Pyrope: A Nicer Jewel

A Comparison and Analysis of Hardware Description Language's

Rashad Kayed

Verilog

- One of the most common hardware description languages is called Verilog.
- Verilog is similar to C language, and is a low level hardware language.
- It is most commonly used when designing and implementing digital circuits at the register-transfer level. Verilog is also important for analog circuits and mixed signal circuits.

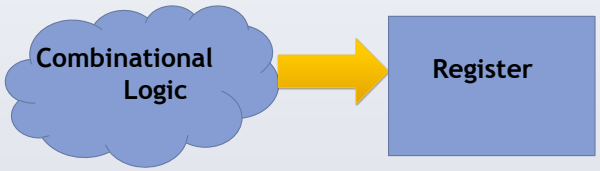
Pyrope

- Pyrope uses some Ruby language aspects, so it was named Pyrope after another similar gemstone.
- Pyrope creates a simpler language for digital architecture by implementing programming constructs. Pyrope's purpose is to maintain the functionality of low level Verilog code, yet also implement a highly expressive language with abstraction capabilities.

Hardware Description Language

- A Hardware Description Language also known as a HDL, is a language used to describe a digital system. A digital system is something like a microprocessor or a flip flop switch.

Pipeline Structure



- One reason Pyrope was developed was to build the pipeline structure on its own. A pipeline structure consists of combinational logic and registers to store the results of that logic.
- Coding in Pyrope takes care of setting up this structure for you, so that the user does not have to keep track of it.
- A block labeled stage in Pyrope code represents the diagram above.

Results & Improvements

- An example of one of the pros of Pyrope is global variable usage, which reduces information programmers need to remember.
- After writing test cases of code using both Verilog and Pyrope it was discovered that Pyrope uses about 25% less lines of code than Verilog, producing a neater and more elegant overall program.
- Advantages of Pyrope include modern language constructs, global type inference, reduction of boilerplate code, and cleaner and simpler programs.

Example

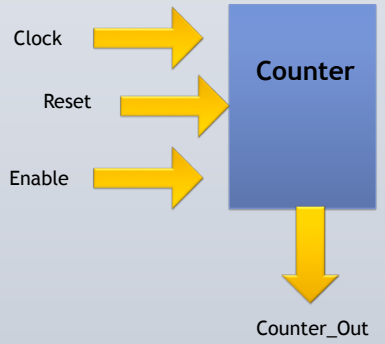
Implementation of a counter in Verilog:

```
module counter(en, clk, reset, amt, x);
input en, clk, reset;
input [3:0] amt;
output [7:0] x;
reg [7:0] x;

always @(posedge clk)
   if(reset) begin
      x = 0;
   end else if(en) begin
      x = x + amt;
   end
endmodule
```

Implementation of a counter in Pyrope:

```
@Counter = 8h0
@Counter = @Counter + Amt if En
```



Acknowledgements

Professor Jose Renau
Haven Skinner