Final Blocks – proposal

Replace section 8.1
OLD:
Procedural statements are introduced by one of:

> **`initial`** // do this statement once
> **`always`**, **`always_comb`**, **`always_latch`**, **`always_ff`** // loop forever (see section 9 on processes)
> **`task`** // do these statements whenever the task is called
> **`function`** // do these statements whenever the function is called and return a value

SystemVerilog has the following types of control flow within a process

> — Selection, loops and jumps
> — Task and function calls
> — Sequential and parallel blocks
> — Timing control

Verilog procedural statements are in **`initial`** or **`always`** blocks, tasks or functions.

With:
NEW:

Procedural statements are introduced by one of the following:

> **`initial`** // do this statement once at the beginning of simulation
> **`final`** // do this statement once at the end of simulation
> **`always`**, **`always_comb`**, **`always_latch`**, **`always_ff`** // loop forever (see section 9 on processes)
> **`task`** // do these statements whenever the task is called
> **`function`** // do these statements whenever the function is called and return a value

SystemVerilog has the following types of control flow within a process

> — Selection, loops and jumps
> — Task and function calls
> — Sequential and parallel blocks
> — Timing control

Verilog procedural statements are in **`initial`** or **`always`** blocks, tasks or functions.
SystemVerilog adds a **`final`** block that executes at the end simulation

ADD as a new section before section 8.6

Final Blocks

The **final** block is like an **initial** block, defining a procedural block of statements, except that it occurs at the end of simulation time. A **final** block is typically used to display

statistical information, such as functional coverage reports, about the simulation. Like Verilog functions, **final** blocks execute in zero time, but without delay or event controls.

```
final_construct ::= final function_statement
```

Final blocks execute after one of the following occurs:

— The event queue is emptied
— Execution of $finish
— Termination of all program blocks. See section 15.9.1 $exit
— PLI execution of $tf_dofinish or similar routines.

An occurrence of one of the above while executing a **final** block will disable the current and all remaining final blocks.
Final blocks execute before any PLI callbacks indicating the end of simulation.

It shall be an error

```
final
  begin
     $display("Number of cycles executed %d",$time/period);
     $display("Final PC = %h",PC);
  end
```

ADD after the last paragraph of section 9.1 – Processes

Since **final** blocks cannot have any delay, they execute as a single thread .The ordering of execution of multiple **final** blocks is deterministic, but arbitrary.