Physical Synthesis Data Format (PSDF) Specification

Version 0.3

May 27, 2005

By

Altera Corporation

Table of Contents:

1. Overview

This document describes the syntax and contents of Physical Synthesis Data File (PSDF).

PSDF file enables EDA synthesis tools to access physical synthesis data from the Quartus® II development tools in a defined and consistent way. EDA tools can use the placement and delay data to resynthesize and optimize the design to meet the user constraints in the final place&routed design.

2. Contents of PSDF

PSDF is contained in an ASCII format file and can be generated after place&route. The file contains information about interconnect delays, atom locations, register-packing in I/Os, LUT input permutations, pin assignments.

This section describes the PSDF file contents. The file is organized as a partially ordered set of statements. Each statement is a single text line delimited by 'newline'. Each statement starts with a unique letter followed by a space. This first letter identifies the type of statement. All lines that start with '#' are single line comments. An ordered subset of the statements is called a record in this document.

2.1 PSDF Version Statement

Letter 'V' identifies this statement. The statement contains the PSDF file version number. This version number can be used to differentiate between changes in the file format specifications. Tools accessing the file can process the rest of the file based on this version number.

V <version number>

E.g. *V 1.0*

2.2 LogicLock Region Statement

Letter 'G' identifies this statement. The statement provides post-fit information about LogicLock regions in the design and assigns a unique ID number to each region that is used throughout the rest of the file to denote region membership on the various instances.

G <id> "<name>" <origin> <state> \"<parent>\" <height> <width> <autosize>

E.g. *G 1 "my logiclock region" X10_Y10 locked "" 10 10 0*
G 2 "my child logiclock region" X12_Y12 floating "my logiclock region" 2 2 1

2.3 ID Statement

Letter 'N' identifies this statement. The statement associates each instance in the design with a unique integer ID. It also provides the name of entity for the instance. All other statements refer to the instance using this ID. The purpose of this ID is to optimize the size of file and subsequent processing time for tools.

N <instance ID> <instance path> <entity name>

E.g. *N 1269 top.myram.ram_inst_1 stratix_ram_block*

2.4 Output Term ID Statement

Letter ‘O’ identifies this statement. The statement associates possible output terms with each instance in the design. The integer ID refers to the instance that the output term is a child of. The statement also provides the name of the entity for the output term, but the main purpose is to identify possible buried instances in the design as a result of register packing.

O <instance ID> <output term path> <entity name> <pin name>

E.g. *O 1269 inst_2[0] stratix_ram_block dataout[0]*

2.5 Pin Statement

Letter ‘I’ identifies this statement. The statement associates each design interface port with its i/o atom location on device and the package pin it connects to. The pin name starts with “Pin_” followed by actual pin name on package.

I <i/f port name> <location> <package pin name>

E.g. *I addr[3] T_50 Pin_A5*

2.6 I/O Register Statement

Letter ‘R’ identifies this statement. The statement associates an instance of register with i/f port. The presence of this statement indicates that the original register instance has been packed into the specified i/f port during place&route. Tools can use this data during back-annotation for the register.

R <instance ID> <i/f port name>

E.g. *R 1376 dout[7]*

2.7 Configuration Statement

Letter ‘C’ identifies this statement. The statement associates an instance with its configuration parameters.

C <instance ID> {<param name>=<param value>}

E.g. *C 967 mode=normal sumc=datac mask=FF00*

The configuration parameters are described below. The relevant names of parameters on wysiwyg atom are provided within paranthesis.

All Atom Type

1. **llr** – LogicLock region membership. Can have any integer value. The value corresponds to the LogicLock region ID to which this atom has been assigned. Atoms can assigned explicitly to only one LogicLock region at time.

Lcell Atom

1. **mode** – Operation mode (*operation_mode*). Can have values of “normal”, “arithmetic”.
2. **sumc** – sum LUT C input (*sum_lutc_input*). Can have values of “datac”, “qfbk”, “cin”.
3. **mask** – LUT mask (*lut_mask*). Can have values of 4 digit hexadecimal number.

