Synopsys Design Constraints (SDC)
Open Source Parser

SDC Parser User's Manual

Synopsys, Inc.
Intrinsix Corp.

Contents

1. Introduction

Synopsys Design Constraints Format used as common format of Design Constraints representation for EDA tools.
SDC Parser supports reading SDC files by EDA tools. SDC Parser command-line shell also could be used as a checker of SDC files (see Figure 1).

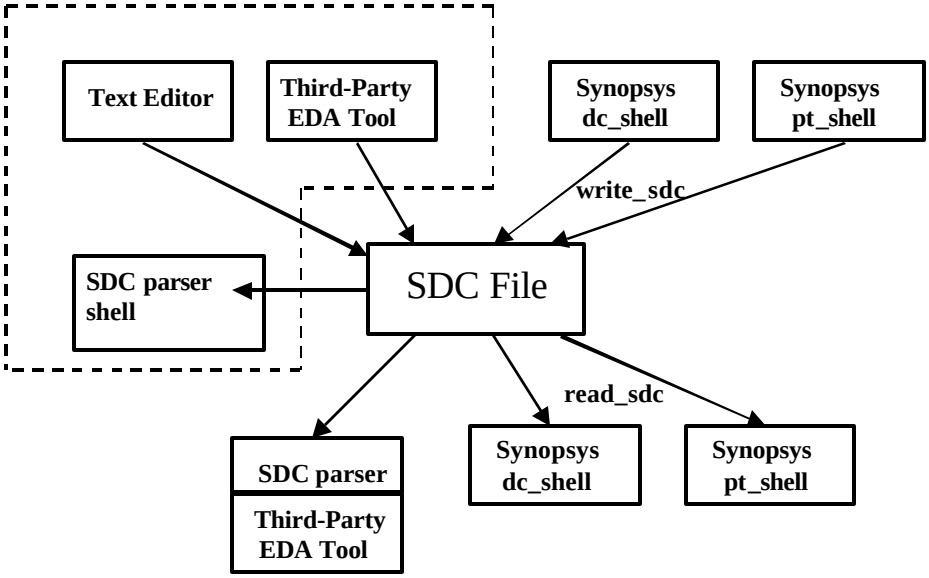

Figure 1. SDC File Generation & Processing

The main features of SDC parser are the following:

- It checks SDC file in accordance with SDC Spec and provides error messages
- It generates internal data structure convenient for processing by EDA tool
- It provides Tcl API and command-line interface for EDA tool
- It supports several SDC versions, which can be defined in SDC file or by EDA application
- It works at any platform supporting TCL 8.0 (or newer) Solaris, Linux, Windows NT/2000/98/95

2. Parser Architecture

2.1. SDC Parser Data Flow

The Data flow shown on Figure 2.

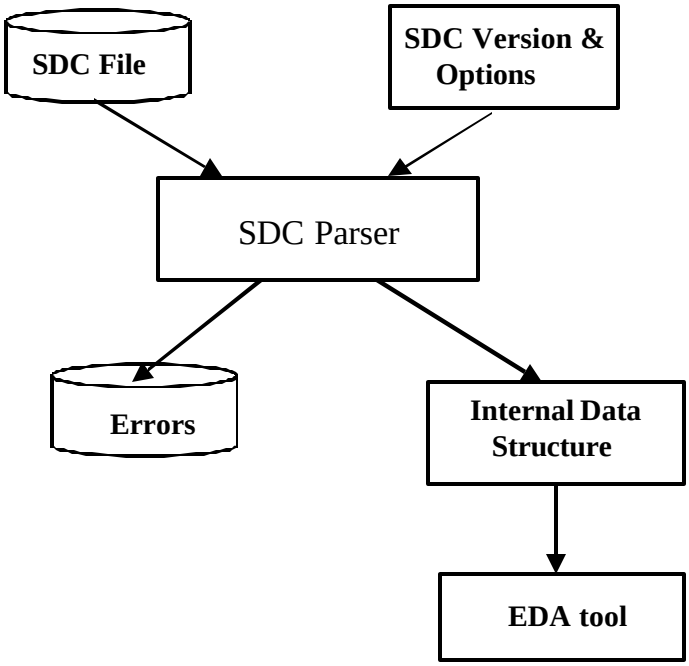

Figure 2. SDC Parser Data Flow

2.2. Program structure of SDC parser

Figure 3 shows the Program Structure.

