

This is a preview - click here to buy the full publication



IEEE

IEC 61523-1

Edition 3.0 2023-10

**INTERNATIONAL
STANDARD**

IEEE Std 1481™



**Delay and power calculation standards –
Part 1: Integrated Circuit (IC) Open Library Architecture (OLA)**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.01, 35.060

ISBN 978-2-8322-7539-9

Warning! Make sure that you obtained this publication from an authorized distributor.

Contents

1. Overview	30
1.1 Scope	30
1.2 Purpose	30
1.3 Introduction	30
1.4 Word usage	31
2. Normative references.....	32
3. Definitions	32
4. Acronyms and abbreviations	40
5. Typographical conventions	41
5.1 Syntactic elements	41
5.2 Conventions	42
6. DPCS flow.....	42
6.1 Overview	42
6.1.1 Procedural interface	44
6.1.2 Global policies and conventions	44
6.2 Flow of control	44
6.3 DPCM—application relationships	44
6.3.1 Technology library	45
6.3.2 Subrule	45
6.4 Interoperability	45
7. Delay calculation language (DCL)	45
7.1 Character set	45
7.2 Lexical elements	45
7.2.1 Whitespace	46
7.2.2 Comments	46
7.2.3 Tokens	46
7.2.4 Header names	56
7.2.5 Preprocessing directives	56
7.3 Context.....	56
7.3.1 Space	56
7.3.2 Plane.....	56
7.3.3 Context operation	56
7.3.4 Library parallelism.....	56
7.3.5 Application parallelism	57
7.4 Data types	57
7.4.1 Base types	57
7.4.2 Native data types	57
7.4.3 Mathematical calculation data types	57
7.4.4 Pointer data types	58
7.4.5 Aggregate data types	58
7.5 Identifiers.....	64
7.5.1 Name spaces of identifiers	64
7.5.2 Storage durations of objects	64
7.5.3 Scope of identifiers	65
7.5.4 Linkages of identifiers	66

7.6 Operator descriptions	66
7.6.1 String prefix operator	66
7.6.2 Explicit string prefix operator	66
7.6.3 Embedded string prefix operator	67
7.6.4 String prefix semantics	67
7.6.5 Assignment operator	67
7.6.6 New operator	67
7.6.7 SCOPE operator(s)	68
7.6.8 Launch operator	69
7.6.9 Purity operator	69
7.6.10 Force operator	70
7.7 Timing propagation	70
7.7.1 Timing checks	71
7.7.2 Test mode operators	71
7.8 Expressions	73
7.8.1 Array subscripting	74
7.8.2 Statement calls	74
7.8.3 General syntax	74
7.8.4 Method statement calls	74
7.8.5 Assign variable reference	75
7.8.6 Store variable reference	75
7.8.7 Mathematical expressions	75
7.8.8 Mathematical operators	76
7.8.9 Discrete math expression	77
7.8.10 INT discrete	78
7.8.11 PINLIST discrete	78
7.8.12 Logical expressions and operators	78
7.8.13 MODE expressions	78
7.8.14 Embedded C code expressions	80
7.8.15 Computation order	81
7.9 DCL mathematical statements	83
7.9.1 Statement names	83
7.9.2 Modifiers	87
7.9.3 Prototypes	89
7.9.4 Statement failure	92
7.9.5 Type definition statements	92
7.9.6 Interfacing statements	93
7.9.7 DCL to C communication	95
7.9.8 Constant statement	95
7.9.9 Calculation statements	96
7.9.10 METHOD statement	98
7.10 Predefined types	99
7.10.1 ACTIVITY_HISTORY_TYPE	99
7.10.2 HISTORY_TYPE	100
7.10.3 LOAD_HISTORY_TYPE	101
7.10.4 CELL_LIST_TYPE	102
7.10.5 TECH_TYPE	102
7.10.6 DELAY_REC_TYPE	102
7.10.7 SLEW_REC_TYPE	103
7.10.8 CHECK_REC_TYPE	103
7.10.9 CCDB_TYPE	103
7.10.10 CELL_DATA_TYPE	103
7.10.11 PCDB_TYPE	103
7.10.12 PIN_ASSOCIATION	104
7.10.13 PATH_DATA_TYPE	104
7.10.14 STD_STRUCT	104

7.11	Predefined variables.....	105
7.11.1	ARGV	105
7.11.2	CONTROL_PARM.....	105
7.12	Built-in function calls	105
7.12.1	ABS.....	106
7.12.2	Complex number components.....	106
7.12.3	EXPAND.....	106
7.12.4	Array functions.....	106
7.12.5	Messaging functions	107
7.13	Tables.....	109
7.13.1	TABLEDEF statement	109
7.13.2	Table visibility rules.....	111
7.13.3	TABLE statement.....	112
7.13.4	LOAD_TABLE statement.....	115
7.13.5	UNLOAD_TABLE statement.....	117
7.13.6	WRITE_TABLE statement	118
7.13.7	ADD_ROW statement.....	118
7.13.8	DELETE_ROW statement.....	119
7.14	Built-in library functions	120
7.14.1	Numeric conversion functions	120
7.14.2	Tech_family functions.....	122
7.14.3	Trigonometric functions.....	123
7.14.4	Context manipulation functions	124
7.14.5	Debug controls.....	126
7.14.6	Utility functions	126
7.14.7	Table functions.....	127
7.14.8	Subrule controls	128
7.15	Library control statements	129
7.15.1	Meta-variables.....	129
7.15.2	TECH_FAMILY.....	129
7.15.3	RULENAME	129
7.15.4	CONTROL_PARM.....	129
7.15.5	SUBRULE statement	129
7.15.6	Path list expansion rules.....	130
7.15.7	SUBRULES statement.....	131
7.15.8	Control file.....	131
7.15.9	TECH_FAMILY statement.....	133
7.15.10	SUBRULE and SUBRULES statements.....	134
7.16	Modeling.....	134
7.16.1	Types of modeling.....	134
7.16.2	Model organization	135
7.16.3	MODELPROC statement.....	136
7.16.4	SUBMODEL statement	137
7.16.5	Modeling statements	139
7.16.6	TEST_BUS statement.....	148
7.16.7	INPUT statement.....	149
7.16.8	OUTPUT statement.....	152
7.16.9	DO statement	153
7.16.10	PROPERTIES statement.....	174
7.16.11	SETVAR statement	175
7.17	Embedded C code.....	176
7.18	Definition of a subrule	176
7.19	Pragma.....	177
7.19.1	IMPORT_EXPORT_TAG.....	177

