CME 341: For Loops in Verilog

Brian Berscheid

Department of Electrical and Computer Engineering
University of Saskatchewan

Today's agenda

For Loops: Theory

What is a for loop?

- Construct provided by Verilog used to generate a set of related hardware
- Gets parsed and expanded into corresponding Verilog statements by pre-compiler
  - ▷ Saves typing
  - ▷ Makes code easy to read
  - ▷ Makes code easy to modify
  - ▷ May reduce chance of typos in code
- Basic version resides within procedures (`initial` or `always` in CME 341)
  - ▷ Other versions exist and may be covered later in the course (time permitting)
- Need to be careful ... does not perform sequential operations as in other programming languages! It is easy to get confused about this when learning Verilog!

A simple example

```
integer i;
always @ (*)
  for (i = 0; i <= 10; i = i+1)
    data_out[i] = data_in[i];
```

A simple example: expanded by compiler

```
integer i;
always @ (*)
  for (i = 0; i <= 10; i = i+1)
    data_out[i] = data_in[i];
```

```
always @ (*)
  begin
    data_out[0] = data_in[0];
    data_out[1] = data_in[1];
    data_out[2] = data_in[2];
    ...
    data_out[10] = data_in[10];
  end
```

for loop syntax

```
integer loop_var; // declare variable outside of procedure

// inside a procedure (initial or always block):
for ( initialization ; condition ; step )
  statement_to_repeat;
```

- Initialization statement sets initial value of loop variable
- Loop is repeated while condition is true
- After each loop iteration, the loop variable is updated according to the step statement
- Can use `begin ... end` to repeat multiple statements

Key points to remember

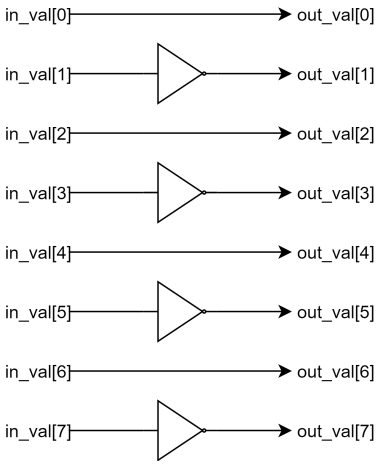
- Loop must sit inside a procedure
- Condition is a test that evaluates to true or false (usually a relational operator)
- No hardware is built corresponding to loop variable and its initialization, updating, or checking against condition
    - ▷ All of that is just used by the pre-compiler to expand the statement to repeat into multiple statements
- No semicolon after for statement

Second example: can you figure out what this does?

```
module my_example (
  input wire [7:0] in_val,
  output reg [7:0] out_val
);
  integer i;
  always @ (*)
    for (i = 0 ; i < 4 ; i = i+1)
      begin
        out_val[2*i]   = in_val[2*i];
        out_val[2*i+1] = ~in_val[2*i+1];
      end
  endmodule
```

Second example: can you figure out what this does?

```
module my_example (
  input wire [7:0] in_val,
  output reg [7:0] out_val
);
  integer i;
  always @ (*)
    for (i = 0 ; i < 4 ; i = i+1)
      begin
        out_val[2*i]   = in_val[2*i];
        out_val[2*i+1] = ~in_val[2*i+1];
      end
  endmodule
```



Sets out_val = in_val, but with odd indices inverted!

Another example

- What does the following code do?

```
module another_example (
  input wire [3:0] in_val,
  output reg [7:0] count
);
integer i;

always @ (*)
for (i = 0; i < 4; i = i + 1)
  count = count + in_val[i];

endmodule
```

Another example

- What does the following code do?

```
module another_example (
  input wire [3:0] in_val,
  output reg [7:0] count
);
integer i;

always @ (*)
for (i = 0; i < 4; i = i + 1)
  count = count + in_val[i];

endmodule
```

Not what was intended! Problems:

- Feedback in combinational circuit (latch!)
- Assigning to a variable multiple times!
    - ▷ What hardware can achieve this??

Remember: We are using Verilog to synthesize hardware! This is not sequential programming!

For Loops: Challenge questions

From last year's midterm

Please try to solve it yourself

Make a module containing an asynchronous circuit[1] that sets the 8-bit output cct_output to be the reverse order of the 8-bit input wire cct_input. For example, if the input is 10100000, then the output should be 00000101.

[1] Asynchronous and combinational have the same meaning in this context.

Another challenge

Please try to solve it yourself

Make a module containing a combinational circuit that accepts a 16-bit input and produces a 4-bit output. The input may be viewed as four hexadecimal digits, with each output bit corresponding to one of the hexadecimal digits. The output bit is set to 1 if the corresponding hexadecimal digit is equal to 4'hA.

Thank you!
Have a great day!