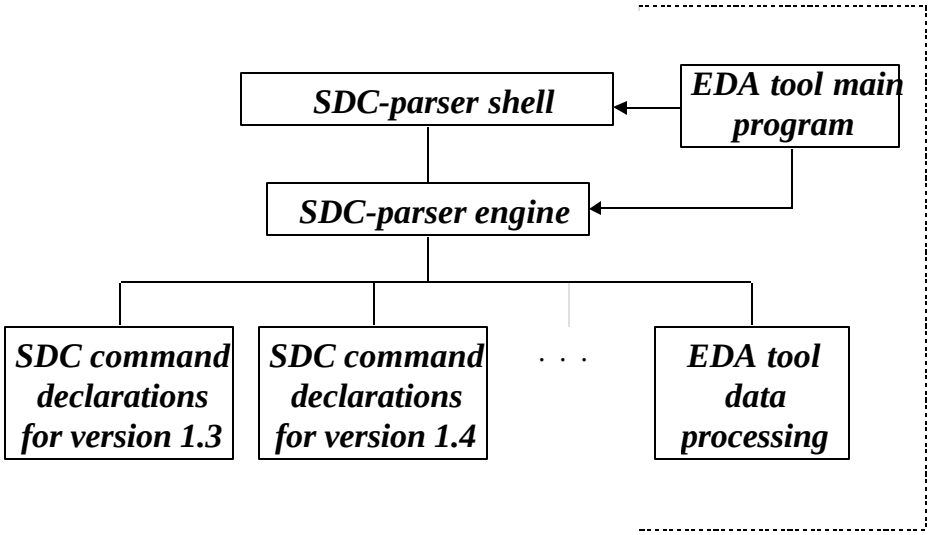

Figure 3. Program Structure of SDC Parser.

Descriptions of modules:

SDC-parser shell

Interface program for working with SDC-files from command prompt through SDC-parser engine API

SDC-parser engine

Contains API and general functions of SDC-parser

SDC-parser declarations for version X.X

Contains descriptions of SDC commands.
There is set of files for SDC versions supported by the pars er.
Each file contains set of functions for supported by particular SDC version SDC commands in the format like this:

- ? name of the command
- ? list of parameters and properties of parameters for this command (name, type, mandatory or optional, restrictions)

EDA tool process data procedure

This procedure extracts data from data structure built by the parser and processes them in accordance with project specification.

2.3. Algorithm of SDC Parser.

General algorithm of SDC Parser presented in Figure 4.

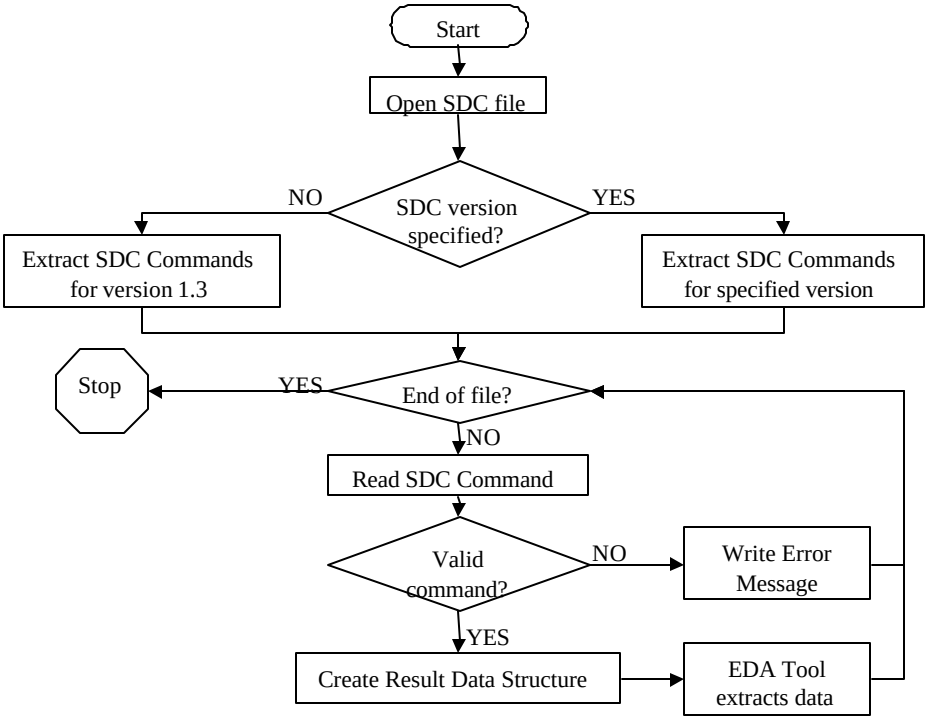

Figure 4. General algorithm of SDC Parser engine.