8. Power modeling and calculation.....	177
8.1 Power overview	177
8.2 Caching state information.....	178
8.2.1 Initializing the state cache.....	178
8.2.2 State cache lifetime	178
8.3 Caching load and slew information	178
8.3.1 Loading the load and slew cache	179
8.3.2 Load and slew cache lifetime.....	179
8.4 Simulation switching events	180
8.5 Partial swing events	180
8.6 Power calculation.....	180
8.7 Accumulation of power consumption by the design.....	182
8.8 Group Pin List syntax and semantics.....	182
8.8.1 Syntax	182
8.8.2 Semantics	183
8.8.3 Example	183
8.9 Group Condition List syntax and semantics	183
8.9.1 Syntax	184
8.9.2 Semantics	184
8.9.3 Example	184
8.10 Sensitivity list syntax and semantics.....	184
8.10.1 Syntax	185
8.10.2 Semantics.....	185
8.10.3 Example	185
8.11 Group condition language.....	185
8.11.1 Syntax	186
8.11.2 Semantics	186
8.11.3 Condition expression operator precedence.....	188
8.11.4 Condition expressions referencing pin states and transitions	188
8.11.5 Semantics of nonexistent pins	189
9. Application and library interaction	189
9.1 behavior model domain	189
9.2 vectorTiming and vectorPower model domains	190
9.2.1 Power unit conversion.....	190
9.2.2 Vector power calculation.....	190
10. Procedural interface (PI).....	191
10.1 Overview	191
10.1.1 DPCM	191
10.1.2 Application.....	192
10.1.3 libdcmlr	192
10.2 Control and data flow	192
10.3 Architectural requirements.....	192
10.4 Data ownership technique.....	193
10.4.1 Persistence of data passed across the PI.....	193
10.4.2 Data cache guidelines for the DPCM.....	193
10.4.3 Application/DPCM interaction.....	194
10.4.4 Application initializes message/memory handling.....	194
10.4.5 Application loads and initializes the DPCM	194
10.4.6 Application requests timing models for cell instances	194
10.5 Model domain issues	194
10.5.1 Model domain selection	194
10.5.2 Model domain determination	195
10.5.3 DPCM invokes application modeling callback functions	195
10.5.4 Application requests propagation delay	195
10.5.5 DPCM calls application EXTERNAL functions.....	196
10.6 Reentry requirements.....	196

10.7 Application responsibilities when using a DPCM	196
10.7.1 Standard Structure rules	196
10.7.2 User object registration	197
10.7.3 Selection of early and late slew values.....	197
10.7.4 Semantics of slew values	198
10.7.5 Slew calculations	198
10.8 Application use of the DPCM	198
10.8.1 Initialization of the DPCM.....	198
10.8.2 Context creation	199
10.8.3 Dynamic linking.....	199
10.8.4 Subrule initialization	200
10.8.5 Use of the DPCM.....	201
10.8.6 Application control.....	201
10.8.7 Application execution	201
10.8.8 Termination of DPCM.....	201
10.9 DPCM library organization.....	202
10.9.1 Multiple technologies.....	202
10.9.2 Model names.....	202
10.9.3 DPCM error handling.....	202
10.10 C level language for EXPOSE and EXTERNAL functions.....	203
10.10.1 Integer return code	203
10.10.2 The Standard Structure pointer.....	203
10.10.3 Result structure pointer	203
10.10.4 Passed arguments	203
10.10.5 DCL array indexing.....	204
10.10.6 Conversion to C data types	204
10.10.7 include files.....	204
10.11 PIN and BLOCK data structure requirements.....	205
10.12 DCM_STD_STRUCT Standard Structure	206
10.12.1 Alternate semantics for Standard Structure fields	208
10.12.2 Reserved fields.....	209
10.12.3 Standard Structure value restriction	209
10.13 DCMTransmittedInfo structure.....	209
10.14 Environment or user variables	209
10.15 Procedural interface (PI) functions summary	209
10.15.1 Expose functions.....	209
10.15.2 External functions	217
10.15.3 Deprecated functions.....	220
10.16 Implicit functions.....	222
10.16.1 libdcmlr.....	222
10.16.2 Run-time library utility functions.....	223
10.16.3 Memory control functions.....	223
10.16.4 Message and error control functions	224
10.16.5 Calculation functions	225
10.16.6 Modeling functions	225
10.17 PI function table description	225
10.17.1 Arguments	226
10.17.2 DCL syntax	227
10.17.3 C syntax	227

10.18 PI function descriptions	227
10.18.1 Interconnect loading related functions	227
10.18.2 Interconnect delay related functions	234
10.18.3 Functions accessing netlist information	237
10.18.4 Functions exporting limit information	246
10.18.5 Functions getting/setting model information	247
10.18.6 Functions importing instance name information	260
10.18.7 Process information functions	263
10.18.8 Miscellaneous standard interface functions	265
10.18.9 Power-related functions	274
10.19 Application context	282
10.19.1 pathData association	282
10.20 Application and library interaction	282
10.20.1 behavior model domain	283
10.20.2 vectorTiming and vectorPower model domains	283
10.20.3 Power unit conversion	284
10.20.4 Vector power calculation	284
10.21 Parasitic analysis	285
10.21.1 Assumptions	285
10.21.2 Parasitic networks	285
10.21.3 Basic definitions	285
10.21.4 Parasitic element data structure	287
10.21.5 Coordinates	291
10.21.6 Parasitic subnets	291
10.21.7 Pin parasitics	298
10.21.8 Modeling internal nodes	301
10.21.9 Load and interconnect models	303
10.21.10 Obtaining parasitic networks	307
10.21.11 Persistent storage of load and interconnect models	308
10.21.12 Calculating effective capacitances and driving resistances	311
10.21.13 Parasitic estimation	314
10.21.14 Threshold voltages	319
10.21.15 Obtaining aggressor window overlaps	320
10.22 Noise analysis	327
10.22.1 Types of noise	328
10.22.2 Noise models	329
10.22.3 Noise waveforms	332
10.22.4 Noise network models	338
10.22.5 Calculating composite noise at cell inputs	344
10.22.6 Calculating composite noise at cell outputs	347
10.22.7 Setting noise budgets	351
10.22.8 Reporting noise violations	352
10.23 Delay and slew calculations for differential circuits	354
10.23.1 Sample figures	354
10.23.2 appGetArrivalOffsetsByName	356
10.23.3 API extensions for function modeling	357
10.23.4 Explicit APIs for user-defined primitives	364
10.23.5 APIs for hierarchy	366
10.23.6 Built-in APIs for function modeling	366
10.23.7 API Extensions for VECTOR modeling	367
10.23.8 APIs for XWF	368
10.23.9 Extensions and changes to voltages and temperature APIs	372
10.23.10 Operating conditions	374
10.23.11 On-chip process variation	377
10.23.12 Accessing properties and attributes	383
10.23.13 APIs for attribute within a PIN object	404

10.23.14 Connectivity	412
10.23.15 Control of timing arc existence and state	414
10.23.16 Modeling cores.....	419
10.23.17 Default pin slews and interface version calls	424
10.23.18 API to access library required resources	426
10.23.19 Resource types	427
10.23.20 Library extensions for phase locked loop processing.....	428
10.23.21 API definitions for external conditions	429
10.23.22 Extensions for listing pins.....	433
10.23.23 Memory BIST mapping	434
10.23.24 dpcmGetCellTestProcedure.....	436
10.24 Interconnect delay calculation intraface	437
10.24.1 Control and data flows	437
10.24.2 Model generation functions.....	438
10.24.3 Calculation functions	440
10.24.4 Cell calculation functions.....	441
10.24.5 <i>ICM initialization</i>	446
10.25 DCL run-time support.....	453
10.25.1 Array manipulation functions.....	453
10.25.2 Memory management	457
10.25.3 Structure manipulation functions	457
10.25.4 Initialization functions	462
10.26 Calculation functions	473
10.26.1 delay.....	473
10.26.2 slew	474
10.26.3 check	474
10.27 Modeling functions	476
10.27.1 modelSearch.....	476
10.27.2 Mode operators	478
10.27.3 Arrival time merging.....	479
10.27.4 Edge propagation communication to the application	480
10.27.5 Edge propagation communication to the DPCM	482
10.27.6 newTimingPin	483
10.27.7 newDelayMatrixRow	483
10.27.8 newNetSinkPropagateSegments	484
10.27.9 newNetSourcePropagateSegments.....	486
10.27.10 newPropagateSegment	487
10.27.11 newTestMatrixRow	488
10.27.12 newAltTestSegment	489
10.27.13 Interactions between interconnect modeling and modeling functions	489
10.28 Deprecated functions	490
10.28.1 Parasitic handling.....	490
10.28.2 Array manipulation functions.....	499
10.28.3 Memory management	501
10.28.4 Initialization functions	504
10.29 Standard Structure (std_stru.h) file.....	515
10.30 Standard macros (std_macros.h) file	535
10.31 Standard interface structures (dcmintf.h) file	543
10.32 Standard loading (dcmload.h) file	547
10.33 Standard debug (dcmdebug.h) file	551
10.34 Standard array (dcmgarray.h) file	578
10.35 Standard user array defines (dcmuarray.h) file.....	582
10.36 Standard platform-dependency (dcmpltfm.h) file	586
10.37 Standard state variables (dcmstate.h) file.....	592