I/O Atom

1. **mode** – Operation mode (*operation_mode*). Can have values of “input”, “output”, “bidir”.

RAM Atom

1. **mode** – Operation mode (*operation_mode*). Can have values of “single_port”, “dual_port”, “bidir_dual_port”, “rom”.
2. **iclka** – Port A data in clock (*port_a_data_in_clock*). Can have values of “clock0”, “clock1”, “none”.
3. **iclra** – Port A data in clear (*port_a_data_in_clear*). Can have values of “clear0”, “none”.
4. **oclka** – Port A data out clock (*port_a_data_out_clock*). Can have values of “clock0”, “clock1”, “none”.
5. **oclra** – Port A data out clear (*port_a_data_out_clear*). Can have values of “clear0”, “clear1”, “none”.
6. **raclka** – Port A read address clock (*port_a_read_address_clock*). Can have values of “clock0”, “clock1”, “none”.
7. **raclra** – Port A read address clear (*port_a_read_address_clear*). Can have values of “clear0”, “clear1”, “none”.
8. **reclka** – Port A read enable clock (*port_a_read_enable_clock*). Can have values of “clock0”, “clock1”, “none”.
9. **reclra** – Port A read enable clear (*port_a_read_enable_clear*). Can have values of “clear0”, “clear1”, “none”.
10. **wlclka** – Port A write logic clock (*port_a_write_logic_clock*). Can have values of “clock0”, “clock1”, “none”.
11. **waclra** – Port A write address clear (*port_a_write_address_clear*). Can have values of “clear0”, “clear1”, “none”.
12. **weclra** – Port A write enable clear (*port_a_write_enable_clear*). Can have values of “clear0”, “clear1”, “none”.
13. **iclkb** – Port B data in clock (*port_b_data_in_clock*). Can have values of “clock0”, “clock1”, “none”.
14. **iclrb** – Port B data in clear (*port_b_data_in_clear*). Can have values of “clear0”, “none”.
15. **oclkb** – Port B data out clock (*port_b_data_out_clock*). Can have values of “clock0”, “clock1”, “none”.
16. **oclrb** – Port B data out clear (*port_b_data_out_clear*). Can have values of “clear0”, “clear1”, “none”.
17. **raclkb** – Port B read address clock (*port_b_read_address_clock*). Can have values of “clock0”, “clock1”, “none”.
18. **raclrb** – Port B read address clear (*port_b_read_address_clear*). Can have values of “clear0”, “clear1”, “none”.
19. **reclkb** – Port B read enable clock (*port_b_read_enable_clock*). Can have values of “clock0”, “clock1”, “none”.
20. **reclrb** – Port B read enable clear (*port_b_read_enable_clear*). Can have values of “clear0”, “clear1”, “none”.
21. **wlclkb** – Port B write logic clock (*port_b_write_logic_clock*). Can have values of “clock0”, “clock1”, “none”.
22. **waclrb** – Port B write address clear (*port_b_write_address_clear*). Can have values of “clear0”, “clear1”, “none”.
23. **weclrb** – Port B write enable clear (*port_b_write_enable_clear*). Can have values of “clear0”, “clear1”, “none”.
24. **type** – RAM block type (*ram_block_type*). Can have values of “auto”, “M512”, “M4K”, “MRAM”.
25. **beclra** – Port A byte enable clear (*port_a_byteenamask_clear*). Can have values of “clear0”, “clear1”, “none”.
26. **dwda** – Port A data width (*port_a_data_width*). Can have a positive integer value.
27. **aclra** – Port A address clear (*port_a_address_clear*). Can have values of “clear0”, “clear1”, “none”.
28. **beclkb** – Port B byte enable clock (*port_b_byteenamask_clock*). Can have values of “clock0”, “clock1”, “none”.