3. Parser Interface

EDA tool could use Parser Engine API or command-line interface.
Simple example of EDA tool main program with call-back procedure presented in Appendix B.

3.1 API Functions.

There is below list of API functions with their descriptions:

sdc::parse_file

The **sdc::parse_file** function runs sdc-parser for reading SDC file. The syntax is:

sdc::parse_file {SDC_file_name}

SDC_file_name – name of SDC file (which contain SDC commands).

sdc::parse_command

The **sdc::parse_command** function runs sdc-parser for one SDC command and returns result of executing EDA callback function. The syntax is:

sdc::parse_command {version command_name args}

version – version number of SDC (Ex.: 1.3)
command_name – name of the command (Ex.: create_clock)
args – list of commands parameters

sdc::declare

The **sdc::declare** function declare one SDC command (each SDC command should be declared for particular SDC version). The syntax is:

sdc::declare {command_name arguments [condition]}

command_name – name of the command (Ex.: create_clock)
arguments - list of descriptions of commands parameters
condition – condition describing correct set of SDC-command's arguments

Example of command declaration:

```
declare  create_clock {
  {-period          Float           {$par >= 0}}
  {-name            String          }
  {-waveform        List            }
  {port_pin_list    List            {type_Float}}
 } {param(-period)}
```

sdc::register_callback

The **sdc::register_callback** function set a callback for EDA tool. The syntax is:

sdc::register_callback {proc_name}

proc_name – name of "callback" procedure (if proc_name is empty callback will be disabled)

Syntax of callback procedure:

proc_name {command_name array_name}
EDA-tool process data and returns flag "continue parsing"
command_name *– name of SDC command*
array_name *– name of array with data after parsing*

sdc::set_version

The **sdc::set_version** function change a current SDC version number. The syntax is:

sdc::set_version {sdc_version}

sdc_version – new version number

sdc::out_message

The **sdc::out_message** function puts message to the console / log file depending on settings. The syntax is:

sdc::out_message {message}

message – text of the message

sdc::set_log_file

The **sdc::set_log_file** function sets name of log-file. The syntax is:

sdc::set_log_file {filename}

filename – name of log-file

sdc::set_debug_level

The **sdc::set_debug_level** function sets debug level. The syntax is:

sdc::set_debug_level {level}

level – level of debug (0-2)

3.2 Command Line Interface.

To call SDC-parser shell use the following command line:

sdcparser.tcl [**options**] [**filename** [...]]

where

<filename> - source file name;
-f <filename> - read command line arguments from file;
-d<debug_level> - set debug level
(d0 – no error messages, quiet mode,
d1 – brief error messages,
d2 – extended error messages);

-l <log_file_name> - set log file name;
-v <sdc_version> - set default sdc version;
-eda <eda_source_file> - file name which contains tcl source of eda tool procedure.

4. Data Structure After Parsing

After parsing, if callback was registered, callback procedure would be called with two parameters:

- SDC command name, which was parsed
- Name of associative array which contains parsed parameters with values (data structure)

For example, after parsing of command

create_clock [get_ports CLK2] -period 12.2 -name CLK2

the data structure will look like

data(-period) = 12.2
data(-name) = CLK2
data(port_pin_list) = result of function **get_ports**

See SDC Syntax description (Appendix A) to figure out the name of "non-dash" parameters.

5. SDC Versions Support

5.1. Declarations of SDC commands

SDC Commands are described in separate files for particular SDC version.

List of all supported versions presented in variable ***validsdcversions*** in file "*sdcparsercore.tcl*"
In format **[list 1.1 1.2 1.3 1.4…]**

File name for particular version has the following format: "*sdc<new_version>.tcl*" (Ex.: *sdc1.3.tcl*)

File structure is the sequence of SDC command declarations.

