VQM Extractor and Language Functional Description

Version 2.0

June 2, 2005

by

Altera Software

Table of Contents:

1. Overview

VQM, or “Verilog Quartus Module” is a restricted form of the Verilog language standard. Its purpose is to allow for fast parsing of technology-mapped (a.k.a. atom-level or WYSIWYG) netlists into the Quartus® software. VQM netlists are produced either by external or 3rd party synthesis tools for input to Quartus, or by Quartus itself to store a mapped file for later re-use in place and route. VQM output from Quartus is usable by 3rd parties or other external tools.

The parser for VQM files and for general Verilog files is not the same code. This allows VQM files to be processed quickly. In general, however, the input is assumed to be program generated, and both syntactic and semantic error reporting is minimal. A human user should generally write their design in behavioural Verilog, VHDL, AHDL or schematic.

This document describes the VQM file format and its use.

2. Input File Descriptions

2.1 Grammar

The VQM extractor can read a subset of the Verilog design language. The formal grammar for VQM is given in standard Backus-Naur Form (BNF)[1].

```
Design          ::=   Module*
Module          ::=   Header  Decl*  Stmt*  Footer
Header          ::=   module regular_id ( [ IdentList ] );
Decl            ::=   PinType  IdentList ;    |
                      PinType  \[ intConstant : intConstant \] IdentList ;
Footer          ::=   endmodule
Stmt            ::=   AssignStmt | TriStmt | WysiwygStmt | ComponentStmt | DefparamStmt
AssignStmt      ::=   assign Ident = RValue ;
RValue          ::=   Ident | ~ Ident | bitConstant | Concat | nonNegConstant
Concat          ::=   { IdentList }
TriStmt         ::=   assign Ident = Ident ? Ident : zConstant ;
WysiwygStmt     ::=   wysiwyg_id Ident (  ConnectionList ) ;
ComponentStmt   ::=   regular_id Ident ( ConnectionList ) ;
DefparamStmt    ::=   defparam Ident.regular_id = pValue ;
pValue          ::=   stringConstant | intConstant
ConnectionList  ::=   Connection [, ConnectionList ]
Connection      ::=   .regular_id(Ident) | .escaped_id(Ident) |
                      .regular_id(Concat) | .escaped_id(Concat)
IdentList       ::=   Ident [, IdentList ]
Ident           ::=   regular_id [ \[ intConstant \] ] | escaped_id
PinType         ::=   input | output | inout | wire
```

[1] For further information on BNF look on google, or any compiler or programming languages textbook.

Note that many of the grammatical features in VQMX are supported only for specific reasons. For example, the TriStmt is only intended for use in implementing a tristate buffer, and thus does not have the full flexibility that one would expect in Verilog or any other programming language.

Synplicity, who is the primary provider of VQMs to Quartus generally writes their VQM at the bit-level, so we would recommend that any other VQM writers conform to this strategy.

2.2 Tokens

Reserved Keywords in VQM

The following are reserved words in Verilog that are also honored in VQM:

`module`, **`endmodule`**, **`assign`**, **`input`**, **`output`**, **`inout`**, **`defparam`**

The remaining reserved words in Verilog, though not used in the VQM grammar, are treated as reserved words in VQM to maintain VQM as a legal subset of Verilog. For a list of these reserved words consult the Verilog language reference (LRM) or Quartus II online help.

Reserved Punctuations in VQM

```
, . ( ) [ ] { } = ~ ! ? ; :
```

Other lexical symbols

`wysiwyg_id` : altera-specific wysiwig identifier name
(e.g. stratix_io, stratix_lcell)
`regular_id` : alpha-numeric identifier
[a-zA-Z_][a-zA-Z_0-9$]*
`escaped_id` : any text enclosed by an escaped character and a space.
Note that the character set {':', '|', '', '~', '[', ']'} is reserved for internal processing of Quartus and hence should be omitted from the identifier.*
\[^:|*~\[\]]+white_space
`intConstant` : any integer number
[-]?[1-9][0-9]*|0+
`bitConstant` : representing bit value of 0 or 1
1'b0 or 1'b1
`nonNegConstant` : any non-negative integer number
[1-9][0-9]*
`zConstant` : representing impedance
1'bz
`stringConstant` : any text enclosed by quotation marks
" [.]* "

2.3 Quartus-specific issues

The expected and typical use of VQM is family-specific. The base level atomic definition of a cell in a quartus netlist assumes the definition of a corresponding atom in the family. The term atom is often used in Quartus documentation to refer to a logic element or other logical block on an Altera® device.