29. **beclrb** – Port B byte enable clear (*port_b_byteenamask_clear*). Can have values of “clear0”, “clear1”, “none”.
30. **dwdb** – Port B data width (*port_b_data_width*). Can have a positive integer value.
31. **aclkb** – Port B address clock (*port_b_address_clock*). Can have values of “clock0”, “clock1”, “none”.
32. **aclrb** – Port B address clear (*port_b_address_clear*). Can have values of “clear0”, “clear1”, “none”.
33. **rweclkb** – Port B read/write enable clock (*port_b_read_enable_write_enable_clock*). Can have values of “clock0”, “clock1”, “none”.
34. **rweclrb** – Port B read/write enable clear (*port_b_read_enable_write_enable_clear*). Can have values of “clear0”, “clear1”, “none”.

MAC MULT Atom

1. **dwda** – Data A width (*dataa_width*). Can have a positive integer value.
2. **dwdb** – Data B width (*datab_width*). Can have a positive integer value.
3. **dclka** – Data A clock (*dataa_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
4. **dclkb** – Data B clock (*datab_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
5. **dclra** – Data A clear (*dataa_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
6. **dclrb** – Data B clear (*datab_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
7. **sclka** – Sign A clock (*signa_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
8. **sclkb** – Sign B clock (*signb_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
9. **sclra** – Sign A clear (*signa_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
10. **sclrb** – Sign B clear (*signb_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
11. **oclk** – Data out clock (*output_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
12. **oclr** – Data out clear (*output_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.

MAC OUT Atom

1. **dwda** – Data A width (*dataa_width*). Can have a positive integer value.
2. **dwdb** – Data B width (*datab_width*). Can have a positive integer value.
3. **dwdc** – Data C width (*datac_width*). Can have a positive integer value.
4. **dwdd** – Data D width (*datad_width*). Can have a positive integer value.
5. **aclk0** – Addnsub 0 clock (*addnsub0_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
6. **aclk1** – Addnsub 1 clock (*addnsub1_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
7. **aclr0** – Addnsub 0 clear (*addnsub0_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
8. **aclr1** – Addnsub 1 clear (*addnsub1_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
9. **zclk** – Zeroacc clock (*zeroacc_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
10. **zclr** – Zeroacc clock (*zeroacc_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
11. **sclka** – Sign A clock (*signa_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
12. **sclkb** – Sign B clock (*signb_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
13. **sclra** – Sign A clear (*signa_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
14. **sclrb** – Sign B clear (*signb_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
15. **apclk0** – Addnsub 0 pipeline clock (*addnsub0_pipeline_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
16. **apclk1** – Addnsub 1 pipeline clock (*addnsub1_pipeline_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
17. **zpclk** – Zeroacc pipeline clock (*zeroacc_pipeline_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
18. **spclka** – Sign A pipeline clock (*signa_pipeline_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
19. **spclkb** – Sign B pipeline clock (*signb_pipeline_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
20. **apclr0** – Addnsub 0 pipeline clear (*addnsub0_pipeline_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.

21. **apclr1** – Addnsub 1 pipeline clear (*addnsub1_pipeline_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
22. **zpclr** – Zeroacc pipeline clear (*zeroacc_pipeline_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
23. **spclra** – Sign A pipeline clear (*signa_pipeline_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
24. **spclrb** – Sign B pipeline clear (*signb_pipeline_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.
25. **oclk** – Data out clock (*output_clock*). Can have values of “0”, “1”, “2”, “3”, “none”.
26. **oclr** – Data out clear (*output_clear*). Can have values of “0”, “1”, “2”, “3”, “none”.

2.8 Placement Statement

Letter ‘P’ identifies this statement. The statement associates an instance with its location on device. It also optionally describes the permutation of LUT inputs for lcell atom.