11. Parasitics.....	595
11.1 Introduction.....	596
11.2 Targeted applications for SPEF.....	596
11.3 SPEF specification.....	596
11.3.1 Grammar.....	596
11.3.2 Escaping rules.....	598
11.3.3 File syntax.....	598
11.3.4 Comments.....	604
11.3.5 File semantics.....	604
11.4 Examples.....	623
11.4.1 Basic *D_NET file.....	623
11.4.2 Basic *R_NET file.....	626
11.4.3 *R_NET with poles and residues plus name mapping.....	627
11.4.4 *D_NET with triplet par_value.....	628
11.4.5 *R_NET with poles and residues plus triplet par_value.....	631
11.4.6 Merging SPEF files.....	633
11.4.7 A SPEF file header section with *VARIATION_PARAMETERS definition.....	637
11.4.8 CAP and RES statements with sensitivity information in a SPEF file.....	637
Annex A (informative) Implementation requirements.....	638
Annex B (informative) Participants.....	640

Figures

Figure 1—High-level DPCS architecture linkage structure	43
Figure 2—Function graph form.....	135
Figure 3—DPCM/application procedural interface.....	192
Figure 4—PIN and PINLIST.....	205
Figure 5—Parallel drivers example.....	239
Figure 6—Subnet node mapping.....	294
Figure 7—Differential buffer chain.....	354
Figure 8—Timing models for a differential buffer chain	355
Figure 9—Arrival offset for differential signals.....	355
Figure 10—Priority operation	360
Figure 11—Precedence.....	361
Figure 12—Strand ranges.....	362
Figure 13—Various methods of using XWF to model waveforms for slew computation.....	368
Figure 14—Propagation of XWF “handles” by application.....	369
Figure 15—Application, CCM, and ICM control and data flows	438
Figure 16—Clock separation.....	475
Figure 17—Bias calculation	476
Figure 18—Clock pulse width.....	476
Figure 19—Sample MODELPROC results.....	486
Figure 20—Additional MODELPROC results.....	486
Figure 21—Capacitance value example	491
Figure 22—Equation for poles and residues	492
Figure 23—Example RC network	498
Figure 24—SPEF targeted applications.....	596

Tables

Table 1—Keywords.....	46
Table 2—DCL predefined references to Standard Structure fields.....	49
Table 3—DCL compiler generated predefined identifiers.....	52
Table 4—Edge types and conversions.....	54
Table 5—Propagation mode conversions.....	54
Table 6—Calculation mode conversions.....	54
Table 7—TEST_TYPE conversions.....	55
Table 8—Purity operator.....	70
Table 9—Timing resolution modes.....	70
Table 10—Test mode operators table.....	71
Table 11—Mathematical operators.....	77
Table 12—Logical operators.....	78
Table 13—Mathematical operator precedence (high to low).....	81
Table 14—Logical operator precedence (high to low).....	81
Table 15—Type definition for ACTIVE_HISTORY_TYPE.....	99
Table 16—Permitted activityCode values.....	100
Table 17—Type definition for HISTORY_TYPE.....	100
Table 18—Rule history info message types.....	101
Table 19—Table History inform message types.....	101
Table 20—Permitted kind values.....	101
Table 21—LOAD_HISTORY_TYPE.....	101
Table 22—CELL_LIST_TYPE.....	102
Table 23—TECH_TYPE.....	102
Table 24—DELAY_REC_TYPE.....	102
Table 25—SLEW_REC_TYPE.....	103
Table 26—CHECK_REC_TYPE.....	103
Table 27—CCDB_TYPE.....	103
Table 28—CELL_DATA_TYPE.....	103
Table 29—CCDB_TYPE.....	104
Table 30—PIN_ASSOCIATION.....	104
Table 31—PATH_DATA_TYPE.....	104
Table 32—STD_STRUCT.....	105
Table 33—ARGV.....	105
Table 34—CONTROL_PARM.....	105
Table 35—Library function floor.....	121
Table 36—Library function ifloor.....	121
Table 37—Library function ceil.....	121
Table 38—Library Function iceil.....	121
Table 39—Library function rint.....	121
Table 40—Library function round.....	122
Table 41—Library function trunc.....	122
Table 42—Library function itrunc.....	122
Table 43—Library function map_tech_family.....	122
Table 44—Library function current_tech_type.....	122
Table 45—Library function subrule_tech_type.....	123
Table 46—Library function subrule_tech_type.....	123
Table 47—Library function get_technology_list.....	123
Table 48—Library function cos.....	123
Table 49—Library function sin.....	123
Table 50—Library function tan.....	124
Table 51—Library function new_plane.....	124
Table 52—Library function get_plane_name.....	124
Table 53—Library function get_space_name.....	124
Table 54—Library function get_max_spaces.....	125

Table 55—Library function <code>get_max_planes</code>	125
Table 56—Library function <code>get_space_coordinate</code>	125
Table 57—Library function <code>get_plane_coordinate</code>	125
Table 58—Library function <code>set_busy_wait</code>	126
Table 59—Library function <code>change_debug_level</code>	126
Table 60—Library function <code>get_caller_stack</code>	126
Table 61—Library function <code>GET_LOAD_HISTORY</code>	126
Table 62—Library function <code>GET_CELL_LIST</code>	127
Table 63—Library function <code>GET_ROW_COUNT</code>	127
Table 64—Library function <code>STEP_TABLE</code>	127
Table 65—Library function <code>GET_LOAD_PATH</code>	128
Table 66—Library function <code>GET_RULE_NAME</code>	128
Table 67—Library function <code>ADD_RULE</code>	128
Table 68—Data type clause.....	143
Table 69—Arc data types	144
Table 70—Validity of predefined identifiers for <code>STORE</code> clause	152
Table 71—Logic operators (valid for behavior, vectorTiming, and vectorPower model domains).....	157
Table 72—Logical equivalence operators (valid for behavior model domain)	157
Table 73—Unary bitwise operators (valid for behavior, vectorTiming, and vectorPower model domains).....	157
Table 74—Binary bitwise operators (valid for behavior, vectorTiming, and vectorPower model domains).....	158
Table 75—Binary operators (valid for behavior model domain).....	158
Table 76—Node primitives for control operators (valid for behavior model domain).....	158
Table 77—Node primitives for edge operators (continued) (valid for behavior, vectorTiming, and vectorPower model domains).....	159
Table 78—Node primitives for precedence control operators (valid for behavior model domain).....	159
Table 79—Node primitives for constant operators (valid for behavior, vectorTiming, vectorPower model domains).....	160
Table 80—Node primitives for user-defined operators	160
Table 81—Node primitives for miscellaneous operators.....	160
Table 82—Binary reduction operators.....	161
Table 83—Bitwise reduction operators	162
Table 84—Logical reduction operators	163
Table 85—Array of bits operators	164
Table 86—Edge operators	165
Table 87—Higher function nodes.....	166
Table 88—Constant value nodes	168
Table 89—Miscellaneous operators	169
Table 90—User-defined operators.....	169
Table 91—Valid modifier enumerations for given node primitive operators	170
Table 92—Syntax for a <code>GroupPinString</code>	183
Table 93—Syntax for a <code>GroupConditionString</code>	184
Table 94—Syntax for a <code>SensitivityPinString</code>	185
Table 95—Syntax for a <code>condition_expression</code>	186
Table 96— <code>PinName_Identifier</code> semantics	187
Table 97— <code>PinName_Level</code> semantics.....	187
Table 98— <code>PinName_State</code> semantics	187
Table 99—Condition expression operators.....	188
Table 100—Interaction between multiple technologies and application	202
Table 101—Return code most significant byte.....	203
Table 102—Return code least significant bytes	203
Table 103—Data types defined in DCL and C	204
Table 104—Header files.....	205
Table 105—Predefined macro names	206
Table 106—Alternate semantics for Standard Structure fields.....	208

