Add the following section

11.6.1 Processes spawned by functions using fork..join_none

Functions in SystemVerilog shall execute with no delay. Thus, the process calling a function shall return immediately (in the same simulation time). However, functions may spawn arbitrary background processes by executing fork..join_none blocks provided that the process calling the function originates in an initial block. The fork..join_none blocks may contain any statements that are legal in tasks. Calling a function that executes a fork..join_none block shall be illegal in any context in which a side effect is disallowed or any context other than procedural code originating in an initial block. Examples of such illegal contexts are continuous assignments, nonblocking assignments, always_comb blocks, static variable declaration initializers, elaboration-time calls, and concurrent assertions. Implementations shall issue an error either at compile time or run time when they have determined the illegal condition.

Examples of a legal and illegal usage of fork..join_none in function are shown below.

```
class IntClass;
   int a;
endclass

IntClass address, stack;

function automatic bit watch_for_zero( IntClass p );
    fork
        forever @ p.a begin
            if( p.a == 0 ) $display( “Unexpected zero” );
        end
    join_none
    return( p.a == 0 );
endfunction

function bit start_check();
    return( watch_for_zero( address ) | watch_for_zero( stack ) );
endfunction

bit y = watch_for_zero( stack );                    // illegal

initial if( start_checks() ) $display ( “OK” );     // legal

initial fork
    if( start_checks() ) $display( “OK too” );      // legal
join_none
```