Thus regular_id in the grammar must be a recognized element type for the current family, also termed a WYSIWYG or WYSIWYG element. Expressing WYSIWYGs follows a standard format, consisting of a WysiwygStmt to identify the WYSIWYG atom, followed by a set of DefParamStmt to configure the atom.

This is best shown by example, in this case a register described for the Stratix™ family.

```
stratix_lcell my_lcell (.datad(in), .clk(clk), .regout(out));
defparam my_lcell.operation_mode = “normal”;
defparam my_lcell.packed_mode = “false”;
defparam my_lcell.lut_mask = “FF00”;
defparam my_lcell.output_mode = “reg_only”;
```

Further examples will be evident from the complete VQM example shown later in an appendix.

Some further notes:

- Contrary to the Verilog standard, 1’b0, 1’b1 and 1’bz are the only constants supported in VQM. Hence, expressions like 1’b0101 will not be honored in VQM.
- VQM is originally a machine generated design file. The Quartus extractor therefore expects minimal user errors. Syntax errors will often result in a general indication of an error exists on a particular line without going into details on the nature of the error. The parser does not have a strong semantic checking phase.
- All identifiers must satisfy the requirement for identifiers in Verilog.
- Any Verilog language semantic requirement that governs the VQM subset must be satisfied.

2.4 Generating a VQM from Quartus II

To generate a VQM from Quartus II, two commands are required; first to synthesize the design itself, and second to output the VQM netlist. They must be executed in sequence.

```
quartus_map <your_project> -c <your_toplevel_design_name>
quartus_cdb <your_project> -c <your_toplevel_design_name> --vqm=<your_vqm_file>
```

There are other command line options for `quartus_map` and `quartus_cdb` which could be explored using the help option `–h`.

```
quartus_map –h  or  quartus_cdb –h
```

2.5 Reading a VQM into Quartus II

Reading a VQM file into Quartus II simply means specifying it as a user design (<design>.quartus) file or through the user interface. The process is identical to specifying a Verilog or VHDL design file.

3. Future Direction

Other than creating WYSIWYGs for new families, the VQM format is largely unchanged since the 99.06 release of Quartus, and is not expected to change in the immediate future unless there is a compelling need.

Appendix A: Example VQM file

The following is an example of a simple Verilog design and a VQM file generated from Quartus II after synthesizing the design. Note that the '\' character is used to escape all names generated by synthesis, to ensure that synthesis of non-Verilog files cannot generate illegal Verilog names.

INPUT:

```
// Simple LUT and FF in Verilog
module foo(henry, starbucks, batman, sysclock, result);
        input henry, starbucks, batman;
        input sysclock;
        reg intermediate_value;
        output result;

        always@ (posedge sysclock)
        begin
                intermediate_value <= (henry & starbucks) | batman;
        end

        assign result = intermediate_value;
endmodule
```

