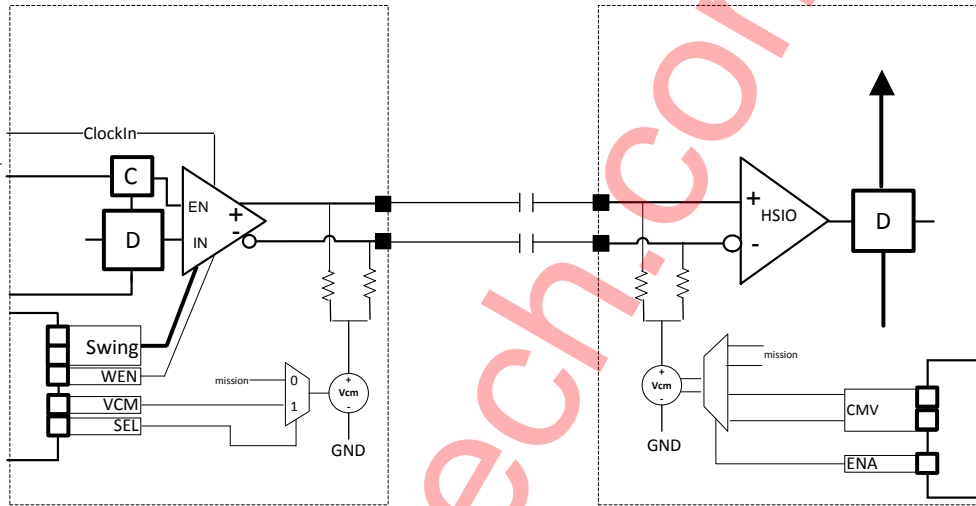
57 Recommendations

- 58
59 a) It is recommended that the common mode voltage be programmable where possible
60 b) If Vcm is not programmable then it is recommended that Vcm be set to zero volts for receivers and for
61 transmitters a non-zero voltage compatible with the mission mode operation.
62

63
64
65 Rules

- 66
67 a) When at least one <twin group> has a programmable common-mode voltage, the set_VCM_top,
68 get_VCM_top and getAll_VCM_top PDL procedures shall be provided.
69
70 b) The set_VCM_top, get_VCM_top and getAll_VCM_top PDL procedures shall be associated with the
71 <component name> specified by the <iProcGroup_cmd>.
72
73 c) A procedure with the name "getAll_VCM_top" shall be a Level-1 PDL procedure that returns a <Tcl
74 Voltage List> representing all the common-mode voltages in millivolts that are available for a <twin
75 group> by specifying the <representative port> as the argument to the procedure.
76
77 d) A procedure with the name "get_VCM_top" shall be a Level-1 PDL procedure that returns the <rvoltage>
78 representing the current common-mode voltage in millivolts for a <twin group> by specifying the
79 <representative port> as the argument to the procedure.
80
81 e) The procedure with the name "set_VCM_top" shall be a PDL procedure that sets the common-mode
82 voltage for a <twin group> by specifying the <representative port> and the <voltage> respectively as the
83 arguments to the procedure.
84
85 f) Any procedure with the name "getAll_VCM_top" and "get_VCM_top" shall return the string "" when a
86 <representative port> does not have a programmable common-mode voltage.
87
88 g) An iProcGroup representing procedures for a <twin group> with a programmable common-mode voltage,
89 shall contain set_VCM, get_VCM and getAll_VCM PDL procedures.
90
91 h) The procedure with the name "getAll_VCM" shall be a Level-1 PDL procedure that returns <Tcl Voltage
92 List> representing all the common-mode voltage in millivolts that are available for the
93 <object_or_instance> specified by the <iProcGroup_cmd>.
94
95 i) The procedure with the name "get_VCM" shall be a Level-1 PDL procedure that returns <rvoltage>
96 representing the current common-mode voltage in millivolts for the <object_or_instance> specified by the
97 <iProcGroup_cmd>.
98
99 j) The procedure with the name "set_VCM" shall be a PDL procedure that sets the common-mode voltage for
100 a <object_or_instance> by specifying the <voltage> as the argument to the procedure.
101
102 k) The set_VCM, get_VCM, ,getAll_VCM, set_VCM_top, get_VCM_top and getAll_VCM_top procedures
103 shall not contain an iTRST or iTMSReset command at any level of the procedure hierarchy and the
104 procedure definitions shall not have the -TRSTreset or -TMSreset keywords.
105
106 l) <voltage> shall be expressed in millivolts.
107
108 m) The get_VCM, getAll_VCM, get_VCM_top and getAll_VCM_top procedures shall return "ERROR"
109 when an error occurs in returning the current common mode voltage.
110
111

112
113
114



Vendor A

Vendor B

115
116
117
118
119

IP Provider A:

BSDL Package

```
125 -- Excerpts from BSDL Package Body
126 attribute REGISTER_MNEMONICS of SerdesA : package is
127 "ONOFF (ON (1) < enable >, " &
128 " OFF (0) < off >), " &
129 "CMV ( 0V (1) < Low power mode >, " &
130 " 500mV (0) < Mission mode common-mode voltage>), " &
131 "SWING (1200mV (0b11) <Boundary Scan Output Swing, mVdfpp>, " &
132 " 1000mV (0b10), " &
133 " 800mV (0b01), " &
134 " 700mV (0b00))" ;
135
136
137 attribute REGISTER_FIELDS of SerdesA : package is
138 "init_data[5] ( "&
139 -- TDI
140 "(SEL [1] IS (4) DEFAULT(OFF(OFF) RESETVAL(OFF(OFF))) ), "&
141 "(VCM [1] IS (3) DEFAULT(CMV(0V)) NOPI NOUPD ), "&
142 "(WEN [1] IS (2) DEFAULT(OFF(OFF) RESETVAL(OFF(OFF))) ), "&
143 "(SWING [2] IS (1 DOWNT0) DEFAULT(SWING(800mV)) NOPI NOUPD) "&
144 " )";
145
146
147 PDL
```

```
148
149 # remember that the voltage passed in is in millivolts without units. Mnemonics should not be passed in here but
150 they can be used within the procedure for ease of use.
151
152 iProc set_VCM { voltage } {
153
154     if { $voltage == 0 } {
155         iWrite VCM 0V
156     }
157     else { $voltage == 500 } {
158         iWrite VCM 500mV
159     }
160     else {
161         puts "The common-mode voltage $voltage is not supported by this IP\n"
162         return
163     }
164
165     iWrite SEL ON
166     iApply
167 }
168
169 # get_VCM returns a voltage.
170 iProc get_VCM {} {
171
172     # Note: the true value of the current VCM is only present
173     # if SEL is set
174
175     set common [iGet -si -nmem VCM]
176     iApply
177
178     # match the strings here with the case of the mnemonics
179     if { $common == "0V" } {
180         return "0"
181     }
182     else if { $common == "500mV" } {
183         return "500"
184     }
185     else {
186         puts "The common-mode voltage has never been set\n"
187         return "ERROR"
188     }
189 }
190
191
192 iProc getAll_VCM {} {
193
194     return "0, 500"
195 }
196
197
198
199 IP Provider B:
200
201
202
203 BSDL Package
204
```

```

205 -- Excerpts from BSDL Package Body
206 attribute REGISTER_MNEMONICS of SerdesB : package is
207 "EnaDis (Enable (1) < enable >, " &
208 "      Disable (0) < off >), " &
209 "CV ( 0V (0) < Low power mode >, " &
210 "   500mV (1) )," &
211 "   750mV (2) )," &
212 "   1V (3) )";
213
214
215 attribute REGISTER_FIELDS of SerdesB : package is
216 "init_data[3] ( "&
217 -- TDI
218 "(ENA [1] IS (2) DEFAULT(EnaDis(Disable)) RESETVAL(EnaDis(Disable))), "&
219 "(CMV [2] IS (1 downto 0) DEFAULT(CV(0V)) NOPI NOUPD ) "&
220 " )";
221
222
223 PDL
224
225 # remember that the voltage passed in is in millivolts without units. Mnemonics should not be passed in here but
226 they can be used within the procedure for ease of use.
227
228 iProc set_VCM { voltage } {
229
230 if {$voltage == 0} {
231 iWrite CMV 0V
232 }
233 else {$voltage == 500} {
234 iWrite CMV 500mV
235 }
236 else {$voltage == 750} {
237 iWrite CMV 750mV
238 }
239 else {$voltage == 1000} {
240 iWrite CMV 1V
241 }
242 else {
243 puts "The common-mode voltage $voltage is not supported by this IP\n"
244 return
245 }
246
247 iWrite ENA Enable
248 iApply
249 }
250
251 # get_VCM returns a voltage.
252 iProc get_VCM {} {
253
254 set currVoltage [iGet -si -mnem CMV]
255 | iApply
256
257 # match the strings here with the case of the mnemonics
258 if {$currVoltage == "0V"} {
259 return "0"
260 }
261 else if {$currVoltage == "500mV" } {

```

```

262 return "500"
263 }
264 else if { $currVoltage == "750mV" } {
265 return "750"
266 }
267 else if { $currVoltage == "1V" } {
268 return "1000"
269 }
270 else {
271 puts "The common-mode voltage has never been set\n"
272 return "ERROR"
273 }
274 }
275 }
276
277 iProc getAll_VCM { } {
278
279 return "0, 500, 750, 1000"
280 }
281
282
283
284
285 IC Level
286
287 attribute REGISTER_ASSEMBLY of MyChip : entity IS
288 "init_data (" &
289 "(i1 IS SerdesA), "&
290 "(i2 IS SerdesB) "&
291 ")";
292
293
294 iProc set_VCM_top { port voltage } {
295
296 if { $port == "TX_P" } {
297 iCall i1.set_VCM( $voltage)
298 }
299 else { $port == "RX_P" } {
300 iCall i2.set_VCM( $voltage)
301 }
302 else {
303 puts "Setting the common-mode voltage is not supported on $port\n"
304 }
305 }
306 }
307
308
309 # get_VCM returns a voltage.
310 iProc get_VCM_top { port } {
311
312 set debug 0
313
314 if { $port == "TX_P" } {
315 set ret [iCall i1.get_VCM()]
316 }
317 else { $port == "RX_P" } {
318 set ret [iCall i2.get_VCM()]

```

```
319 }
320 else {
321 # tools call this procedure to generate tests hence we only desire
322 # user output information when in some type of debug mode
323 if { $debug = 1 } {
324 puts "getting the common-mode voltage is not supported on $port\n"
325 }
326 return ""
327 set ret ""
328 }
329
330 return $ret
331 }
332
333 iProc getAll_VCM_top { port } {
334 set debug 0
335
336 if { $port == "TX_P" } {
337 set ret [iCall i1.getAll_VCM()]
338 }
339 else { $port == "RX_P" } {
340 set ret [iCall i2.getAll_VCM()]
341 }
342 else {
343 # tools call this procedure to generate tests hence we only desire
344 # user output information when in some type of debug mode
345 if { $debug = 1 } {
346 puts "getting the common-mode voltage is not supported on $port\n"
347 }
348
349 return ""
350 }
351 return $ret
352 }
353
354 return $ret
355 }
356
```