As duas linguagens permitem modelagem em nível de portas lógicas, contudo VERILOG é mais indicada que VHDL para uma descrição Low-Level (observer a Figura 1). Justifico essa afirmação tendo em vista que a origem do VERILOG foi primordialmente para modelar e simular portas lógicas. VERILOG possui primitivas nativas ou portas lógicas, nas quais um desenvolvedor pode instanciar sem maiores traumas, quando comparada com VHDL. A dica para modelar Low-Level em VHDL é utilizar os operadores lógicos NOT, AND, NAND, OR, NOR, XOR, XNOR. Abaixo é possível comparar essas primitivas e seus equivalentes em VHDL e VERILOG.

VERILOG possui suporte a duas dúzias de primitivas para modelagem de lógica estrutural. Adicionalmente a essas primitivas, o VERILOG possui suporte a primitivas definidas por usuários, UDP (User-Defined Primitives).

VHDL é muito melhor para descrever hardware em alto-nível.

Pacotes (package) para reuso em projetos. Subprogramas e tipos de dados reutilizados podem ser declarados em um pacote VHDL e reutilizados em outras entidades e arquiteturas. VERILOG possui o equivalente através da diretiva de compilação include " ".

Declaração de configuração em VHDL permite que um projeto em VHDL possua muitas entidades de projetos com arquiteturas diferentes. Esse recurso é muito útil para os projetistas que utilizam VHDL e precisam gerenciar um projeto de alto nível, cujas características podem variar.

Gerenciamento de Bibliotecas. Essas bibliotecas VHDL possuem arquiteturas, entidades, pacotes e configurações já compiladas. Esse recurso é muito útil no gerenciamento de grandes estruturas de projeto.

Tão quanto o fluxo de projeto não requeira uma modelagem em baixo-nível, é preferível o uso de VHDL. Ou seja, se você for um projetista de sistemas em FPGA, use VHDL.

VERILOG é case-sensitive, o que não ocorre com VHDL.

Em VERILOG para usar um componente em um módulo basta fazer uma instância e fazer correto "port_map". Em VHDL, para instanciar um componente é necessário declarar as bibliotecas, os pacotes, a entidade, seguido da arquitetura, para finalmente realizar o port-map dos sinais.

VERILOG possui diversas diretivas de compilação como "timescale", "ifdef", "else", "include". Diretivas de compilação não foram permitidas em VHDL até a especificação VHDL2008. Aqui vale uma observação lembrada pelo engenheiro Ricardo Fialho - "...sempre foi possível colocar código dentro de uma architecture usando procedure e functions...mas só servem para alguns casos...". Valeu Tafas. A lógica abaixo é um bom exemplo de função que implementa a lógica OR bit-a-bit (unária) em VHDL.

https://www.embarcados.com.br/verilog-vs-vhdl/ 11/12/18, 16:50.