Table 107—Expose functions.....	209
Table 108—External functions.....	217
Table 109—Deprecated functions.....	220
Table 110—libdcmlr functions.....	222
Table 111—Module control functions.....	223
Table 112—Memory control functions.....	223
Table 113—Message and error control functions.....	224
Table 114—Calculation functions.....	225
Table 115—Modeling functions.....	225
Table 116—PI function table example.....	226
Table 117—Standard Structure field semantics.....	226
Table 118—appGetTotalLoadCapacitanceByPin.....	227
Table 119—appGetTotalLoadCapacitanceByName.....	228
Table 120—appGetTotalPinCapacitanceByPin.....	228
Table 121—appGetTotalPinCapacitanceByName.....	229
Table 122—appGetSourcePinCapacitanceByPin.....	229
Table 123—appGetSourcePinCapacitanceByName.....	230
Table 124—dpcmGetDefCellSize.....	230
Table 125—appGetCellCoordinates.....	230
Table 126—appGetCellOrientation.....	231
Table 127—dpcmGetEstLoadCapacitance.....	232
Table 128—dpcmGetEstWireCapacitance.....	232
Table 129—dpcmGetEstWireResistance.....	233
Table 130—dpcmGetPinCapacitance.....	233
Table 131—dpcmGetCellIOlists.....	234
Table 132—appGetRC.....	234
Table 133—dpcmGetDelayGradient.....	235
Table 134—dpcmGetSlewGradient.....	235
Table 135—dpcmGetEstimateRC.....	236
Table 136—dpcmGetDefPortSlew.....	236
Table 137—dpcmGetDefPortCapacitance.....	237
Table 138—appGetNumDriversByPin.....	237
Table 139—appGetNumDriversByName.....	237
Table 140—appForEachParallelDriverByPin.....	238
Table 141—appForEachParallelDriverByName.....	240
Table 142—appGetNumPinsByPin.....	241
Table 143—appGetNumPinsByName.....	242
Table 144—appGetNumSinksByPin.....	242
Table 145—appGetNumSinksByName.....	242
Table 146—dpcmAddWireLoadModel.....	243
Table 147—dpcmGetWireLoadModel.....	244
Table 148—dpcmGetWireLoadModelForBlockSize.....	245
Table 149—appGetInstanceCount.....	245
Table 150—dpcmGetCapacitanceLimit.....	246
Table 151—dpcmGetSlewLimit.....	246
Table 152—dpcmGetXovers.....	247
Table 153—dpcmGetFunctionalModeArray.....	247
Table 154—dpcmGetBaseFunctionalMode.....	248
Table 155—appGetCurrentFunctionalMode.....	249
Table 156—dpcmGetControlExistence.....	249
Table 157—dpcmSetLevel.....	250
Table 158—dpcmGetLibraryAccuracyLevelArrays.....	252
Table 159—dpcmSetLibraryAccuracyLevel.....	252
Table 160—dpcmGetExposePurityAndConsistency.....	253
Table 161—DCM_Purity.....	253
Table 162—DCM_Consistency.....	254

Table 163—dpcmGetRailVoltageArray.....	254
Table 164—dpcmGetBaseRailVoltage	254
Table 165—appGetCurrentRailVoltage.....	255
Table 166—dpcmGetWireLoadModelArray	255
Table 167—dpcmGetBaseWireLoadModel.....	256
Table 168—appGetCurrentWireLoadModel	256
Table 169—dpcmGetBaseTemperature.....	257
Table 170—dpcmGetBaseOpRange.....	257
Table 171—dpcmGetOpRangeArray	258
Table 172—appGetCurrentTemperature.....	258
Table 173—appGetCurrentOpRange.....	259
Table 174—dpcmGetTimingStateArray.....	259
Table 175—appGetCurrentTimingState	260
Table 176—dpcmGetCellList.....	261
Table 177—appGetCellName.....	262
Table 178—appGetHierPinName	262
Table 179—appGetHierBlockName.....	263
Table 180—appGetHierNetName	263
Table 181—dpcmGetThresholds.....	264
Table 182—appGetThresholds	264
Table 183—appGetExternalStatus.....	265
Table 184—appGetVersionInfo.....	265
Table 185—appGetResource	266
Table 186—dpcmGetRuleUnitToSeconds.....	266
Table 187—dpcmGetRuleUnitToOhms	267
Table 188—dpcmGetRuleUnitToFarads	267
Table 189—dpcmGetRuleUnitToHenries.....	268
Table 190—dpcmGetRuleUnitToWatts.....	268
Table 191—dpcmGetRuleUnitToJoules	269
Table 192—dpcmGetTimeResolution	269
Table 193—dpcmGetParasiticCoordinateTypes	270
Table 194—dpcmIsSlewTime	270
Table 195—dpcmDebug.....	271
Table 196—dpcmGetVersionInfo.....	271
Table 197—dpcmHoldControl	272
Table 198—dpcmFillPinCache.....	272
Table 199—dpcmFreePinCache	273
Table 200—appRegisterCellInfo	273
Table 201—dpcmGetCellPowerInfo	274
Table 202—dpcmGetCellPowerWithState	275
Table 203—dpcmGetAETCellPowerWithSensitivity	276
Table 204—Integer LSB example	276
Table 205—Mask encoding.....	276
Table 206—dpcmGetPinPower	277
Table 207—dpcmAETGetSettlingTime	277
Table 208—dpcmAETGetSimultaneousSwitchTime	278
Table 209—dpcmGroupGetSettlingTime	278
Table 210—dpcmGroupGetSimultaneousSwitchTime.....	279
Table 211—dpcmCalcPartialSwingEnergy	279
Table 212—dpcmSetInitialState.....	280
Table 213—dpcmFreeStateCache	281
Table 214—appGetStateCache.....	281
Table 215—dpcmGetNetEnergy	282
Table 216—parasiticElement structure.....	287
Table 217—Parasitic element variables.....	288
Table 218—DCM_ElementTypes	289