Format of command declarations is the following:

[sdc::]**declare {command_name arguments [condition]}**

Arguments have the following format:

```
{
{key_name    type  [additional_parameters]}
{key_name    type  [additional_parameters]}
…
{key_name    type  [additional_parameters]}
}
```

where

key_name – name of key (if first char is "-", example: "-delay") or internal name of positional parameter.

type – type of parameter. Valid values are:

String	- value of this parameter must be a string type
Int	- value of this parameter must be an integer type, ***additional_parameters*** must be a expression (if no restrictions needed, put **"1"** to this parameter)
Float	- value of this parameter must be a float type, ***additional_parameters*** must be a expression (if no restrictions needed, put **"1"** to this parameter)
Flag	- no value for this parameter
Enum	- value of this parameter must be an enumerated string type, ***additional_parameters*** must be a list of valid values of parameter
List	- value of this parameter mu st be a list type, ***additional_parameters*** can be a list which contains additional flags and condition of list length
Unknown	- no checking of type will processed

Examples:

```
declare create_clock  {
  {-period          Float     {$par>=0}}
  {-name            String     }
  {-waveform        List       {type_Float {length($length>=2 && ($length % 2)==0)}}}
  {port_pin_list    List       }
 }  {param(-period)}

declare set_min_delay {
  {delay_value     Float     {1}}
  {-rise           Flag      }
  {-fall           Flag      }
  {-from           List      }
  {-to             List      }
  {-through        List      {dup}}
 } {param(delay_value)}


declare get_cells  {
  {-of_objects     List          }
  {patterns        List          }
  {-hierarchical   Flag          }
 }  {(param(patterns) && !param(-of_objects)) || (param(-of_objects) &&
param(patterns) && !param(-hierarchical))}
```

Enabled combinations of arguments here are:

patterns and no ***-of_objects***;

-of_objects and no ***patterns*** and no ***-hierarchical***.

5.2. Modification of SDC command set

To add support of new version of SDC commands, you must:

1) Add a new version value to list
`variable validsdcversions [list 1.2 1.3 1.4` **`<new_version>`**`]`
in file *“sdcpak.tcl”*
2) Create a new file which named “sdc<new_version>.tcl” (Ex.: *sdc1.4.1.tcl*)
3) Declare SDC command for added version

To modify list of commands / command parameters for already presented version just edit appropriate file “sdc<version>.tcl”

Appendix A

For latest version SDC Commands description download “Using Synopsys Design Constraints Format Application Note” from Synopsys Tap-in SDC download selection.

SDC commands description for version 1.4

command **option-type**	**result-type** **summary**
1. all_clocks	RESULT: collection of objects
2. all_inputs	RESULT: collection of objects
-level_sensitive	boolean optional
-edge_triggered	boolean optional
-clock name	list optional
3. all_outputs	RESULT: collection of objects
-level_sensitive	boolean optional
-edge_triggered	boolean optional
-clock name	list optional
4. create_clock	RESULT: 1/0
-period value	float >= 0 **required**
-name value	string optional
-waveform list	float-list even-length optional
port_pin_list	list optional
-add	boolean optional
Note: Either Object list or clock_name(-name) must be specified	
5. current_design	RESULT: collection of objects (1 object)
6. current_instance	RESULT: none
7. get_cells	RESULT: collection of objects
-hierarchical	boolean optional
-hsc	boolean optional
-of_objects oovalue	list exclusive-of-patterns-one-required
patterns	list exclusive-of-oovalue-one-required
Note: -of_objects and patterns are mutually exclusive; you must specify one, but not both	
8. get_clocks	RESULT: collection of objects
patterns	list **required**

9. get_lib_cells	RESULT: collection of objects
-hsc	boolean optional
patterns	list **required**
10. get_lib_pins	RESULT: collection of objects
-hsc	boolean optional
patterns	list **required**
11. get_libs	RESULT: collection of objects
patterns	list **required**
12. get_nets	RESULT: collection of objects
-hierarchical	boolean optional
patterns	list **required**
13. get_pins	RESULT: collection of objects
-hierarchical	boolean optional
patterns	list **required**
14. get_ports	RESULT: collection of objects
patterns	list **required**
15. set_case_analysis	RESULT: 1/0
value	enum {0 1 rising falling zero one rise fall} **required**
port_or_pin_list	list **required**
16. set_clock_latency	RESULT: 1/0
-rise	boolean optional
-fall	boolean optional
-min	boolean optional
-max	boolean optional
-late	boolean optional
-early	boolean optional
-source	boolean optional
delay	float **required**
object_list	list **required**
17. set_clock_transition	RESULT: 1/0
-rise	boolean optional
-fall	boolean optional
-min	boolean optional
-max	boolean optional
transition	float **required**
clock_list	list **required**

