A Dyadic Transformation Based Methodology To Achieve Coverage Driven Verification Goal

Swapnajit Mitra

Feb 2017

Meeting Coverage Goal is EVERYTHING

- Goals for Constrained Random Verification Methodology are coverage driven.
- Unless coverage goals are met, verification is incomplete.
  - Both functional cover group coverage goals and
  - Cover property coverage goals must be met.
- Meeting cover property coverage goal is challenging.
  - Majority of the cover properties are hit without much effort in a constrained random regression.
  - However, hitting the rest is always a challenge.
- If a cover property is not hit, the usual solution is to iterate over the following steps:
  - Designer analyzes/guesses why a cover property is not hit
  - Verifier tries to reproduce it
- This paper is proposing a methodology to eliminate the need for guesses from the designer and automate the process of analysis.

What Would It Take to Hit This Cover Property?

Two Simple Examples

```
sequence seq;
   s;
endsequence

cp: cover property @(posedge clk) seq;
```

```
s = a & b;
```

- Conjunctive Normal Form.

```
s = a | b;
```

- Disjunctive Normal Form.

- As long as we can convert a sequence expression to CNF or DNF, we know precisely the condition to get the cover property get hit.

Things Quickly Get Messy for Temporal Expressions

- How do we analyze easily why this cover property is not hit?

```
sequence seq;
   !req ##1 req[*1:$] ##1 !req |-> !ack[*1:$] ##1 ack[*1:$]
##1 !ack ;
endsequence

cp: cover property @(posedge clk) seq;
```

- It would be very convenient, if we can convert the above sequence expression to a CNF.
- This paper proposes a methodology to do this.

The Math

Definitions

- A *Nominally Expressed Property* (NEP) in this paper is defined as a concurrent property $P(e(\{v\}), T)$ whose property expression is a generalized heterogeneous combination of conjunctive normal and disjunctive normal forms involving temporal or immediate events *e* on the set of variables *{v}* with *T* as the time horizon for the life cycle of a single spawn of the property.
- *Temporal Transformation Function* (TTF) $\Phi^{m,n}(c):c \leftarrow (c, m, n)$ is a function that returns true if any of the set of values a literal *c* had from *m* to *n* clock cycles earlier was true.
  - $\Phi^{n,n}(c):c \leftarrow (c, n, n)$ represents if a literal *c* was true exactly *n* clock cycles earlier.
  - $\Phi^{n}(c)$ is used to represent $\Phi n,n(c)$.
- $\mathfrak{T}$ as a dyadic transformation that maps an NEP compliant expression to a CNF involving a new set of events *e'* involving a transformed set of variables *v'* in the same time horizon *T*.

$$\mathfrak{T}(e(\{v\}), T) \rightarrow \wedge e'(\{v'\}, T)$$

 |

BROADCOM

The Math (Contd.)

- The two parts that the dyadic transformation $\mathfrak{T}$ consists of are
  - Part A) the transformation on the variables that do not need any temporal transformation and
  - Part B) on the ones that do.
- Since any normalized boolean expression can be represented in a CNF without any change in the set of variables, *v'* can be expressed as a union of a proper subset of *v* and a converted form of rest of the variables of *v* through function *Φ* for the corresponding values of *n* (or *m, n*) associated with each variable.

$$v' = s1 \cup s2 \mid s1 \subset \{v\},\ s2 = \{\ \Phi^{m,n}(\{v\}-s1)\}$$

- Combining:

$$\mathfrak{T}\ (e(\{v\}),\ T) \rightarrow \wedge e'(\{s1 \cup s2 \mid s1 \subset \{v\},\ s2 = \{\ \Phi^{m,n}(\{v\}-s1)\}\},\ T)$$