Table 219—Node variables	289
Table 220—Value variables.....	290
Table 221—Coordinate structure.....	291
Table 222—parasiticSubnet structure.....	291
Table 223—DCM_NodeTypes	292
Table 224—dpcmCreateSubnetStructure	295
Table 225—dpcmGetDefaultInterconnectTechnology	297
Table 226—dpcmScaleParasitics.....	297
Table 227—dpcmGetSinkPinParasitics.....	300
Table 228 Table 228—dpcmGetSourcePinParasitics	300
Table 229—dpcmGetPortNames	301
Table 230—Application timing arcs.....	302
Table 231—dpcmBuildLoadModels	304
Table 232—dpcmBuildInterconnectModels.....	305
Table 233—appGetInterconnectModels.....	305
Table 234—appGetLoadModels.....	306
Table 235—appGetParasiticNetworksByPin.....	307
Table 236—appGetParasiticNetworksByName	308
Table 237—dpcmPassivateLoadModels	309
Table 238—dpcmPassivateInterconnectModels.....	309
Table 239—dpcmRestoreLoadModels	310
Table 240—dpcmRestoreInterconnectModels	310
Table 241—appGetCeff.....	311
Table 242—dpcmCalcCeff	312
Table 243—dpcmCalcSteadyStateResistanceRange	312
Table 244—dpcmCalcTristateResistanceRange.....	313
Table 245—appSetCeff	314
Table 246—dpcmCalcCouplingCapacitance.....	315
Table 247—dpcmCalcSubstrateCapacitance.....	316
Table 248—dpcmCalcSegmentResistance	317
Table 249—Manufacturing layer type values.....	317
Table 250—dpcmGetLayerArray	318
Table 251—dpcmGetRuleUnitToMeters.....	318
Table 252—dpcmGetRuleUnitToAmps	319
Table 253—appGetDriverThresholds.....	319
Table 254—appGetAggressorOverlapWindows.....	320
Table 255—appSetAggressorInteractWindows.....	322
Table 256—appGetOverlapNWFs.....	324
Table 257—appSetDriverInteractWindows.....	325
Table 258—dpcmCalcOutputResistances	326
Table 259—DCM_NoiseTypes	328
Table 260—docmGetLibraryNoiseTypesArray.....	329
Table 261—appNewNoiseCone	331
Table 262—NWF type	333
Table 263—PWF type	334
Table 264—PWFdriverModel	335
Table 265—dpcmGetPWFarray	335
Table 266—dpcmCreatePWF	336
Table 267—dpcmCopyNWFarray.....	336
Table 268—dpcmCopyPWF.....	337
Table 269—dpcmCreatePWFdriverModel.....	337
Table 270—dpcmGetPWFdriverModelArray	338
Table 271—dpcmGetSinkPinNoiseParasitics.....	341
Table 272—dpcmGetSourcePinNoiseParasitics.....	341
Table 273—dpcmBuildNoiseInterconnectModels.....	342
Table 274—dpcmBuildNoiseLoadModels	343

Table 275—driverPinNoise	344
Table 276—dpcmCalcInputNoise	346
Table 277—relatedPinNoise	347
Table 278—dpcmCalcOutputNoise	349
Table 279—appForEachNoiseParallelDriver	350
Table 280—dpcmSetParallelRelatedNoise	351
Table 281—appSetParallelOutputNoise	351
Table 282—dpcmSetNoiseLimit	352
Table 283—noiseViolationInfo	352
Table 284—appSetNoiseViolation	353
Table 285—dpcmGetNoiseViolationDetails	354
Table 286—appGetArrivalOffsetsByName	356
Table 287—appGetArrivalOffsetArraysByName	356
Table 288—Arc ordering	359
Table 289—PathDataBlock->modifiers enumeration values for priority operation	360
Table 290—DCM_TestTypes	364
Table 291—dpcmPerformPrimitive	364
Table 292—appGetArcStructure	365
Table 293—dpcmGetNodeSensitivity	365
Table 294—dpcmModelMoreFunctionDetail	366
Table 296—appSetXWF	370
Table 297—appGetXWF	371
Table 298—dpcmCalcXWF	372
Table 299—dpcmGetCellRailVoltageArray	373
Table 300—dpcmGetBaseCellRailVoltageArray	373
Table 301—dpcmGetBaseCellTemperature	374
Table 302—dpcmGetOpPointArray	375
Table 303—dpcmGetBaseOpPoint	376
Table 304—dpcmSetCurrentOpPoint	376
Table 305—DCM_ProcessVariations	377
Table 306—New predefined identifiers	378
Table 307—DCM_CalculationModes	379
Table 308—dpcmSetCurrentProcessPoint	379
Table 309—dpcmGetBaseProcessPoint	380
Table 310—dpcmGetProcessPointRange	380
Table 311—dpcmGetRailVoltageRangeArray	381
Table 312—dpcmGetCellRailVoltageRangeArray	382
Table 313—dpcmGetTemperatureRange	382
Table 314—dpcmGetCellTemperatureRange	383
Table 315—dpcmGetPinPinTypeArray	384
Table 316—dpcmGetPinPinType	385
Table 317—dpcmGetPinSignalTypeArray	385
Table 318—dpcmGetPinSignalType	387
Table 319—dpcmGetPinActionArray	387
Table 320—dpcmGetPinAction	388
Table 321—dpcmGetPinPolarityArray	388
Table 322—dpcmGetPinPolarity	389
Table 323—dpcmGetPinEnablePin	389
Table 324—dpcmGetPinConnectClass	390
Table 325—dpcmGetPinScanPosition	390
Table 326—dpcmGetPinStuckArray	391
Table 327—dpcmGetPinStuck	391
Table 328—dpcmGetDifferentialPairPin	392
Table 329—dpcmGetPathLabel	392
Table 330—dpcmGetPowerStateLabel	393
Table 331—dpcmGetCellTypeArray	393

Table 332—dpcmGetCellType	395
Table 333—dpcmGetCellSwapClassArray	395
Table 334—dpcmGetCellSwapClass.....	396
Table 335—dpcmGetCellRestrictClassArray	396
Table 336—dpcmGetCellRestrictClass	398
Table 337—dpcmGetCellScanTypeArray	398
Table 338—dpcmGetCellScanType	399
Table 339—dpcmGetCellNonScanCell.....	399
Table 340—DCM_PinMappingTypes	401
Table 341—appSetVectorOperations.....	401
Table 342—DCM_VectorOperations	402
Table 343—dpcmGetLevelShifter.....	403
Table 344—dpcmGetPinTiePolarity.....	404
Table 345—dpcmGetPinReadPolarity.....	404
Table 346—dpcmGetPinWritePolarity.....	405
Table 347—dpcmGetSimultaneousSwitchTimes	405
Table 348—appGetSwitchingBits	406
Table 349—dpcmGetFrequencyLimit	407
Table 350—appGetPinFrequency.....	407
Table 351—dpcmGetBasePinFrequency.....	408
Table 352—dpcmGetPinJitter	408
Table 353—dpcmGetInductanceLimit	409
Table 354—dpcmGetOutputSourceResistances	409
Table 355—appSetPull.....	410
Table 356—DCM_PullType.....	410
Table 357—dpcmGetPull	410
Table 358—dpcmGetPinDriveStrength.....	411
Table 359—dpcmGetCellVectorPower	412
Table 360—dpcmGetPinConnectivityArrays	412
Table 361—dpcmGetLibraryConnectClassArray.....	413
Table 362—dpcmGetLibraryConnectivityRules	413
Table 363—DCM_ConnectRules.....	414
Table 364—dpcmGetExistenceGraph	415
Table 365—dpcmGetTimingStateGraphs.....	416
Table 366—dpcmGetTimingStateStrings	418
Table 367—dpcmGetVectorEdgeNumbers.....	419
Table 368—appSetSignalDivision.....	420
Table 369—appSetSignalMultiplication	421
Table 370—appSetSignalGeneration.....	423
Table 371—dpcmGetDefPinSlews.....	424
Table 372—appGetInterfaceVersion	425
Table 373—Valid interface version strings.....	425
Table 374—dpcmSetResource	426
Table 375—dpcmGetAllResources	426
Table 376—DCM_ResourceTypes	428
Table 377—appGetExternalDelayByPin.....	429
Table 378—Description of multiArcMultiEdgePath (XXXX are do not care bits).....	430
Table 379—appGetExternalDelayByName.....	431
Table 380—appGetLogicLevelByName	432
Table 381 Table 381—DCM_LogicLevel	432
Table 382—appGetLogicLevelByPin	433
Table 383—dpcmGetPinIndexArrays	433
Table 384—dpcmGetSupplyPins.....	434
Table 385—DCM_BistInversion.....	435
Table 386—dpcmGetPhysicalBISTMap	435
Table 387—dpcmGetLogicalBISTMap.....	436

