SV Notes #7: Register Transfer Datapaths

New Keywords and Operators

1. **`parameter`**

   Used in a module header to allow a variable parameter to be set when instantiated. For example:

   ```
   #(parameter WIDTH = 8)
   ```

2. `'` - Example: `Q <= '0;`

   The single apostrophe is the **numeric literal fill** operator. This is used to fill an unknown-width signal with as many logic values (`1`, `0`, `X`, or `Z`) as necessary, no matter what its width is.

   If there is a register **`Q`** with its width defined by a **`parameter`**,

   - `Q <= '0` means fill `Q` with as many zeros as necessary
   - `Q <= 'z` means fill `Q` with as many high-impedance states as necessary

Code Listings

Variable-width Registers

```
module register
  #(parameter WIDTH = 8)        // default width is 8 bits
  (input  logic [WIDTH-1:0] D,  // WIDTH parameterizes D and Q
   output logic [WIDTH-1:0] Q,
   input  logic clock,
                reset_L);

  always_ff @(posedge clock,
              negedge reset_L)
    if (~reset_L)
      Q <= '0;  // use the numeric literal fill operator to ensure
    else        //   that Q is filled with 0's regardless of WIDTH
      Q <= D;

endmodule: register
```

Once this module is declared, it can be used in instantiations like in the following module which declares three registers:

```
module three_registers
  #(parameter V = 16);

  logic             clock,    reset_L;
  logic   [3:0] nibble_in, nibble_out;
  logic   [7:0]   byte_in,   byte_out;
  logic [V-1:0] varbit_in, varbit_out;

  // r1 is 4 bits wide
  register #(4) r1 (.D(nibble_in), .Q(nibble_out), .clock, .reset_L);
  //         ^  sets parameter WIDTH = 4 for this instantiation

  // r2 is 8 bits wide
  register      r2 (.D(  byte_in), .Q(  byte_out), .clock, .reset_L);
  //         ^  omits the parameter for the default value (8 in this case)

  // r3 is V bits wide (default V = 16)
  register #(V) r3 (.D(varbit_in), .Q(varbit_out), .clock, .reset_L);
  //         ^  can use module parameter V as an instantiation parameter!

endmodule : three_registers
```