

3

Using ATM Generator Components

This chapter includes the following information about using ATM generator components to generate ATM cell streams with specific physical interfaces.

Topic	Page
ATM Generator Components: Operating Principles	page 3-2
Instantiating ATM Generator Components	page 3-9
Defining Extra TAG Bytes	page 3-12
External Files Used by ATM Generator Components	page 3-14
Using ATM Generator Components	page 3-18
Additional Generator Components	page 3-66
Generic Interface Component	page 3-67
Utopia Components	page 3-71
Utopia Manager Components	page 3-94
Utopia Watchdog Components	page 3-98

ATM Generator Components: Operating Principles

ATM generator components are programmable devices you use to generate ATM cell streams with specific physical interfaces. The ATM_COMPONENTS library provides basic components for ATM cell generation programming, physical interfaces, bit error insertion, and event generation. You can combine these different components to build new hierarchical components that provide additional testbench functionality. In addition, the library includes some components that have been built from a combination of other components to provide ready-to-use features (such as generation + interface) more conveniently. There is no restriction on the combination of components, although not every combination would necessarily be suitable for your needs. In such a case, you can build your own combination from the basic blocks provided.

Example 3-1 illustrates the VHDL declaration of an ATM cell generator with bus interface. Following the illustration, are detailed descriptions of common principles of operation pertaining to naming conventions, generic parameters, and inputs and outputs.

```

component ATM_BYTE_GENERATOR
  generic (COMMAND_FILE : string;
          DISPLAY_FILE  : string := "";
          DUMP_FILE     : string := "";
          DISPLAY       : boolean := true;
          EMPTY_CELL    : T_ATM_CELL := C_UNASSIGNED_CELL;
          BUFFER_SIZE   : natural := 50;
          HTAG_NBIT     : natural := 0;
          TTAG_NBIT     : natural := 0);
  port (DATA : out std_logic_vector;
        START_OF_CELL : out std_logic := '0';
        HEC : out std_logic := '0';
        PAYLOAD : out std_logic := '0';
        HTAG : out std_logic := '0';
        TTAG : out std_logic := '0';
        BYTE_INDEX : out natural := '0';
        CLOCK : in std_logic;
        EOG : out boolean := false);
end component;

```

Example 3-1 VHDL declaration of an ATM cell generator with bus interface

Naming Conventions

Component Names

All generator names are built using this convention:

<OBJECT1>_GENERATOR_<OBJECT2>

Example:

ATM_CELL_GENERATOR_UTOPIA_TX_M_1

Configuration Names

The ATM_COMPONENTS library includes VHDL configurations for each component. You must include these configurations in VHDL configuration clauses you place either in the declarative part of the testbench architecture or in a separate configuration section, just like any other VHDL component being instantiated.

Use any of these three configurations for each component to apply to your needs without modifying the testbench architecture:

- **C_<COMPONENT_NAME>_0**—use for the empty configuration when the component needs to be disabled.
- **C_<COMPONENT_NAME>**—use for the normal configuration to generate frames from the COMMAND_FILE.
- **C_<COMPONENT_NAME>_2**—use for the file reader configuration to generate frames from the DUMP_FILE.

Example:

C_ATM_CELL_GENERATOR_UTOPIA_TX_M_1

See “Instantiating ATM Generator Components” on page 3-9 for an example of this type of configuration clause.

Common Generic Parameters

Because of the large variety of physical interfaces, the generic parameters of generator components differ from one component to another. All generators, however, use the set of generic parameters explained in the following subsections.

COMMAND_FILE

This is the name of the file containing commands that program the component. This file is mandatory when using the configuration **C_<COMPONENT_NAME>**, and must be written by you either when creating the configuration or by adapting an existing file (for example, a

command file taken from the Testbench layer). There is no parameter default value for `COMMAND_FILE`; this parameter is mandatory even if the component is configured in file reader mode (`C_<COMPONENT_NAME>_2`) in which case no use of this file is made.

DISPLAY_FILE

This is the name of the file where the simulation results are displayed. When set to `<filename>`, all simulation results, except warning and error messages generated from parsing, are written in filename, even when parameter `DISPLAY` is set to false (see the information in the section `DISPLAY`, below).

The parameter default value is an empty string (`""`), meaning that no display file is produced during simulation.

DUMP_FILE

This is the name of the file where the results of `DUMP` commands are written if a `DUMP` command is used in the generation command file when the normal configuration `C_<COMPONENT_NAME>` is used. The `DUMP_FILE` also can be the name of the ASCII file containing hexadecimal characters to output as ATM Cells when the component is configured in file reader mode configuration `C_<COMPONENT_NAME>_2`.

