4.3.2 Clocking Rules in Assertions

📖 **Rule**: Clocking in a property propagates to verification directives: A clocking event specified inside a PROPERTY declaration is propagated to the enclosing verification directive (e.g., **`assert, cover, assume`**). Thus,

```
property p_with_one_clock; @(posedge clk) a |=> b; endproperty : p_with_one_clock
ap_with_one_clock : assert property (p_with_one_clock);  // ✓
```

4.3.3 Clock Flow

📖 **Rule:** Flow through **`|->, |=>`** **:** When no explicit clock event is specified in an implication operator, the clock from the end point of the antecedent is understood to flow across the operator. Thus, the following to property expressions are identical:

```
@(posedge clk0) a0 |-> @(posedge clk0) a1 ##1 @(posedge clk2) a2;  // ✓
@(posedge clk0) a0 |-> a1 ##1 @(posedge clk2) a2;  // ✓
```

`clk0` implicitly flows across the implication operator. `a1` is clocked with

📖 **Rule:** The clock definition can change when crossing **`|->, |=>`** operators.

```
@(posedge clk0) a0 |-> @(posedge clk2) a1;  // ✓
@(posedge clk0) a0 |=> @(posedge clk2) a1;  // ✓
```

📖 **Rule:** Clock flow can change in conditional branch of **`if`:** A property expression of the form

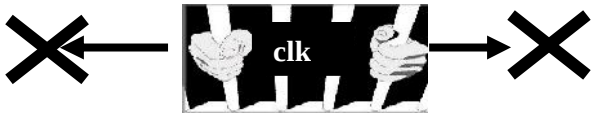
```
if ( expression_or_dist ) property_expr [ else property_expr ]
```

Clock flow can change in the conditional branch of **`if`** property operator (i.e., from the Boolean condition in the **`if`** statement to the beginnings of the **`if`** and **`else`** clause properties). Thus:

```
ap_ifOK: assert property (@ (posedge clk)
          if (a) @ (posedge clk1) b|=> c    // ✓ b, c tested at clk1
          else d ##1 e);                    // ✓ "a", "d", "e" tested at current (posedge clk)

ap_ifOK2: assert property ( @ (posedge clk)
          if (a) b|=> @ (posedge clk1) c    // ✓ "a", "b","d" tested @ (posedge clk)
          else   d ##1 @ (posedge clk2) e); // ✓ " c" @ (posedge clk1), e @ (posedge clk2),
```

📖 **Rule**: *[1] A clocking event in the declaration of a sequence or property does not flow out of an instance of that sequence or property*. However, the clock flows across elements of same sequence



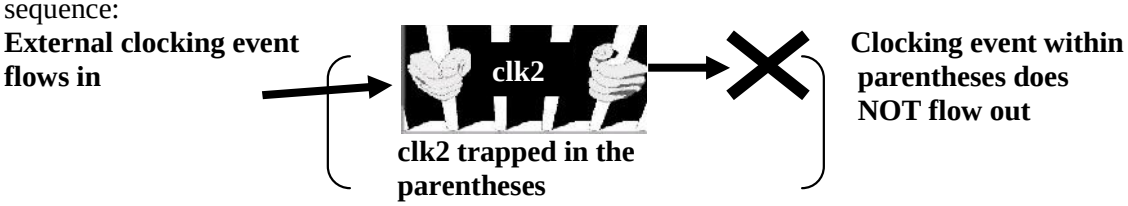
clk trapped in sequence

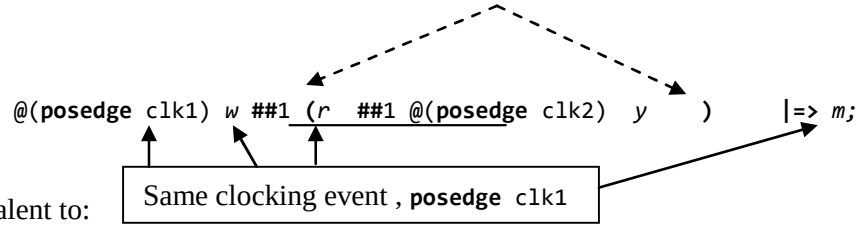
Clocking event within sequence or property does NOT flow out

For example, assuming no default clocking:

```
sequence q_ef;   @ (posedge clk) e ##[1:5] f;   endsequence : q_ef
ap_q_ef_a: assert property (q_ef ##1 a); // 💣 Illegal, clk does not flow into "a"
                                         // (i.e., no clocking event).
ap_ok: assert property(@ (posedge clk)
                       e ##[1:5] f ##1 a); // ✓ clock flows across elements of same sequence
ap_error: assert property (not q_ef);  // 💣 Illegal. Clocking event does not flow out of an
  // instance of the sequence q_ef. Thus, the not property operator has no leading clocking event.
ap_qWith_one_clock : assert property (q_ef);  // ✓ leading clocking event specified inside a
                // named sequence is propagated to the enclosing assertion statement
```

🕮 **Rule:** Clocking event is trapped in parenthesized sequence: *[1] The scope of a clocking event flows into parenthesized subexpressions and, if the subexpression is a sequence, also flows left-to-right across the parenthesized subexpression. However, the scope of a clocking event does not flow out of enclosing parentheses.* The standard also states that when sequence instances are flattened, the resulting expression that is returned is enclosed in parenthesis; therefore clocks do not flow out of sequence instances either. In the following example, the parentheses are within a sequence:



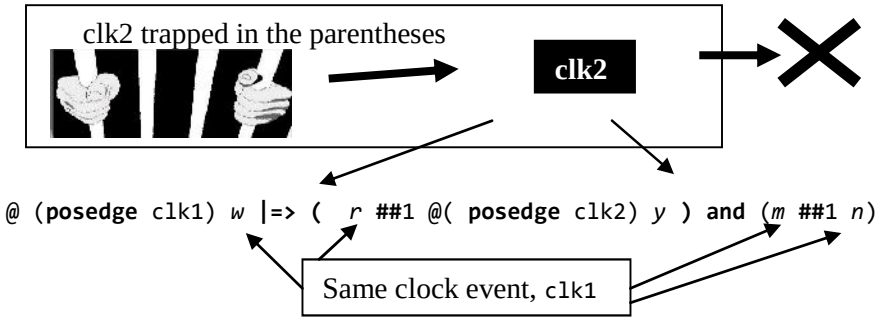


Is equivalent to:

```
@(posedge clk1) w ##1
                (@(posedge clk1) r ##1
                 @(posedge clk2)  y    |=>
                                          @(posedge clk1) m;
```

w, r, m are clocked at **posedge** clk1 and y is clocked at **posedge** clk2. Clock **posedge** clk1 flows across **##**1, over the parenthesized subsequence (*r* **##1 @(** **posedge** clk2) *y*), and across the non-overlapping implication **|=>** operator. Clock **posedge** clk1 also flows into the parenthesized subsequence, but it does not flow through **@(posedge** clk2). Clock **posedge** clk2 does not flow out of its enclosing parentheses; thus it does not flow into m.

Consider the following example where the parentheses are within a property:



w, r, m, n are clocked at **posedge** clk1, and y is clocked at **posedge** clk2. Clock clk1 flows across the non-overlapping operator **|=>**, distributes to both operands of the property **and** operator, and flows into each of the parenthesized subexpressions. Within (r **##1 @(posedge** clk2) y), clk1 flows across **##1** but does not flow through **@(posedge** clk2). Clock **posedge** clk2 does not flow out of its enclosing parentheses. Within (m **##1** n), **posedge** clk1 flows across the **##1** delay.