P <instance ID> <location> [<permutation vector>]

E.g. *P 967 LC_X47_Y30_N4 {2,1,3,0}*

The permutation vector is of the form { i1, i2, i3, i4 }, where i1, i2, i3 & i4 are input indices of LUT. The LUT inputs dataa, datab, datac, datad have indices of 1, 2, 3, 4 respectively. The index 0 indicates absence of a connection. The permutation implies that original signal connected to LUT input indexed i1 is now connected to dataa, original signal connected to LUT input indexed i2 is now connected to datab and so forth.

2.9 Driver/Load Record

This record contains statements which specify a driver instance followed by a set of load instances. It represents the connectivity between the instances.

<driver statement>

<load statement>

…

…

<load statement>

2.10 Driver Statement

Letter ‘D’ identifies this statement. The statement specifies an instance and its port which is driver for a signal. The statement is followed by a set of load statements.

D <instance ID> <instance port name>

E.g. *D 967 regout*

2.11 Load Statement

Letter ‘L’ identifies this statement. The statement specifies an instance and its port which is load on the previously specified driver. It specifies the interconnect delay between the driver and load. The delay values are in picoseconds. The statement is preceded by either other load statements or a driver statement.

L <instance ID> <instance port name> <i/c delay>

E.g. *L 968 dataa 924*

2.12 Black Box Statement

Letter ‘B’ identifies this statement. The statement associates an instance within a black box entity with the instance of the black box entity.

B <instance ID> <instance ID>

E.g. *B 1437 1438*

Where ID 1437 is for instance top.myram.ram_block_inst_1 and

ID 1438 is for instance top.myram

3. Syntax of PSDF

The formal description of the PSDF file syntax is as below.

integer – A positive integer number. E.g. 23, 498.

real – A real number. E.g. 1476, 43.8.

string – Any string. E.g. 1.3, abc.

identifier – A string containing only alphanumeric characters (alphabet, number, underscore). E.g. top, sub_inst.

PSDF Syntax

psdf_file	::= records_list
records_list	::= version_statement
	||= logiclock_statement ||= id_statement ||= oterm_id_statement ||= pin_statement ||= ioreg_statement ||= config_statement ||= placement_statement ||= driver_load_record ||= blackbox_statement
driver_load_record	::= driver_statement { load_statement }
version_statement	::= ‘V’ version_string
logiclock_statement llr_width llr_autosize	::= ‘G’ llr_id “llr_name” llr_origin llr_state “llr_parent” llr_height
id_statement	::= ‘N’ instance_id instance_path entity_name
oterm_id_statement	::= ‘O’ instance_id output_term_path entity_name pin_name
pin_statement	::= ‘I’ port_name location pin_name
ioreg_statement	::= ‘R’ instance_id port_name
config_statement	::= ‘C’ instance_id {param_name=param_value}

placement_statement	::= ‘P’ instance_id location ||= ‘P’ instance_id location permutation_vector
driver_statement	::= ‘D’ instance_id port_name
load_statement	::= ‘L’ instance_id port_name delay
blackbox_statement	::= ‘B’ instance_id instance_id
instance_path	::= *identifier*{‘.’*identifier*}
pin_name	::= *string*
entity_name	::= *identifier*
instance_id	::= *integer*
version_string	::= *string*
port_name	::= *identifier* ||= identifier ‘[‘ *integer* ‘:’ *integer* ‘]’
pin_name	::= Pin_*string*
param_name	::= *identifier*
param_value	::= *string*
	||= *integer*
location	::= *string*
delay	::= *real*
permutation_vector	::= ‘{‘ *integer* ‘,’ *integer* ‘,’ *integer* ‘,’ *integer* ‘}’
llr_id	::= *integer*
llr_name	::= string
llr_parent	::= string
llr_height	::= integer
llr_width	::= integer
llr_autosize	::= 0 | 1
llr_state	::= floating | locked | soft
llr_origin	::= ‘X’*integer*’_Y’*integer*