The parameter default value is an empty string (`""`), meaning that no data is dumped during simulation (even if a `Dump` command is used in the generation command file).

DISPLAY

This is a boolean parameter controlling the display of results on the screen. This parameter has no effect on the display of results into files (see the information in the section `DISPLAY_FILE`, above).

Nothing is shown on the screen when `DISPLAY` is set to false. The parameter default value is true.

BUFFER_SIZE

This natural parameter is used to tune the size of internal data structures. `BUFFER_SIZE` defines the maximum number of cell sequences, connections, OAM/RM/User periodic flows, traffics, and sources which can be declared at a time in the command file specified by the generic parameter `COMMAND_FILE`. When overload occurs, a warning message is sent to the screen, indicating that you need to raise up `BUFFER_SIZE`, and the simulation is stopped.

When cells are generated in full-stochastic mode, `BUFFER_SIZE` also tunes the depth of the internal FIFO, where the cells are stored before being sent on the multiplex. When overload occurs, a warning message is sent to the screen, indicating that you need to raise up `BUFFER_SIZE` or adjust the traffic specifications to lower values. The simulation is not stopped; however, the extra cells are lost.

The parameter default value is 50.

Clock Signal

All generators are driven by a clock signal called `CLOCK`. One data object (of type depending on the physical interface) is generated on each rising edge of the clock signal.

The clock signal must be generated in the VHDL testbench, either using the `CLOCK_GENERATOR` component or by writing a simple VHDL process. There is no constraint on this signal—you can build it to your needs according to the application (asynchronous signal, gapped clock, etc.).

End of Generation Signal

End Of Generation (EOG) is a boolean set to 'true' as soon as the last command of the generator's command file has been processed, and stays set to 'true' for the remainder of the simulation.

Figure 3-1 illustrates the EOG signal in the case where a string interface which is set to true when the last cell defined in `COMMAND_FILE` is output and remains set to true for the rest of the simulation.

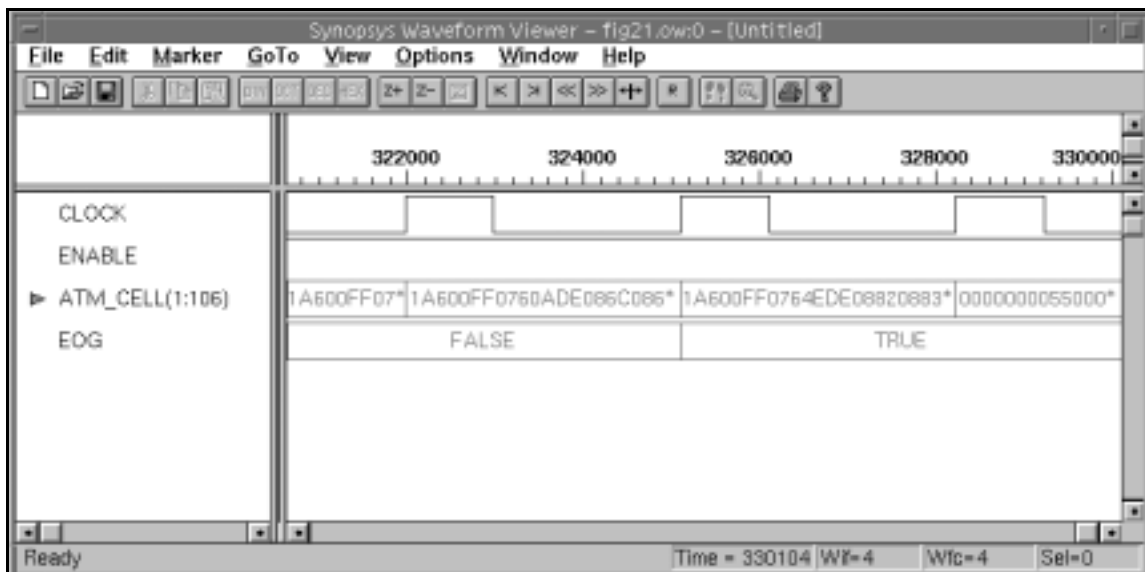


Figure 3-1 End of Generation (EOG) for a string interface

Figure 3-2 illustrates the EOG signal in the case where a byte interface which is set to true when the first byte of the last cell defined in `COMMAND_FILE` is output and remains set to true for the rest of the simulation.