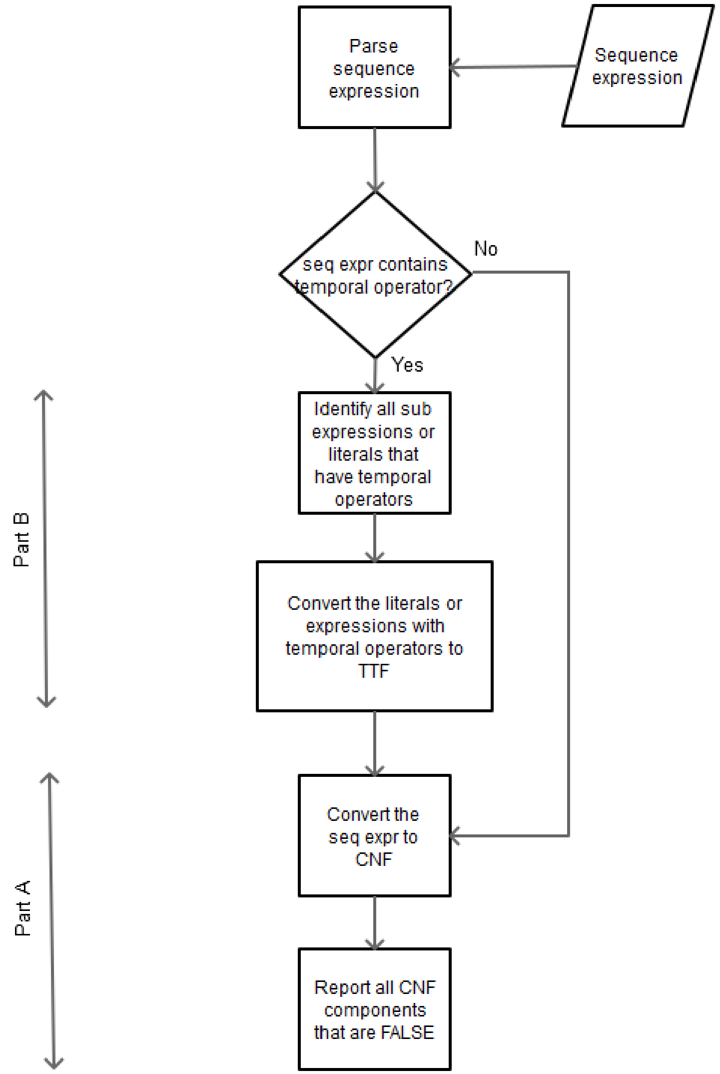
BROADCOM

Flow Chart

Also A Sample Implementation for TTF

```
function bit ttf (bit c, int m, int n=-1);
bit equality;
int i;

if (n == -1) begin // if only m has been passed
   ttf = $past(c, m); // note that some simulators
need constant m to be passed
end else begin // if both m and n have been passed
   equality = 1'b0;
   for (i=m; i<=n; i++) begin
      equality = equality | $past(c, i);
      if (equality == 1'b1) begin
         i = n+1;
      end
   end
   ttf = equality;
end
endfunction
```



 |

BROADCOM

Examples:

- Case 1: *s: (a∧cvb∧c)*
  - Immediate sequence expression → there is no Part A, and the transformation has Part B only.
  - Corresponding CNF can be derived using only non-temporal transformation for *s* which is *c(avb)*.
  - To hit *cp,* both *c* and *(avb)* need to be hit. Since *(avb)* can be hit if either *a* or *b* is hit, *c* and one of *a* or *b* need to be hit.

```
sequence seq;
   (irdy & trdy) | (irdy & stop);
endsequence
cp: cover property @(posedge clk) seq;
```

→

```
sequence seq;
   irdy & (trdy | stop);
endsequence
cp: cover property @(posedge clk) seq;
```

- Case 2: *s: avb∧c*
  - no temporal transformation and corresponding CNF for *s* is *(avb)∧(avc)*.
  - To hit *cp,* both *(avc)* and *(avb)* need to be hit, i.e., a or both *b* and *c* need to be hit.

```
sequence seq;
   (csr1[4] | csr2[7]) & csr7[0];
endsequence
cp: cover property @(posedge clk) seq;
```

→

```
sequence seq;
   (csr1[4]&csr7[0])|(csr2[7]& csr7[0]);
endsequence
cp: cover property @(posedge clk) seq;
```


Examples (Contd.)

- Case 3: *s: a ##`N b*
  - In this case, *s* has a temporal transformation on *a*. Here, *s* can be expressed as a CNF as *ttf(a,`N)∧b*.
  - To hit *cp* in this case, *ttf(a, `N)* and *b* need to be hit.

```
sequence seq;
   a ##5 b;
endsequence
cp: cover property @(posedge clk) seq;
```

→

```
sequence seq;
   ttf(a, 5) & b;
endsequence
cp: cover property @(posedge clk) seq;
```

```
sequence seq;
   (a | b) ## 3 c;
endsequence
cp: cover property @(posedge clk) seq;
```

→

```
sequence seq;
   ttf(a, 3) & c | ttf(b, 3) & c;
endsequence
cp: cover property @(posedge clk) seq;
```


Examples (Contd.):

- Case 4: *s: a ## [`M:`N]b*
  - *s* has a temporal transformation on *a* over a spread of *`M* to *`N* clock cycles. Here, *s* can be expressed as a CNF as *ttf(a,`M,`N) ∧b*.
  - To hit *cp* in this case, *ttf(a, `M,`N)* and *b* must be hit.
  - This is similar to Case 3, but has a temporal spread.

```
sequence seq;
   a ##[5:9] b;
endsequence
cp: cover property @(posedge clk) seq;
```

➡

```
sequence seq;
   ttf(a, 5, 9) & b;
endsequence
cp: cover property @(posedge clk) seq;
```


Examples (Contd.)

- Case 5: *s: a |-> ##[`M:`N]b*
  - The operator |-> is not a temporal transformation by itself. Thus, s in this case will be same as *ttf(a,`M,`N) |-> b.*
  - It is easy to see though that the non-vacuous success case will be *ttf(a,`M,`N)∧b*.
  - Thus, to hit *cp*, both *ttf(a,`M,`N)* and *b* must be hit.

```
sequence seq;
   a |-> ##[5:9] b;
endsequence
cp: cover property @(posedge clk) seq;
```

→

```
sequence seq;
   ttf(a, 5, 9) & b;
endsequence
cp: cover property @(posedge clk) seq;
```

```
sequence seq;
   a |=> ##[5:9] b;
endsequence
cp: cover property @(posedge clk) seq;
```

→

```
sequence seq;
   ttf(a, 6, 9) & b;
endsequence
cp: cover property @(posedge clk) seq;
```


A Complex Example and Summary

- A complex example:

```
sequence seq;
   a ##1 !b ##[2:4] b ##1 !b ##1 c;
endsequence
cp: cover property @(posedge clk) seq;
```

```
sequence seq;
   ttf(ttf(ttf(ttf(a, 1) & !b, 2, 4) & b,
1) & !b, 1) & c;
endsequence
cp: cover property @(posedge clk) seq;
```

- A methodology based on a dyadic transformation is proposed
  - To aid, if not eliminate, human need for analysis of an un-hit cover property expression.
  - Both temporal and immediate components are considered.
  - A mathematical structure of the transformation is shown with associated examples.

BROADCOM

BROADCOM®

connecting everything®