Table 388—dpcmGetCellTestProcedure	436
Table 389—icmBuildLoadModels	438
Table 390—icmBuildInterconnectModels.....	439
Table 391—icmCalcInterconnectDelaySlew.....	440
Table 392—icmCalcCellDelaySlew	442
Table 393—ccmCalcDelaySlew.....	443
Table 394—ccmEarlyLateIdentical.....	443
Table 395—ccmGetICMcontrolParams	444
Table 396—icmCalcOutputResistances	444
Table 397—icmCalcTotalLoadCapacitances.....	445
Table 398—icmCalcXWF	446
Table 399—icmInit.....	447
Table 400—dcmRT_new_DCM_ARRAY	454
Table 401—DCM_ATYPE enumeration.....	454
Table 402—DCM_AINIT enumeration.....	455
Table 403—DCM_ArrayInitUserFunction.....	456
Table 404—dcmRT_sizeof_DCM_ARRAY	456
Table 405—dcmRT_claim_DCM_ARRAY	456
Table 406—dcmRT_disclaim_DCM_ARRAY.....	457
Table 407—dcmRT_disclaim_DCM_STRUCT.....	458
Table 408—dcmRT_disclaim_DCM_STRUCT.....	458
Table 409—dcmRT_longlock_DCM_STRUCT	459
Table 410—dcmRT_longunlock_DCM_STRUCT	459
Table 411—dcmRT_getNumDimensions.....	460
Table 412—dcmRT_getNumElementsPer.....	460
Table 413—dcmRT_getNumElements.....	461
Table 414—dcmRT_getElementType.....	461
Table 415—dcmRT_arraycmp.....	461
Table 416—dcmRT_InitRuleSystem.....	462
Table 417—dcmRT_BindRule	463
Table 418—dcmRT_AppendRule	464
Table 419—dcmRT_UnbindRule	465
Table 420—dcmRT_FindFunction	466
Table 421—dcmRT_FindAppFunction	466
Table 422—dcmRT_QuietFindFunction	467
Table 423—dcmRT_MakeRC	467
Table 424—dcmRT_HardErrorRC.....	467
Table 425—dcmRT_SetMessageIntercept	468
Table 426—dcmRT_IssueMessage	468
Table 427—dcmRT_new_DCM_STD_STRUCT	469
Table 428—dcmRT_delete_DCM_STD_STRUCT	469
Table 429—dcmRT_setTechnology	470
Table 430—dcmRT_getTechnology.....	470
Table 431—dcmRT_getAllTechs	471
Table 432—dcmRT_freeAllTechs.....	471
Table 433—dcmRT_isGeneric	471
Table 434—dcmRT_takeMappingOfNugget.....	472
Table 435—dcmRT_registerUserObject.....	472
Table 436—dcmRT_DeleteRegisteredUserObjects.....	472
Table 437—dcmRT_DeleteOneUserObject	473
Table 438—delay.....	473
Table 439—slew.....	474
Table 440—check.....	474
Table 441—modelSearch	477
Table 442—Mode propagation operators	478
Table 443—Mode computation operators for delay and slew	479

Table 444—Mode operator enumerators for check	479
Table 445—Edge propagation enumeration pairs	480
Table 446—Edge propagation communication with DPCM	483
Table 447—newTimingPin.....	483
Table 448—newDelayMatrixRow	484
Table 449—newNetSinkPropagateSegments	485
Table 450—newNetSourcePropagateSegments	487
Table 451—newPropagateSegment.....	488
Table 452—newTestMatrixRow.....	488
Table 453—newAltTestSegment	489
Table 454—appGetPiModel	490
Table 455—appGetPolesAndResidues.....	491
Table 456—appGetCeffective	492
Table 457—appGetRLCnetworkByPin	493
Table 458—appGetRLCnetworkByName.....	494
Table 459—dpcmCalcPiModel	494
Table 460—dpcmCalcPolesAndResidues	495
Table 461—dpcmCalcCeffective.....	496
Table 462—dpcmSetRLCmember.....	496
Table 463—dpcmAppendPinAdmittance	498
Table 464—dpcmDeleteRLCnetwork	499
Table 465—dcm_copy_DCM_ARRAY	499
Table 466—dcm_new_DCM_ARRAY	500
Table 467—dcm_sizeof_DCM_ARRAY	500
Table 468—dcm_lock_DCM_ARRAY	500
Table 469—dcm_unlock_DCM_ARRAY	501
Table 470—dcm_lock_DCM_STRUCT	501
Table 471—dcm_unlock_DCM_STRUCT	502
Table 472—dcm_getNumDimensions.....	502
Table 473—dcm_getNumElementsPer	502
Table 474—dcm_getNumElements.....	503
Table 475—dcm_getElementType	503
Table 476—dcm_arraycmp	503
Table 477—dcmCellList.....	504
Table 478—dcmSetNewStorageManager.....	504
Table 479—dcmMalloc	505
Table 480—dcmFree	505
Table 481—dcmRealloc	505
Table 482—dcmBindRule	506
Table 483—dcmAddRule	506
Table 484—dcmUnbindRule	507
Table 485—dcmFindFunction	507
Table 486—dcmFindAppFunction	507
Table 487—dcmQuietFindFunction	508
Table 488—dcmMakeRC	508
Table 489—dcmHardErrorRC.....	509
Table 490—dcmSetMessageIntercept	509
Table 491—dcmIssueMessage	509
Table 492—dcm_rule_init.....	510
Table 493—DCM_new_DCM_STD_STRUCT.....	511
Table 494—DCM_delete_DCM_STD_STRUCT	511
Table 495—dcm_setTechnology	512
Table 496—dcm_getTechnology	512
Table 497—dcm_getAllTechs	513
Table 498—dcm_freeAllTechs.....	513
Table 499—dcm_isGeneric	513

