<span class="hljs-meta">`<span class="hljs-meta-keyword">timescale</span> 1ns / 1ps</span>

<span class="hljs-comment">/\*\*

\* counter: a generic clearable up-counter

\*/</span>

<span class="hljs-keyword">module</span> counter

#(<span class="hljs-keyword">parameter</span> WIDTH=<span class="hljs-number">64</span>)

(

<span class="hljs-keyword">input</span> clk,

<span class="hljs-keyword">input</span> ce,

<span class="hljs-keyword">input</span> arst\_n,

<span class="hljs-keyword">output</span> <span class="hljs-keyword">reg</span> [WIDTH-<span class="hljs-number">1</span>:<span class="hljs-number">0</span>] q

);

<span class="hljs-comment">// some child</span>

clock\_buffer <span class="hljs-variable">#(WIDTH)</span> buffer\_inst (

<span class="hljs-variable">.clk</span>(clk),

<span class="hljs-variable">.ce</span>(ce),

<span class="hljs-variable">.reset</span>(arst\_n)

);

<span class="hljs-comment">// Simple gated up-counter with async clear</span>

<span class="hljs-keyword">always</span> @(<span class="hljs-keyword">posedge</span> clk <span class="hljs-keyword">or</span> <span class="hljs-keyword">negedge</span> arst\_n) <span class="hljs-keyword">begin</span>

<span class="hljs-keyword">if</span> (arst\_n == <span class="hljs-number">1'b0</span>) <span class="hljs-keyword">begin</span>

q &lt;= {WIDTH {<span class="hljs-number">1'b0</span>}};

<span class="hljs-keyword">end</span>

<span class="hljs-keyword">else</span> <span class="hljs-keyword">begin</span>

q &lt;= q;

<span class="hljs-keyword">if</span> (ce == <span class="hljs-number">1'b1</span>) <span class="hljs-keyword">begin</span>

q &lt;= q + <span class="hljs-number">1</span>;

<span class="hljs-keyword">end</span>

<span class="hljs-keyword">end</span>

<span class="hljs-keyword">end</span>

<span class="hljs-keyword">endmodule</span>