HW5 **Due: Dec 12 2015**

Use the intermediate file format discussed in class to complete this assignment. The intermediate file consists of lines where each line represents a gate in the circuit. The format of each line in the file in shown below:

GateType	Output	GateLevel	#faninN	fin1	fin2	...	finN	#fanoutM	fout1	fout2	...	foutM	GateName

In this format, each gate has a name except the added buffers. The test bench circuits do not contain buffers.

A. Use 3-valued logic { 0, 1, X } to create a two inputs lookup tables for the following gates AND, OR, XOR, and NOT. Gates with more than two inputs can be evaluated by repeated evaluation using the two inputs table lookup.
B. Add to every gate structure a pointer 'sched' and set that pointer initially to zero. This pointer should be used to schedule the corresponding gate. Create a dummy gate structure and keep a pointer to it in variable 'dummy gate'.
C. Create an array 'levels' of pointers to a gate structure of size 'max level'. 'max_level' is an integer holding the value of max level in your design. Initially, levels[i] is set to dummy gate for all i's.
D. Use 3-valued logic {0, 1, X} to create two inputs lookup tables for the following gates AND, OR, XOR, and NOT. Gates with more than two inputs can be evaluated by more than one table lookup.
E. To schedule events due to a change of the state of gate i:
    a. For each gate f in the fanout list of gate i, if the field 'sched' of gate f is zero, then insert f at the head of the list at the corresponding level. Otherwise, no action is needed (schedule_fanout(gaten) in **Figure P1**).
F. Implement the algorithm shown in Figure P1. In this algorithm, Flip-Flop do not need to be scheduled:

```
while() {
        print logic values at PI, PO, and States
        read inputs and schedule fanouts of changed inputs.
        load next state and schedule fanout.
        i = 0;
        while( i < max level) {
           gaten = levels[i];
           while( gaten != dummy_gate ) {
                   newstate = evaluate( gate);
                   if( new_state != gate.state ) {
                           gate[gaten].state = new_state ;
                           schedule_fanout( gaten );
                   }
                   tempn = gaten ;
                   gaten = gate[gaten].sched ;
                   gate[gaten].sched = 0 ;
                   gaten = tempn;
           }
           levels[i] = gaten ;
           i = i + 1 ;
  }
}
```

Figure P1: Simulation Flow

HW5 **Due: Dec 12 2015**

Implement the 'evaluate' routine using the two techniques discussed in class:

a. Input scanning:

Controlling value ‘c’ and inversion ‘i’

	c	i
AND	0	0
OR	1	0
NAND	0	1
NOR	1	1

```
evaluate(Gn){
        Uvalue = FALSE
        for(i=0; i<gate[Gn].nfanin;i=i+1){
                V = gate [ gate[Gn].fanin[i] ].state;
                if( V = c ) return( c ⊗ i)
                if( V = X) Uvalue = TRUE;
        }
        if( Uvalue ) return X; else return c̄ ⊗ i;
}
```

b. Table lookup:

```
evaluate(Gn){
    state_fanin0 = gate[ gate[Gn].fanin[0] ].state ;
    gate_type = gate[Gn].gtype;
    if gate_type == INV then return Inv_table[ state_fanin0 ] ;
    if gate_type == BUF then return state_fanin0 ;
    state_fanin1 = gate[ gate[Gn].fanin[1] ].state ;
    v = Table[gate_type][ state_fanin0 ][ state_fanin1 ]
    nfanin = gate[Gn].nfanin;
    for( i = 2; i < nfanin ; i++) {
        state_fanin0 = gate[ gate[Gn].fanin[i] ].state ;
        v  = Table[gate[Gn].gate_type][ state_fanin0 ][ v ] ;
    }
    If( gate[Gn].i ) return (Inv_table[v]; else return ( v ) ;
}
```

c. Record the CPU time your program requires to run both examples circuit posted on the website using the input scanning and table-lookup. Compare the two techniques.

HW5 **Due: Dec 12 2015**

Here is an input/output for s27

```
input file for s27: G0, G1, G2, G3

        0000
        0010
        0100
        1000
        1111

output file: 4 represents undefined

        INPUT   :0000 // G0, G1, G2, G3
        STATE   :444  // G5, G6, G7
        OUTPUT  :4    // G17

        INPUT   :0010
        STATE   :044
        OUTPUT  :4

        INPUT   :0100
        STATE   :040
        OUTPUT  :4

        INPUT   :1000
        STATE   :041
        OUTPUT  :1

        INPUT   :1111
        STATE   :101
        OUTPUT  :1
```