Table 500—dcm_mapNugget.....	514
Table 501—dcm_takeMappingOfNugget.....	514
Table 502—dcm_registerUserObject	514
Table 503—dcm_DeleteRegisteredUserObjects	515
Table 504—dcm_DeleteOneUserObject	515
Table 505—Design flow values	605
Table 506—conn_attr	609
Table 507—Variation effect equations	612

BNF Syntax

Syntax 7.1—token	46
Syntax 7.2—identifier	49
Syntax 7.3—double_quoted_character.....	49
Syntax 7.4—constant.....	53
Syntax 7.5—string_literal.....	55
Syntax 7.6—operator.....	55
Syntax 7.7—punctuator.....	55
Syntax 7.8—native_type	57
Syntax 7.9—mathematical_type.....	58
Syntax 7.10—pointer_data_type	58
Syntax 7.11—aggregate_type.....	59
Syntax 7.12—aggregate_access	59
Syntax 7.13—array_type.....	61
Syntax 7.14—var.....	62
Syntax 7.15—cast.....	64
Syntax 7.16—new_operator	67
Syntax 7.17—scope_change.....	69
Syntax 7.18—launch	69
Syntax 7.19—FORCE operator.....	70
Syntax 7.20—array_index.....	74
Syntax 7.21—statement_call.....	74
Syntax 7.22—method_statement_call.....	75
Syntax 7.23—assign_variable_reference	75
Syntax 7.24—store_variable_reference.....	75
Syntax 7.25—expression.....	76
Syntax 7.26—discrete_expression.....	77
Syntax 7.27—logical_expression	78
Syntax 7.28—pin_range_list.....	79
Syntax 7.29—c_statement_reference	81
Syntax 7.30—passed_clause	84
Syntax 7.31—result_prototype.....	84
Syntax 7.32—conditional_result	85
Syntax 7.33—local_clause	86
Syntax 7.34—default_clause.....	87
Syntax 7.35—default_clause (result variable).....	87
Syntax 7.36—prototype_modifier.....	90
Syntax 7.37—common_prototype.....	90
Syntax 7.38—tabledef_prototype.....	91
Syntax 7.39—load_table_prototype.....	91
Syntax 7.40—add_row_prototype.....	91
Syntax 7.41—delay_prototype	91
Syntax 7.42—check_prototype	91
Syntax 7.43—submodel_prototype	92
Syntax 7.44—typedef.....	92
Syntax 7.45—expose_statement.....	93
Syntax 7.46—external_statement.....	94
Syntax 7.47—internal_statement.....	94
Syntax 7.48—constant_statement	95
Syntax 7.49—calculation_body.....	96
Syntax 7.50—calc_statement	96
Syntax 7.51—assign_statement.....	96
Syntax 7.52—delay_statement	97
Syntax 7.53—slew_statement	97

Syntax 7.54—check_statement	98
Syntax 7.55—check_statement	98
Syntax 7.56—ABS	106
Syntax 7.57—IMAG_PART	106
Syntax 7.58—REAL_PART	106
Syntax 7.59—EXPAND	106
Syntax 7.60—IS_EMPTY	107
Syntax 7.61—NUM_DIMENSIONS	107
Syntax 7.62—NUM_ELEMENTS	107
Syntax 7.63—ISSUE_MESSAGE	107
Syntax 7.64—PRINT_VALUE	108
Syntax 7.65—SOURCE_STRANDS_MSB	108
Syntax 7.66—SOURCE_STRANDS_LSB	109
Syntax 7.67—SINK_STRANDS_MSB	109
Syntax 7.68—SINK_STRANDS_LSB	109
Syntax 7.69—tabledef_statement	109
Syntax 7.70—table_statement	112
Syntax 7.71—load_table_statement	115
Syntax 7.72—unload_table_statement	117
Syntax 7.73—unload_table_statement	118
Syntax 7.74—add_row_statement	118
Syntax 7.75—delete_row_statement	119
Syntax 7.76—subrule_statement	130
Syntax 7.77—subrules_statement	131
Syntax 7.78—tech_family_statement	133
Syntax 7.79—tech_family_statement	136
Syntax 7.80—model_procedure	137
Syntax 7.81—submodel_procedure	138
Syntax 7.82—path_separator_stmt	139
Syntax 7.83—path_statement	140
Syntax 7.84—path_list	141
Syntax 7.85—clkflg_clause	142
Syntax 7.86—ckdtype_clause	142
Syntax 7.87—object_type_clause	143
Syntax 7.88—data_type_sequence	143
Syntax 7.89—bus_statement	145
Syntax 7.90—test_statement	145
Syntax 7.91—compare_list	146
Syntax 7.92—compare_clause	146
Syntax 7.93—edges_clause	147
Syntax 7.94—test_type_clause	147
Syntax 7.95—cycleadj_clause	147
Syntax 7.96—checks_clause	148
Syntax 7.97—methods_list	148
Syntax 7.98—store_clause	148
Syntax 7.99—test_bus_statement	149
Syntax 7.100—input_statement	149
Syntax 7.101—methods_clause	150
Syntax 7.102—store_clause	150
Syntax 7.103—output_statement	152
Syntax 7.104—do_statement – BREAK and CONTINUE	154
Syntax 7.105—do_statement	155
Syntax 7.106—statement_reference	155
Syntax 7.107—statement_reference	155
Syntax 7.108—node_sequence	156
Syntax 7.109—function_assignment_expression	172

Syntax 7.110—vector_sequence.....	173
Syntax 7.111—import_export_sequence.....	174
Syntax 7.112—properties_statement.....	175
Syntax 7.113—setvar_statement.....	175
Syntax 7.114—embedded_C_code.....	176
Syntax 7.115—subrule.....	176
Syntax 7.116—pragma_declare.....	177
Syntax 11.1—Alphanumeric characters.....	597
Syntax 11.2—SPEF names.....	597
Syntax 11.3—SPEF_file.....	598
Syntax 11.4—header_def.....	599
Syntax 11.5—unit_def.....	599
Syntax 11.6—name_map.....	599
Syntax 11.7—power_def.....	599
Syntax 11.8—external_def.....	600
Syntax 11.9—conn_attr.....	600
Syntax 11.10—define_def.....	600
Syntax 11.11—variation_def.....	601
Syntax 11.12—internal_def.....	601
Syntax 11.13—d_net.....	601
Syntax 11.14—conn_sec.....	601
Syntax 11.15—cap_sec.....	602
Syntax 11.16—res_sec.....	602
Syntax 11.17—induc_sec.....	602
Syntax 11.18—r_net.....	602
Syntax 11.19—load_desc.....	603
Syntax 11.20—d_pnet.....	603
Syntax 11.21—pconn_sec.....	603
Syntax 11.22—pcap_sec.....	603
Syntax 11.23—pres_sec.....	603
Syntax 11.24—pinduc_sec.....	604
Syntax 11.25—r_pnet.....	604

DELAY AND POWER CALCULATION STANDARDS –

Part 1: Integrated Circuit (IC) Open Library Architecture (OLA)

FOREWORD

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC document(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation.

IEEE Standards documents are developed with IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. *IEEE documents are made available for use subject to important notices and legal disclaimers (see <https://standards.ieee.org/ipr/disclaimers.html> for more information).*

IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations. This Dual Logo International Standard was jointly developed by the IEC and IEEE under the terms of that agreement.