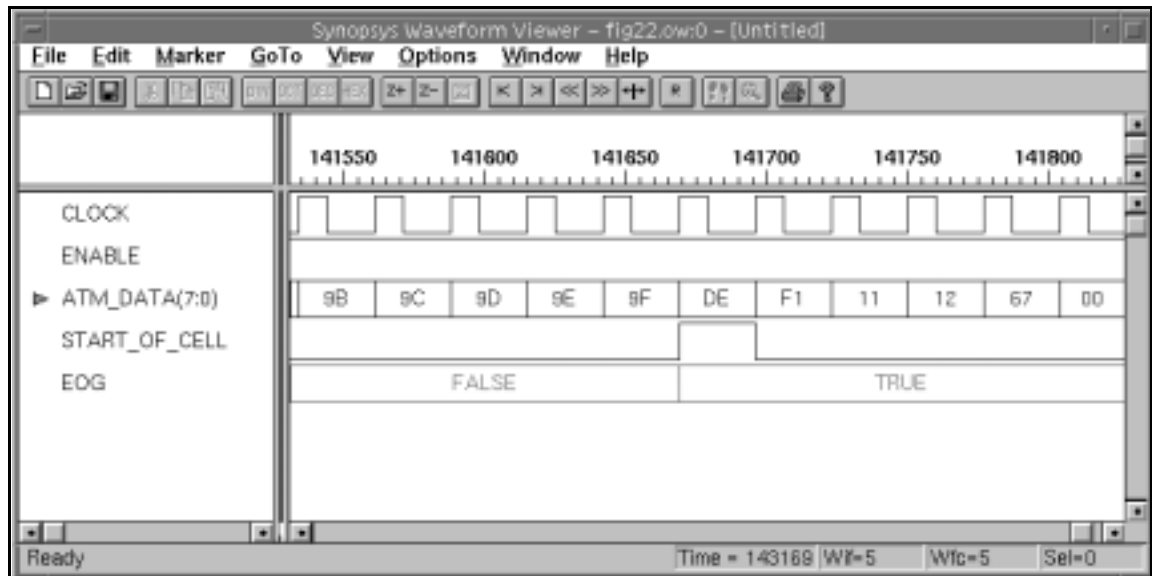


Figure 3-2 End of Generation (EOG) in the case of a byte interface

Example:

Example 3-2 provides a sample of the use of the EOG signal to stop a simulation. The example shows the body of a testbench architecture instantiating one ATM_BYTE_GENERATOR component and one CLOCK_GENERATOR component. ENABLE is a std_logic signal connected to the ENABLE input port of the CLOCK_GENERATOR component, controlling the value on the output port CLOCK. When the EOG output port of the ATM_BYTE_GENERATOR component is set to true (the last command of the command file has been processed, and the last programmed cell is being generated), then ENABLE is set to '0,' forcing the CLOCK output port to '0'. None of the ATM_BYTE_GENERATOR and CLOCK_GENERATOR components have events to process; therefore, the simulation discontinues.

Note: Only the first byte of the last cell is output in this example.

```
architecture A of DEMO1 is
    ...
begin
    ATM_GEN : ATM_BYTE_GENERATOR
        generic map (COMMAND_FILE => "generator1.cmd",
                    DISPLAY_FILE  => "demo1.out",
                    DUMP_FILE     => "generator1.dmp")
        port      map (DATA        => ATM_DATA,
                    START_OF_CELL => START_OF_CELL,
                    CLOCK         => CLOCK,
                    EOG           => EOG);
    BIT_CLK : CLOCK_GENERATOR
        generic map (PERIOD => 27 ns,
                    WIDTH  => 10 ns,
                    EDGE   => '1')
        port      map (CLOCK => CLOCK,
                    ENABLE => ENABLE);

    ENABLE <= '1' when not EOG else '0';
end A;
```

Standard Instantiation Statement

Condition Stopping the Simulation

Example 3-2 Using the EOG signal to stop a simulation

Instantiating ATM Generator Components

Instantiating an ATM Workbench component is similar to instantiating any VHDL component. The steps you use to provide VHDL requirements are graphically summarized in Figure 3-3 and detailed in the procedure below.

