4.3.1 Multiclocked Sequences and Properties

📖 **Rule**: *[1] Multiclocked sequences are built by concatenating singly clocked subsequences using the single delay concatenation operator ##1 or the zero-delay concatenation operator ##0. The single delay indicated by ##1is understood to be from the end point of the first sequence, which occurs at a tick of the first clock, to the nearest strictly subsequent tick of the second clock, where the second sequence begins. The zero delay indicated by ##0 is understood to be from the end point of the first sequence, which occurs at a tick of the first clock, to the nearest possibly overlapping tick of the second clock, where the second sequence b*

📖 **Rule**: Properties can use `|->, |=>` : Multiclocked properties can use the overlapping `|->` or non-overlapping implication `|=>` operators to create a multiclocked property from an antecedent sequence and a consequent property. The `|=>` or the `|->` operators synchronize the last expression of the antecedent clocked with the antecedent clock and the first elements of the consequent property being evaluated clocked with the consequent clock. The synchronization is the same as the one used with `##1` (for the `|=>`) and `##0` (for the `|->`) operators.
Consider the following two assertions (***/ch4/4.3/mclk2.sv***)

```
ap0: assert property(@(posedge clk1) $rose(a) |-> @(posedge clk2)  b);
ap1: assert property(@(posedge clk1) $rose(a) |=> @(posedge clk2)  b);
```

Condition	$rose(a)	@(**posedge** clk1) **$rose**(a) |-> @(**posedge** clk2) b)	@(**posedge** clk1) **$rose**(a) |=> @(**posedge** clk2) b)
At time t1 **posedge** clk1 is true event **posedge** clk2 is true event	True	**b is evaluated at t1**	**b is evaluated at** @(**posedge** clk2) after t1
At time t2 **posedge** clk1 is true event **posedge** clk2 is false	True	**b is evaluated at** @(**posedge** clk2) after t2	**b is evaluated at** @(**posedge** clk2) after t2

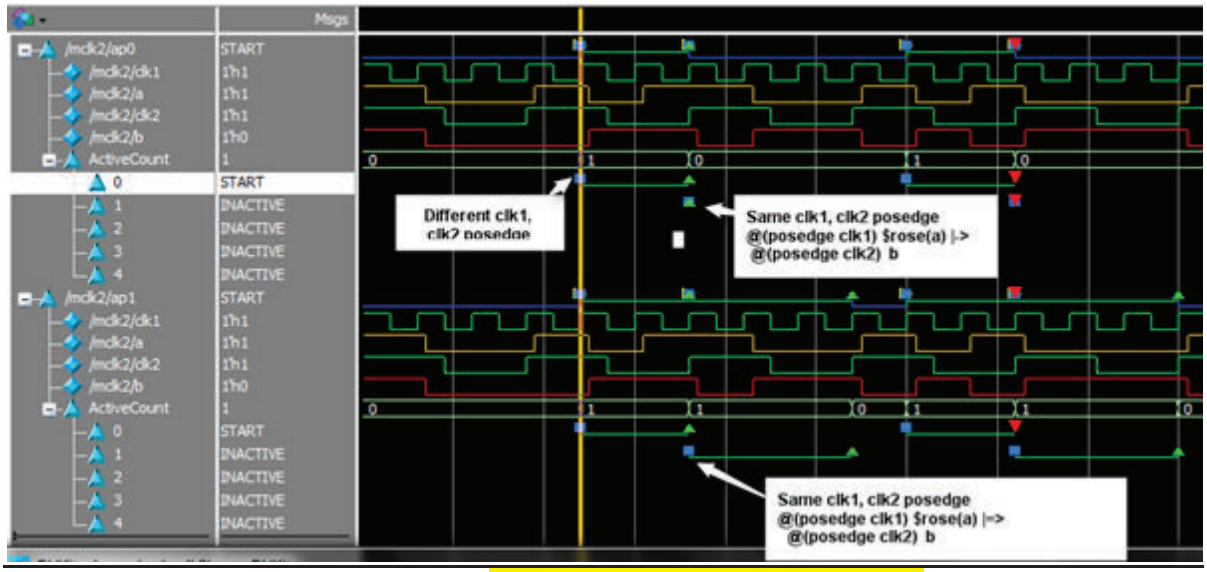

Figure 4.3.1-1 Multiclocked property ***/ch4/4.3/mclk2.png*** ***/ch4/4.3/mclk2.sv***

📖 **Rule**: `##1, ##0` concatenation: Multiclocked subsequences can only be combined with the `##1` or `##0` operators. The use of the `and`, `or`, `throughout`, `within`, or `intersect` operator would be illegal to combine multiclocked subsequences. Thus,

```
@(posedge clk1) a  ##2 @(posedge clk2) b // 💣 ##2 is illegal
    ##1 @(posedge clk3) (c##1 d)
   intersect   @(posedge clk4) e ##1 f; // 💣 intersect is illegal for multiclocked subsequences
```