Loops in SystemVerilog

Loops exist in SystemVerilog, but they do not behave like in computer programs. Loops in computer programs are executed sequentially, but it's not possible to define a sequential loop in synthesizable SystemVerilog. Instead, when the loop is synthesized, each “iteration” of the loop will be synthesized into hardware, generating hardware that will perform all iterations in parallel, effectively unrolling the loop. Here's an example of a synthesizable for loop:

```
logic [SIZE-1:0] x;
logic [SIZE-1:0] y;

genvar i;
always_comb begin
    for (i = 0; i < SIZE; i = i + 1) begin
        y = !x;
    end
end
```

Optionally, the for loop can be placed inside a `generate` block. This is required if unique hardware (such as an instance or `always_` block) will be generated for each loop iteration.

```
logic [SIZE-1:0] x;
logic [SIZE-1:0] y;

genvar i;
generate
    for (i = 0; i < SIZE; i = i + 1) begin
        always_comb begin
            y[i] = !x[i];
        end
    end
endgenerate
```

Limitations

Because of the nature of loops, there are some limitations on loops in synthesizable SystemVerilog.

- The number of iterations must be fixed.
- The iterations in the loop must be independent.