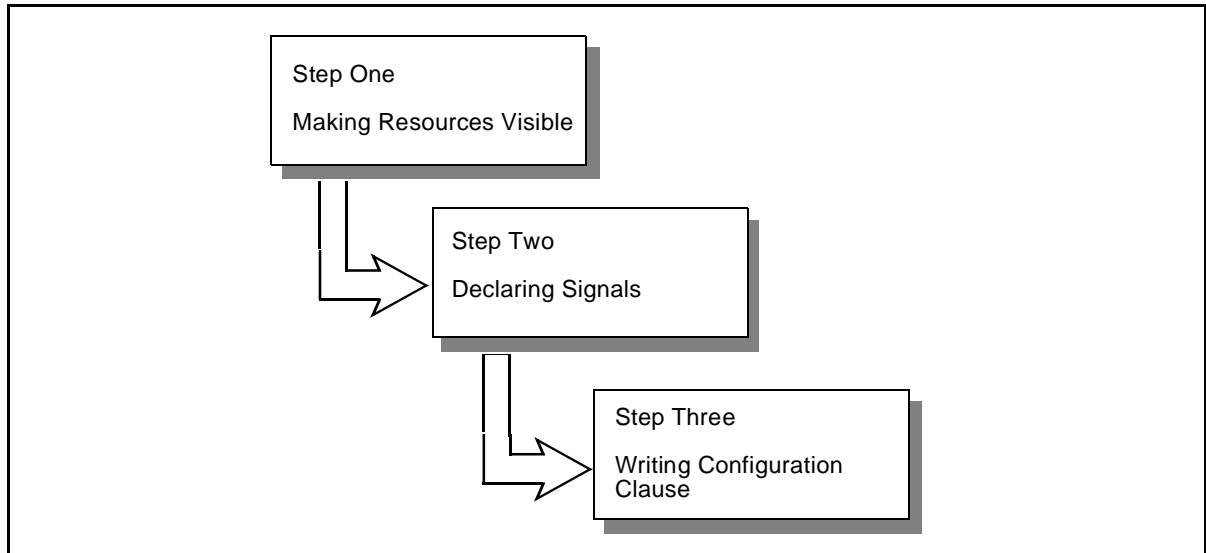


Figure 3-3 Instantiating components—providing VHDL requirements

Example 3-3 illustrates a sample of a VHDL testbench where an `ATM_BYTE_GENERATOR` is instantiated. As the example shows, you must follow this procedure to provide several VHDL requirements before performing instantiation.

To provide the VHDL requirements,

1. Write *library* and *use* clauses to make VHDL resources visible.

The clause below makes all components in `ATM_COMPONENTS` visible.

```
library ATM_COMPONENTS;
use ATM_COMPONENTS.ATM_COMPONENTS.all;
```

2. Declare signals associated with the formal ports of instantiated components.

In the example, the bus width is set to '1' to generate a bit stream.

```
signal BIT_STREAM : std_logic_vector(0 downto 0);
```

3. Write a *configuration* clause to associate a VHDL entity and architecture to the instantiated components.

In the example below, this instance of the ATM_BYTE_GENERATOR component is called CELL_TX.

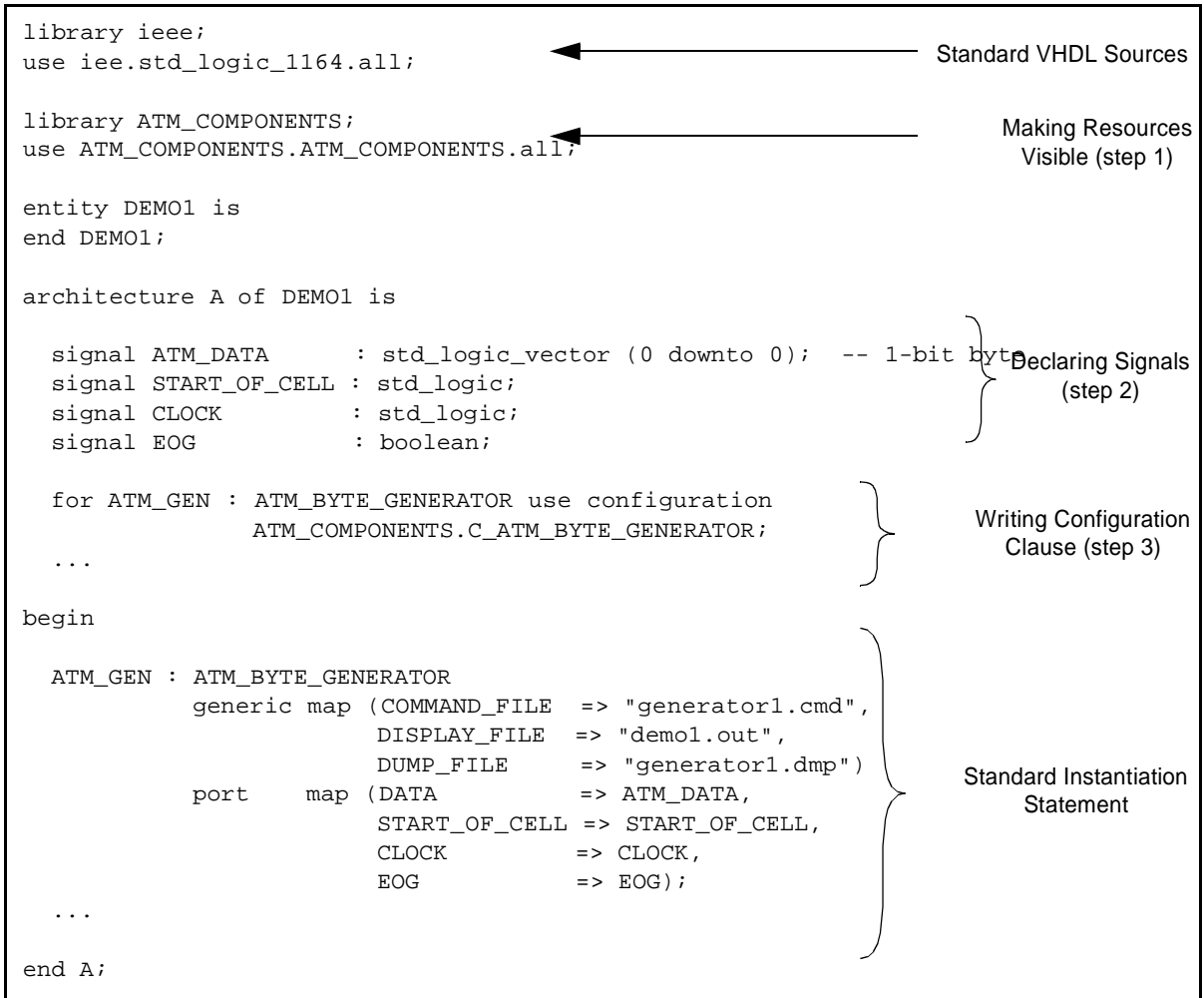
```
for CELL_TX : ATM_BYTE_GENERATOR use configuration
  ATM_COMPONENTS.C_ATM_BYTE_GENERATOR;
```

