DPRSim Watchpoint Support

Enabling Watchpoints (in your .exp file):

```
ENABLEDEBUGGING=true
WP=<watchpoint expression> //WP0…WP9 also supported
```

Enabling Watchpoints (in your .hxx file):

Replace member variable declarations with `IVAR(type,name);` (for historian support)

Watchpoint Examples:

```
//count all pairs of catoms where a & b's random variable is 0
WP=(a,b); (a.RandomValue:value == 0) & (b.RandomValue:value ==
0); count

//draw arrows illustrating gradient function
WP=(a,b); (a.Gradient:value == (b.Gradient:value + 1)); arrows

//halt program if any catom is in state 3, and one of its
//neighbors was in state 2 last timestep
WP=(a,b); (a.CodeModule:state == 3) & (b.last.CodeModule:state
== 2); halt
```

Watchpoint Syntax:

- WP = module_list; expression; action
- module list: (a,b,c) declares the number and names of catoms in the trigger group
- expression: boolean expression that triggers action when satisfied
    - and (&) or (|) not (!) to join clauses
    - state variable access: catom_name.code_module:varName
        - append ".last"/".next" an arbitrary number of times to catom name to get state history
        - note that the variable must be monitored by Historian via the `IVAR(type,varname);` declaration
        - all state variables are stored/evaluated as floats
    - comparison operators: (value_a != value_b), supports all C comparisons
    - math operations: (value_a + value_b), supports all +,-,*,/
        - no explicit precedence, must use parens for grouping
    - explicit topology restrictions: n(a,b) requires that a and b are neighbors
        - by default, the watchpoint searches for ordered, nonlinear subensembles (those that can be found by filling variables in order, with potential backtracking)
- actions:
    - "halt" exit simulator
    - "count" count number of occurences of watchpoint this timestep
    - "arrows" draw arrows between all members of the subensemble
    - "color" set all members of the subensemble to the same (random) color

Comments/Suggestions Welcome!