Note: -rise and -fall mutually exclusive,

18. set_clock_uncertainty	RESULT: 1/0
-from fva	list optional
-to tva	list optional
-rise	boolean optional
-fall	boolean optional
-setup	boolean optional
-hold	boolean optional
uncertainty	float **required**
object_list	list optional

*** Either -from/-to or object_list, but not both.*

20. set_disable_timing	RESULT: 1/0
-from fva	string optional
-to tva	string optional
object_list	list **required**

21. set_drive	RESULT: 1/0
-rise	boolean optional
-fall	boolean optional
-min	boolean optional
-max	boolean optional
resistance	float >= 0 required
port_list	list **required**

22. set_driving_cell	RESULT: 1/0
-lib_cell	string **optional**
-rise	boolean optional
-fall	boolean optional
-library lb	string optional
-pin pn	string optional
-from_pin fp	string optional
-multiply_by mb	float >= 0 optional
-dont_scale	boolean optional
-no_design_rule	boolean optional
-input_transition_rise itr	float >= 0 optional
-input_transition_fall itf	float >= 0 optional
port_list	list **required**
-min	boolean optional
-max	boolean optional
-clock	list optional
-clock_fall	boolean optional

23. set_false_path	RESULT: 1/0
-setup	boolean optional
-hold	boolean optional
-rise	boolean optional
-fall	boolean optional
-from fl	list optional
-to tl	list optional
-through thl	list optional duplicates -merge

Must specify one of from_list, to_list or through_list

24. set_fanout_load	RESULT: 1/0
value	float >= 0 **required**
port_list	list **required**

25. set_input_delay	RESULT: 1/0
-clock	list optional
-clock_fall	boolean optional
-level_sensitive	boolean optional
-rise	boolean optional
-fall	boolean optional
-max	boolean optional
-min	boolean optional
-add_delay	boolean optional
delay_value	float **required**
port_pin_list	list **required**
-network_latency_included	boolean optional
-source_latency_included	boolean optional

26. set_input_transition	RESULT: 1/0
-rise	boolean optional
-fall	boolean optional
-min	boolean optional
-max	boolean optional
transition	float >= 0 **required**
port_list	list **required**
-clock	list optional
-clock_fall	boolean optional

27. set_load	RESULT: 1/0
-min	boolean optional
-max	boolean optional
-pin_load	boolean optional
-wire_load	boolean optional
-subtract_pin_load	boolean optional
value	float >= 0 **required**

objects	list **required**
28. set_logic_dc	RESULT: 1/0
port_list	list **required**
29. set_logic_one	RESULT: 1/0
port_list	list **required**
30. set_logic_zero	RESULT: 1/0
port_list	list **required**
31. set_max_area	RESULT: 1/0
area_value	float >= 0 **required**
32. set_max_capacitance	RESULT: 1/0
capacitance_value	float >= 0 **required**
object_list	list **required**
33. set_max_delay	RESULT: 1/0
-rise	boolean optional
-fall	boolean optional
-from fl	list optional
-to tl	list optional
-through thl	list optional duplicates -merge
delay_value	float **required**
34. set_max_fanout	RESULT: 1/0
fanout_value	float >= 0 **required**
object_list	list **required**
35. set_max_time_borrow	RESULT: 1/0
delay_value	float >= 0 **required**
object_list	list **required**
36. set_max_transition	RESULT: 1/0
transition_value	float >= 0 **required**
object_list	list **required**
37. set_min_capacitance	RESULT: 1/0
capacitance_value	float >= 0 required
object_list	list **required**
38. set_min_delay	RESULT: 1/0
-rise	boolean optional
-fall	boolean optional

-from fl	list optional
-to tl	list optional
-through	list optional duplicates -merge
delay_value	float **required**

39. set_multicycle_path	RESULT: 1/0
-setup	boolean optional
-hold	boolean optional
-rise	boolean optional
-fall	boolean optional
-start	boolean optional
-end	boolean optional
-from fl	list optional
-to	list optional
-through thl	list optional duplicates -merge
path_multiplier	integer **required**

40. set_operating_conditions	RESULT: 1/0
-analysis_type	enum {single bc_wc on_chip_variation} optional
-library lib	list optional
-max mxv	string optional
-min mnv	string optional
-max_library mxl	list optional
-min_library mnl	list optional
condition	string optional