Note: The configuration of the ATM_BYTE_GENERATOR component occurs either in the declaration part of the testbench architecture or in a separate configuration section, just like any other VHDL component being instantiated.

Example 3-3 shows the following information about component instantiation:

- Name of the command file is `generator1.cmd` (COMMAND_FILE in the example).
- Results are shown on the screen (because the DISPLAY parameter is omitted; hence, left to its default value true) and written to the file `demo1.out` (DISPLAY_FILE in the example).
- If a Dump is programmed in the command file, then data is dumped into the file `generator1.dmp` (DUMP_FILE in the example).
- DATA is the name of the bit stream bus.

See Demo1 for a complete example of, and detailed information on, component instantiation.



Example 3-3 Sample VHDL instantiation of ATM_BYTE_GENERATOR from demo1

Defining Extra TAG Bytes

You can define and program extra fields in front of and following the standard 53-byte ATM cell, which is useful for accommodating user-defined internal cell formats. The components contain generic parameters to define the tag bytes' length, according to the abstraction level, by number of bits (HTAG_NBIT, TTAG_NBIT in case of physical interfaces such as ATM_BYTE_GENERATOR) or string length (HTAG_LENGTH, TTAG_LENGTH in case of string level components such as ATM_CELL_GENERATOR).

You can set the values of the TAG bytes in the command files using the SET_SEQUENCE_ATM_TAG, SET_CONNECTION_TAG and SET_HTAG, SET_TTAG commands. Both the header and trailer TAG bytes may have up to 200 characters, thus 800 bits.

The maximum cell length you are able to set is:

$$100 (\text{HTAG_LENGTH}) + 106 (\text{ATM_CELL_LENGTH, constant defined in the ATM_DATA package}) + 100 (\text{TTAG_LENGTH})$$

equal to a total of 306 characters, thus:

$$400 (\text{HTAG_NBIT}) + 424 (\text{ATM_CELL_LENGTH_B, constant defined in the ATM_DATA package}) + 400 (\text{TTAG_NBIT})$$