- 2) The formal decisions of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees. The formal decisions of IEEE on technical matters, once consensus within IEEE Societies and Standards Coordinating Committees has been reached, is determined by a balanced ballot of materially interested parties who indicate interest in reviewing the proposed standard. Final approval of the IEEE standards document is given by the IEEE Standards Association (IEEE SA) Standards Board.
- 3) IEC/IEEE Publications have the form of recommendations for international use and are accepted by IEC National Committees/IEEE Societies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC/IEEE Publications is accurate, IEC or IEEE cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications (including IEC/IEEE Publications) transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC/IEEE Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC and IEEE do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC and IEEE are not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or IEEE or their directors, employees, servants or agents including individual experts and members of technical committees and IEC National Committees, or volunteers of IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board, for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC/IEEE Publication or any other IEC or IEEE Publications.
- 8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that implementation of this IEC/IEEE Publication may require use of material covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. IEC or IEEE shall not be held responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

IEC 61523-1/IEEE Std 1481 was processed through IEC technical committee 91: Electronics assembly technology, under the IEC/IEEE Dual Logo Agreement. It is an International Standard.

The text of this International Standard is based on the following documents:

IEEE Std	FDIS	Report on voting
1481 (2019)	91/1868/FDIS	91/1883/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

The IEC Technical Committee and IEEE Technical Committee have decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn, or
- revised.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)

Developed by the

Design Automation Standards Committee
of the
IEEE Computer Society

Approved 7 November 2019

IEEE SA Standards Board

Abstract: Ways for integrated circuit designers to analyze chip timing and power consistently across a broad set of electric design automation (EDA) applications are covered in this standard. Methods by which integrated circuit vendors can express timing and power information once per given technology are also covered. In addition, the means by which EDA vendors can meet their application performance and capacity needs are discussed.

Keywords: chip delay, electronic design automation (EDA), IEEE 1481™, integrated circuit (IC) design, power calculation

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (“IEEE SA”) Standards Board. IEEE (“the Institute”) develops its standards through a consensus development process, approved by the American National Standards Institute (“ANSI”), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore at <https://ieeexplore.ieee.org> or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website at <http://standards.ieee.org>.

Errata

Errata, if any, for IEEE standards can be accessed via <https://standards.ieee.org/standard/index.html>. Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore: <https://ieeexplore.ieee.org/browse/standards/collection/ieee/>. Users are encouraged to periodically check for errata.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at <https://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IEEE Introduction

This introduction is not part of IEEE Std 1481-2019, IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA).

The objective of the delay and power calculation system (DPCS) is to make it possible for integrated circuit designers to consistently calculate chip delay and power across electronic design automation (EDA) applications and for integrated circuit vendors to express delay and power information only once per technology while enabling sufficient EDA application accuracy.

This is accomplished by a coordinated set of standards that support a standard method to describe timing and power characteristics of integrated circuit design units (cells and higher level design elements); a standard method for EDA applications to calculate chip design instance specific delay, slew, and power for logic and interconnects; and standard file formats to exchange chip parasitic and cluster information.

Acknowledgments

Royalty-free nonexclusive permission has been granted by International Business Machines (IBM) Corporation for all written contributions made by IBM under the direction of Harry J. Beatty III.

Royalty-free permission has been granted by Silicon Integration Initiative, Inc. (Si2) to reprint material from Specification for the Open Library Architecture (OLA), Version 2.0-00, March 1, 2003.

IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)

1. Overview

The delay and power calculation system (DPCS) is a coordinated set of standards that support a standard method to describe timing and power characteristics of integrated circuit (IC) design units (cells and higher level design elements); a standard method for electronic design automation (EDA) applications to calculate chip design instance specific delay, slew, and power for logic and interconnects; and standard file formats to exchange chip parasitic and cluster information. The standard specifications covered in this document are as follows:

- Description language for timing and power modeling, called the “delay calculation language” (DCL)
- Software procedural interface (PI) for communications between EDA applications and compiled libraries of DCL descriptions
- Standard file exchange format for parasitic information about the chip design: Standard Parasitic Exchange Format (SPEF)
- Informative usage examples
- Informative notes

Notes and examples are informative. All other components of this specification are considered normative unless otherwise directed.

1.1 Scope

The scope of this standard focuses on delay and power calculation for integrated circuit design with support for modeling logical behavior and signal integrity.

1.2 Purpose

To improve the IEEE Std 1481™-1999 system for integrated circuit designers to more accurately and more completely analyze semiconductor designs across EDA applications and for integrated circuit vendors to express logical behavior, signal integrity, delay, and power information only once per technology while enabling sufficient EDA application accuracy.

1.3 Introduction

The DPCS standard covers delay and power calculation for integrated circuit design with support for modeling logical behavior and signal integrity, which makes it possible for integrated circuit designers to

analyze chip timing and power consistently across a broad set of EDA applications, for integrated circuit vendors to express timing and power information once (for a given technology), and for EDA vendors to meet their application performance and capacity needs. The intended use for this standard is IC timing and power. This standard may be applied to both unit logic cells supplied by the IC vendor and logical macros defined by the IC designer. Although this standard is written toward the integrated circuit supplier and EDA developer, its application applies equally well to representation of timing and power for designer-defined macros (or hierarchical design elements).

These specifications make it possible to achieve consistent timing and power results, but they do not guarantee it. They provide for a single executable software program that computes delay and power based on IC vendor-supplied algorithms (or designer-supplied algorithms for macros) but does not guarantee EDA applications can correctly communicate the design-specific information required for these algorithms. By specifying standard exchange formats for parasitic data and floorplanning information, this standard provides a marked improvement over design environments with no such standards. However, it is the responsibility of the EDA application to correctly correlate the information between these standard exchange files and the actual design. This standard also does not detail how the information contained within the standard exchange files shall be obtained.

As feature sizes for chips have shrink below 0.25 μm , interconnect delay effects have begun to outweigh those of the logic cells. This means placement of cells and wire routing of the interconnects become as important a factor as the type of cell drivers and receivers on the interconnect. As a result, EDA logic design applications (such as synthesis) have begun to interact closely with physical design applications (such as floorplanning and layout). Applications that before could consider only simple delay and power models now need to deal with complex, interrelated delay and power algorithms. Plus, due to the complexities of the delay and power algorithms, the integrated circuit vendor needs to have control of application calculations and not be restricted by the limitations of a broad set of applications demanded by the customers (the designers).

Over the past few years, it has become increasingly apparent that modern very large-scale integration (VLSI) design is no longer bounded only by timing and area constraints. Power has become significantly more important. In an era of hand-held devices, ranging from mobile computing to wireless communication systems, managing and controlling power takes on an important role. Several benefits can be attained from low-power designs in addition to extended battery life. Low-power devices often run at a lower junction temperature, which leads to higher reliability and lower cost cooling systems. There are also several challenges for calculation and modeling of power (and delay) in deep submicron (less than 0.25 μm) designs. EDA tools can now accurately calculate and model power by using this DPCS standard.

1.4 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (shall equals is required to).^{1,2}

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (should equals is recommended that).

The word *may* is used to indicate a course of action permissible within the limits of the standard (may equals is permitted to).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (can equals is able to).

¹ The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.

² The use of *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

ISO/IEC 9899:1990, Programming Languages—C.³

ISO/IEC 14882:2003, Programming Languages—C++.

³ISO publications are available from the International Organization for Standardization (<http://www.iso.org/>) and the American National Standards Institute (<http://www.ansi.org/>). IEC publications are available from the International Electrotechnical Commission (<http://www.iec.ch>) and the American National Standards Institute (<http://www.ansi.org/>).