OUTPUT:

```
module foo (
        batman,
        henry,
        starbucks,
        sysclock,
        result);
input   batman;
input   henry;
input   starbucks;
input   sysclock;
output  result;
wire \batman~padio ;
wire \henry~padio ;
wire \starbucks~padio ;
wire \sysclock~padio ;
wire \result~padio ;
wire \sysclock~combout ;
wire \batman~combout ;
wire \henry~combout ;
wire \starbucks~combout ;
wire intermediate_value;


wire gnd;
wire vcc;

assign gnd = 1'b0;
assign vcc = 1'b1;


stratix_io \sysclock~I (
        .combout(\sysclock~combout ),
        .padio(sysclock));
defparam \sysclock~I .operation_mode = "input";
defparam \sysclock~I .ddio_mode = "none";
defparam \sysclock~I .input_register_mode = "none";
defparam \sysclock~I .output_register_mode = "none";
defparam \sysclock~I .oe_register_mode = "none";
defparam \sysclock~I .input_async_reset = "none";
defparam \sysclock~I .output_async_reset = "none";
defparam \sysclock~I .oe_async_reset = "none";
defparam \sysclock~I .input_sync_reset = "none";
defparam \sysclock~I .output_sync_reset = "none";
defparam \sysclock~I .oe_sync_reset = "none";
defparam \sysclock~I .input_power_up = "low";
defparam \sysclock~I .output_power_up = "low";
defparam \sysclock~I .oe_power_up = "low";

stratix_io \batman~I (
        .combout(\batman~combout ),
        .padio(batman));
defparam \batman~I .operation_mode = "input";
defparam \batman~I .ddio_mode = "none";
defparam \batman~I .input_register_mode = "none";
defparam \batman~I .output_register_mode = "none";
defparam \batman~I .oe_register_mode = "none";
defparam \batman~I .input_async_reset = "none";
defparam \batman~I .output_async_reset = "none";
defparam \batman~I .oe_async_reset = "none";
defparam \batman~I .input_sync_reset = "none";
defparam \batman~I .output_sync_reset = "none";
defparam \batman~I .oe_sync_reset = "none";
defparam \batman~I .input_power_up = "low";
defparam \batman~I .output_power_up = "low";
defparam \batman~I .oe_power_up = "low";

stratix_io \henry~I (
        .combout(\henry~combout ),
        .padio(henry));
defparam \henry~I .operation_mode = "input";
defparam \henry~I .ddio_mode = "none";
defparam \henry~I .input_register_mode = "none";
defparam \henry~I .output_register_mode = "none";
defparam \henry~I .oe_register_mode = "none";
defparam \henry~I .input_async_reset = "none";
defparam \henry~I .output_async_reset = "none";
defparam \henry~I .oe_async_reset = "none";
defparam \henry~I .input_sync_reset = "none";
defparam \henry~I .output_sync_reset = "none";
defparam \henry~I .oe_sync_reset = "none";
defparam \henry~I .input_power_up = "low";
defparam \henry~I .output_power_up = "low";
defparam \henry~I .oe_power_up = "low";

stratix_io \starbucks~I (
        .combout(\starbucks~combout ),
        .padio(starbucks));
defparam \starbucks~I .operation_mode = "input";
defparam \starbucks~I .ddio_mode = "none";
defparam \starbucks~I .input_register_mode = "none";
defparam \starbucks~I .output_register_mode = "none";
defparam \starbucks~I .oe_register_mode = "none";
defparam \starbucks~I .input_async_reset = "none";
defparam \starbucks~I .output_async_reset = "none";
defparam \starbucks~I .oe_async_reset = "none";
defparam \starbucks~I .input_sync_reset = "none";
defparam \starbucks~I .output_sync_reset = "none";
defparam \starbucks~I .oe_sync_reset = "none";
defparam \starbucks~I .input_power_up = "low";
defparam \starbucks~I .output_power_up = "low";
defparam \starbucks~I .oe_power_up = "low";

stratix_lcell \intermediate_value~I (
        .clk(\sysclock~combout ),
        .dataa(\batman~combout ),
        .datab(\henry~combout ),
        .datac(\starbucks~combout ),
        .aclr(gnd),
        .regout(intermediate_value));
defparam \intermediate_value~I .operation_mode = "normal";
defparam \intermediate_value~I .synch_mode = "off";
defparam \intermediate_value~I .register_cascade_mode = "off";
defparam \intermediate_value~I .sum_lutc_input = "datac";
defparam \intermediate_value~I .lut_mask = "EAEA";
defparam \intermediate_value~I .output_mode = "reg_only";

stratix_io \result~I (
        .datain(intermediate_value),
        .padio(result));
defparam \result~I .operation_mode = "output";
defparam \result~I .ddio_mode = "none";
defparam \result~I .input_register_mode = "none";
defparam \result~I .output_register_mode = "none";
defparam \result~I .oe_register_mode = "none";
defparam \result~I .input_async_reset = "none";
defparam \result~I .output_async_reset = "none";
defparam \result~I .oe_async_reset = "none";
defparam \result~I .input_sync_reset = "none";
defparam \result~I .output_sync_reset = "none";
defparam \result~I .oe_sync_reset = "none";
defparam \result~I .input_power_up = "low";
defparam \result~I .output_power_up = "low";
defparam \result~I .oe_power_up = "low";
endmodule
```