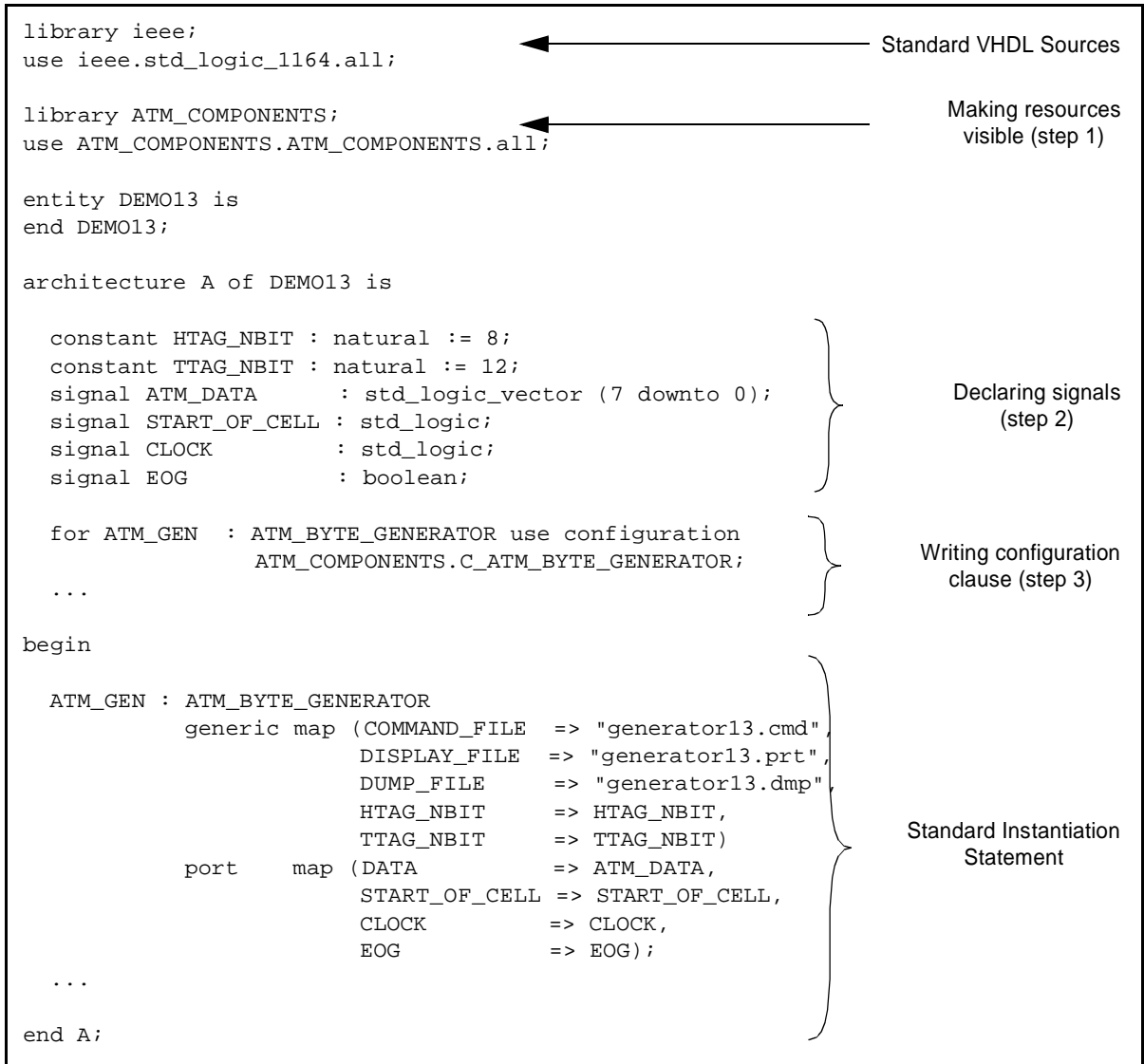
equal to a total of 1224 bits.

Note: In the case of using an ATM_BYTE_GENERATOR component, if the generic parameters HTAG_NBIT and TTAG_NBIT are set to a non-nibble length (multiple of 4) such as 11, then printing format problems may occur with the display on the screen, and with the display and dump files since data are read/written as hexadecimal characters.

Example:

Example 3-4 shows the instantiation of an ATM_BYTE_GENERATOR component defining tag bytes length.

- Name of the command file is `generator13.cmd` (COMMAND_FILE).
- Results are shown on the screen (because the DISPLAY parameter is omitted, hence left to its default value 'true') and written to the file `generator13.prt` (DISPLAY_FILE).
- If a Dump command is programmed in the command file, then data is dumped into the file `generator13.dmp` (DUMP_FILE).
- Eight extra bits are added at the beginning of the cell (HTAG_NBIT), and 12 bits are added at the end (TTAG_NBIT).
- The bus name is ATM_DATA (8-bit width).



Example 3-4 Instantiation of an ATM_BYTE_GENERATOR component defining tag bytes length

A complete demo of this capability is provided in demo13 in the ATM Testbench layer.

External Files Used by ATM Generator Components

ATM generator components use four categories of external files:

- Command file
- Display file
- Dump file
- Source file

Command File

One command file is mandatory for each generator even if the generator is used in file reader mode (configuration **C_<COMPONENT_NAME>_2**) in which case, no use of this file is made. Command files are normal ASCII text files.

Note: Although the command syntax is similar to VHDL procedure call syntax, the command file is not a VHDL design unit and must not be compiled. Commands are interpreted during simulation by the generator.