41. set_output_delay	RESULT: 1/0
-clock	list optional
-clock_fall	boolean optional
-level_sensitive	boolean optional
-rise	boolean optional
-fall	boolean optional
-max	boolean optional
-min	boolean optional
-add_delay	boolean optional
delay_value	float **required**
port_pin_list	list **required**
-network_latency_included	boolean optional
-source_latency_included	boolean optional

42. set_port_fanout_number	RESULT: 1/0
fanout_number	integer 0 to 100000 **required**
port_list	list **required**

43. set_propagated_clock	RESULT: 1/0

object_list	list **required**

44. set_resistance	RESULT: 1/0
-min	boolean optional
-max	boolean optional
value	float >= 0 **required**
net_list	list **required**

45. set_wire_load_min_block_size	RESULT: 1/0
size	float >= 0 **required**

46. set_wire_load_mode	RESULT: 1/0
mode_name	enum {top enclosed segmented} **required**

47. set_wire_load_model	RESULT: 1/0
-name wln	string **required**
-library	list optional
-min	boolean optional
-max	boolean optional
object_list	list optional

48. set_wire_load_selection_group	RESULT: 1/0
-min	boolean optional
-max	boolean optional
-library lv	list optional
group_name gn	string **required**
object_list	list optional

Notes:
+ Sometimes, it is convenient for an option to be a list, even if it is really only a single object. Example: -clock in all_outputs.

49. set_hierarchy_separator	RESULT: 1/0
separator	enum {/, @, ^, #, . |}

50. create_generated_clock	RESULT: 1/0
-name value	string optional
-source name	list **required**
-divide_by divide_factor	float >=0 optional
-multiply_by multiply_fac	float >=0 optional
-duty_cycle percent	float >=0 ,<=100 optional
-invert	boolean optional
-edges edge_list	list optional

-edge_shift	boolean optional
port_pin_list	list **required**
-add	boolean optional
-master_clock	list optional

Notes:

-divide_by and multiply_by are mutually exclusive
-Must specifies one of these options: -edges or -divide_by or -multiply_by

51. set_clock_gating_check

-setup setup_value	float, >= 0
-hold hold_value	float, >= 0
-rise	boolean optional
-fall	boolean optional
-high	boolean optional
-low	boolean optional
object_list	list **required**

Notes:
Must specify -setup, -hold, -high, or -low
Only one of -high and -low

52. set_data_check	RESULT: 1/0
-from from_object	list optional
-to to_object	list optional
-rise_from rise_from_object	list optional
-fall_from fall_from_object	list optional
-rise_to rise_to_object	list optional
-fall_to fall_to_object	list optional
-setup	boolean optional
-hold	boolean optional
-clock clock_object	list optional
value	float **required**

53. set_max_dynamic_power	RESULT: 1/0
power	float required
-unit	enum { GW MW KW W mW uW nW pW fW aW}

54. set_max_leakage_power	RESULT: 1/0
power	float **required**
-unit	enum { GW MW KW W mW uW nW pW fW aW}

55. set_min_porosity	RESULTS: 1/0
value	float **required**

design_list string optional

Note: porosity value must be a real number between 0 and 90

Appendix B

Example of EDA application

```
#!/bin/sh
#
# This is a simple example of EDA application
#
# It contains
#
# - EDA callback procedure "callback_simple_example"
# - main program (register callback and parse file)
#
# The callback procedure
#   - prints parameter values for SDC commands
#     - create_clock
#     - set_input_delay
#   - returns parameter values for SDC Object Access
Functions
#     (here without searching in Design Data Base)
#     - get_clocks
#     - get_ports
#
#\
exec tclsh "$0" "$@"

# include parser engine

source [file join [file dirname [info script]]
sdcparsercore.tcl]

# callback procedure

proc callback_simple_example {command parsing_result} {

    # put reference to data structure after parsing

    upvar $parsing_result res

    # Switch on command type

    switch -- $command {


        create_clock -
        set_input_delay  {
            puts "Command: $command"
```

```
            foreach arg [array names res] {
                puts "  Argument $arg = $res($arg)"
            }
            puts ""
            return ""
        }


        get_clocks -
        get_ports {
            return $res(patterns)
        }

        default {
        }
    }
}

# main program

sdc::register_callback callback_simple_example

sdc::parse_file [lindex $argv 0]
```