

Counter coding

need for a mix of blocking and non-blocking assignments

This is a made-up problem. It is a 5-bit truncated UP counter going from 0 to 20 and back to zero.

When the JUMP is asserted, the counter should jump to the mid-point M or a little beyond the midpoint M as detailed below. The midpoint M is the middle point between the current count and 20.

Example: Say, the current count is 12. Distance from 20 is 8. Half is 4. So the mid point M is at 16 ($12+4 = 16$).

But what if the difference is an ODD number? Then you need to jump by either the "floor" or the "ceiling" of the half of the difference depending on whether the difference is less than ten or more than ten respectively.

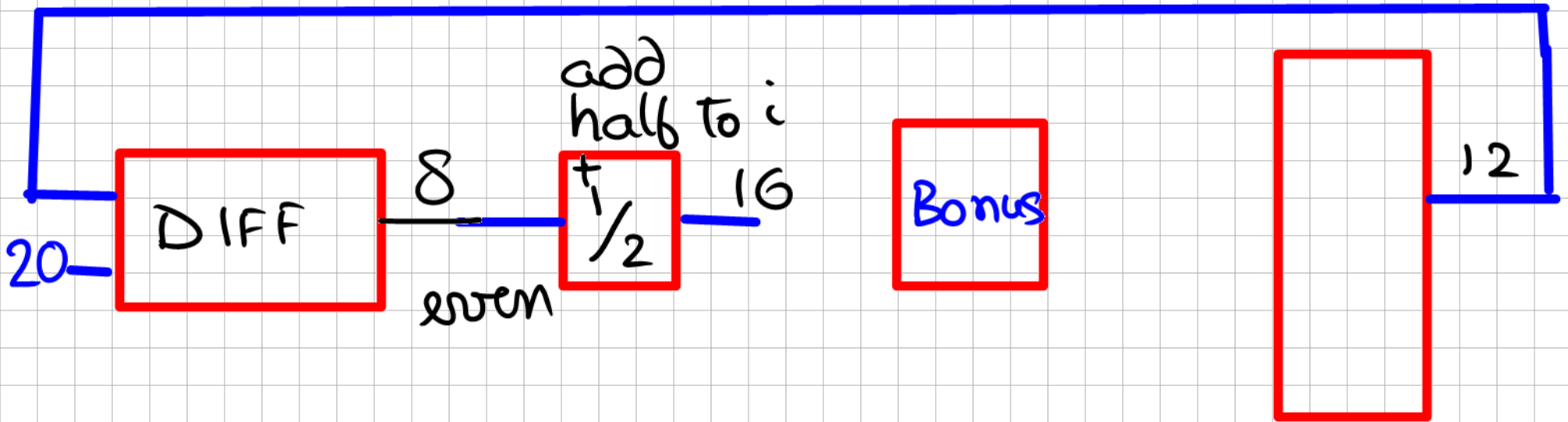
We said, "or a little beyond the mid point M".

After calculating the mid point M, a bonus to the jump by a constant of 4 is allowed if this bonus addition does not cause wrapping around (due to going beyond 20).

Examples: So, for the current count of 12, the jump takes you to 20, but, for ~~the~~ a current count of 14, the jump takes you only to 17.

Special Counter

→ 0 → 20



Part of the counter coding (in a clocked always block) illustrating proper mix of blocking and non-blocking assignments

```
if (jump)
  begin
    Q_next = Q; // notice blocking assignment
    diff = 20 - Q_next; // (20 - Q) also works
    if (diff[0] == 0) // if the difference is even
      begin
        Q_next = Q_next + { 1'b0 , diff[4:1] };
      end
    else // if odd
      begin
        if (diff < 10)
          begin
            Q_next = Q_next + { 1'b0 , diff[4:1] };
          end
        else
          begin
            Q_next = Q_next + { 1'b0 , diff[4:1] } + 1'b1;
          end
        end
      end
    if ( (20 - Q_next) >= 4)
      begin
        Q_next = Q_next + 4; // bonus 4
      end
    Q <= Q_next; // notice the non-blocking assignment
  end
end
```