Command file parsing is NOT performed in one pass. Each time a Transmit command (TRANSMIT or TRANSMIT_MIX) is reached, the parsing is stopped and the programmed sequence of cells is generated. Once the last cell of the sequence is sent over, then parsing starts again from the first command following the TRANSMIT command up to the next TRANSMIT command, and so on.

Note: PRINT and STATUS commands are executed during the parsing (when they are read). They do not require a TRANSMIT command in order to perform an action.

Detailed syntactic rules are given in Chapter 2, Using ATM Generator Components, under Command File Syntax Rules. The command set is detailed in the ATM Workbench Command Reference Manual.

Display File

Simulation results produced by ATM generators are stored in the file indicated by the DISPLAY_FILE parameter. Results are stored in this file as they appear. The parameter DISPLAY is used for screen output only. Example 3-5 illustrates the file contents as a result of including the Print commands in the command file.

```

-----
--                               ATM CELL GENERATOR
--   Generation of an AAL1 Cell stream on a given connection
--   VPI/VCI = DEF/1111
--   Starting with 3 empty cells
--   Specific payload in cell number 11
-----
Time 0 NS - Cell Count : 1  Transmit/User/Empty : 1/1/1 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|000|0000|0|0|55|0|0|00000000....00000000

Time 11448 NS - Cell Count : 2  Transmit/User/Empty : 1/2/2 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|000|0000|0|0|55|0|0|00000000....00000000

Time 22896 NS - Cell Count : 3  Transmit/User/Empty : 1/3/3 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|000|0000|0|0|55|0|0|00000000....00000000

Time 34344 NS - Cell Count : 4  Transmit/User : 1/4 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|DEF|1111|1|0|67|0|0|00010203....2B2C2D2E

Time 45792 NS - Cell Count : 5  Transmit/User : 1/5 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|DEF|1111|1|0|67|1|7|2F303132....5A5B5C5D
.
.
Time 103032 NS - Cell Count : 10  Transmit/User : 1/10 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|DEF|1111|1|0|67|6|3|1A1B1C1D....45464748

Time 114480 NS - Cell Count : 11  Transmit/User : 1/11 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|DEF|1111|1|0|67|7|4|AAAAAAAA....AAAAAAAA

Time 125928 NS - Cell Count : 12  Transmit/User : 1/12 - AAL1
|VPI|VCI|PTI|CLP|HEC|SN|SNP|PAYLOAD
|DEF|1111|1|0|67|0|0|78797A7B....A3A4A5A6

```

} User Comments
 } Empty Cells
 } User Cells
 } User Cell
 } Cell With Specific Payload
 } User Cell

Example 3-5 Display file contents example

Dump File

You can program generators to dump generated data into a dump file during simulation. Data is dumped as text in hexadecimal format, as follows:

One cell per line = 106 characters + extra tag bytes if existing

You can use a dump file for several applications, including

When the end of the source file is reached, the generator loops back to the first character of the file.

Any dump file generated by a Telecom Workbench component may be used as a source file. This enables, for example, the mapping of a true ATM-compliant stream into a SONET path. The ATM stream can be dumped by an ATM Workbench component into a file which is then used as a source for the SONET Workbench generator component, which is described in detail in the SONET Workbench Programmer's Solution Guide.

Using ATM Generator Components

ATM generator components are described in the following sections. For more information on the complete set of ATM Workbench components, please refer to the ATM Workbench Command Reference Manual.

ATM_CELL_GENERATOR

Usage:

The ATM_CELL_GENERATOR component is the basic block for all other ATM generator components.

This component generates cells and outputs them on ATM_CELL (generic width). One cell is generated and output at each rising edge of CLOCK (the bus clock).

Component Description:

```
component ATM_CELL_GENERATOR
  generic (COMMAND_FILE : string;
          DISPLAY_FILE  : string    := "";
          DUMP_FILE     : string    := "";
          DISPLAY       : boolean   := true;
          EMPTY_CELL    : T_ATM_CELL := C_UNASSIGNED_CELL;
          BUFFER_SIZE   : natural   := 50;
          HTAG_LENGTH   : natural   := 0;
          TTAG_LENGTH   : natural   := 0;
          ASYNCH_MUX    : boolean   := false);
  port (ATM_CELL : out string;
        I        : out time      := 0 ns;
        L        : out time      := 0 ns;
        CLOCK    : in  std_logic;
        EOG      : out boolean := false);
end component;
```

Parameters:

Parameter/Port Name	Type	Description	Default Value
Parameters			
COMMAND_FILE	string	Name of the file containing commands programming the component. As there is no parameter default value, this parameter is always mandatory even if the component is configured in file reader mode (C_<COMPONENT_NAME>_2) in which case, no use of this file is made.	None
DISPLAY_FILE	string	Name of the file where the simulation results are to be displayed. When set to <filename>, all simulation results, except warning and error messages generated by the parsing of COMMAND_FILE, are written in <filename>, even when parameter DISPLAY is set to false. By default, no display file will be produced during simulation.	Empty string (" ")

