Proposals for the process/thread spawning options:

Option 1	**Option 2**	**Option 3**	**Option 4**	**Option 5**
fork...join all	fork...join_all	fork..join => all	fork..join -1	fork..join(expression)
fork...join none	fork...join_none	fork..join => none	fork..join 0	fork..join(expression)
fork...join any	fork...join_any	fork..join => any	fork..join 1	fork..join(expression)

Option 1: This is the original donation. Adds three new keywords: all, any, none.

Option 2: Adds three keywords less likely to collide: join_all, join_none, join_any

Option 3: Adds a lexical construct (in this example by overloading the implication operator) to enable the use of context sensitive keywords. In this case, all, none, and join are NOT keywords. The BNF would reflect this by adding explicit rules for this.

Option 4: Use constant numbers (digits?) to identify the type of join. An extension to this might be to include a constant expression and multiple numbers to define the number of threads that have to return before continuing.

Option 5: Use of a constant expression to indicate number of threads. Values can be -1 (for all), 0 (for none), and 1 (for any)

Note: In all five options, the first form (join all) is not strictly needed since this is the current (or default) behavior of Verilog. It exists only for the sake of documentation and orthogonality with the other join mechanisms.

Vote 1:

Prioritize 1 – 5
Vote with greater than 50% wins.
If no vote greater than then top two selected vote repeated.

Vote 2:

Deprecate process statement and associated references in 3.1 LRM
Yes or no vote (> 50% wins)