Parameter/Port Name	Type	Description	Default Value
DUMP_FILE	string	When the normal configuration (C_<COMPONENT_NAME>) is used, this parameter is the name of the file where the results of Dump commands will be written, if a Dump commands are used in COMMAND_FILE. By default, no dump file will be produced during the simulation. When the component is configured in file reader mode (C_<COMPONENT_NAME>_2), this parameter is the name of the ASCII file containing hexadecimal characters to output as ATM cells. By default, an empty filename is specified.	Empty string ("")
DISPLAY	boolean	Controls the display of results on the screen. When set to true, all simulation results are displayed on screen else, when set to false, only the error and warning messages generated by the parsing of COMMAND_FILE are displayed. The parameter value has no effect on the display of results into DISPLAY_FILE.	true
EMPTY_CELL	T_ATM_CELL	Sets the coding of empty cells generated when no user cells are defined (determinist generation) or available (stochastic generation).	C_ UNASSIGNED _CELL

Parameter/Port Name	Type	Description	Default Value
BUFFER_SIZE	natural	<p>Tunes the size of internal data structures; it defines the maximum number of cell sequences, connections, OAM/RM/User periodic flows, traffics, and sources which can be declared at a time in COMMAND_FILE. When overload occurs, a warning message is sent to the screen, indicating that you need to raise up BUFFER_SIZE, and the simulation is stopped.</p> <p>When cells are generated in full-stochastic mode, BUFFER_SIZE also tunes the depth of the internal FIFO where the cells are stored before being sent on the multiplex. When overload occurs, a warning message is sent to the screen, indicating that you need to raise up BUFFER_SIZE or adjust the traffic specifications to lower values. The simulation is not stopped; however, the extra cells are lost.</p>	50
HTAG_LENGTH	natural	String length (number of characters) of the HTAG tag field.	0
TTAG_LENGTH	natural	String length (number of characters) of the TTAG tag field.	0

Parameter/Port Name	Type	Description	Default Value
ASYNCH_MUX	boolean	<p>Useful only for the full-stochastic generation of cells performed using DEFINE_TRAFFIC_CBR, DEFINE_TRAFFIC_VBR, and DEFINE_TRAFFIC_ABR commands (Command Reference Manual). When set to false (synchronous cell generation), at each rising edge of the multiplex clock, if the time at which a cell was expected is lower than the current time, then the cell is output and a draw is performed to have the time delay up to the next cell generation. This time delay is then added to the current time (VHDL function now). In that case, the cells are output on the multiplex without violation with the traffic parameters (PCR, SCR, etc.) set with the traffic definition command.</p> <p>When set to true (asynchronous cell generation reflecting the real-world case), the different connections are considered to be independent with the multiplex clock. At each rising edge of the multiplex clock, if the time at which a cell was expected is lower than the current time (VHDL function now), then the cell is output and a draw is performed to have the time delay up to the next cell generation. This time delay is then added to the time at which the cell was expected. In such a case, the cells are not output on the multiplex in exact conformance with the traffic parameters (PCR, SCR, etc.) set with the traffic definition command but at the line rate.</p>	false

Parameter/Port Name	Type	Description	Default Value
Ports			
ATM_CELL	out string	(Mandatory) output data bus (generic width) on which is output one new cell at each rising edge of CLOCK.	None
I, L	out time	(Optional) GCRA parameters attached to one specific connection. (I,L) represents the couple (PCR, CDVT_PCR) for CBR or ABR traffic, and (PCR, CDVT_PCR) or (SCR, CDVT_SCR) for VBR traffic. Values on these two ports are set using DEFINE_I_L commands.	0 ns
CLOCK	in std_logic	(Mandatory) bus clock. At each rising edge of CLOCK, one new cell is generated and output on ATM_CELL.	None
EOG	out boolean	(Optional) End Of Generation signal is set to true when the last cell defined in COMMAND_FILE is output and remains to true for the rest of the simulation.	false

Configurations Description:

Three configurations are available for this component, enabling coverage of three different configuration needs without modifying the testbench architecture.

Configuration	Description/Use
C_ATM_BYTE_GENERATOR_0	Empty configuration to use when the component needs to be disabled.
C_ATM_BYTE_GENERATOR	Normal configuration, for which the ATM cells to generate are performed by the commands in COMMAND_FILE.
C_ATM_BYTE_GENERATOR_2	File reader configuration, for which the ATM cells to generate are read from the file DUMP_FILE.

