ECE 2195 - Homework 2

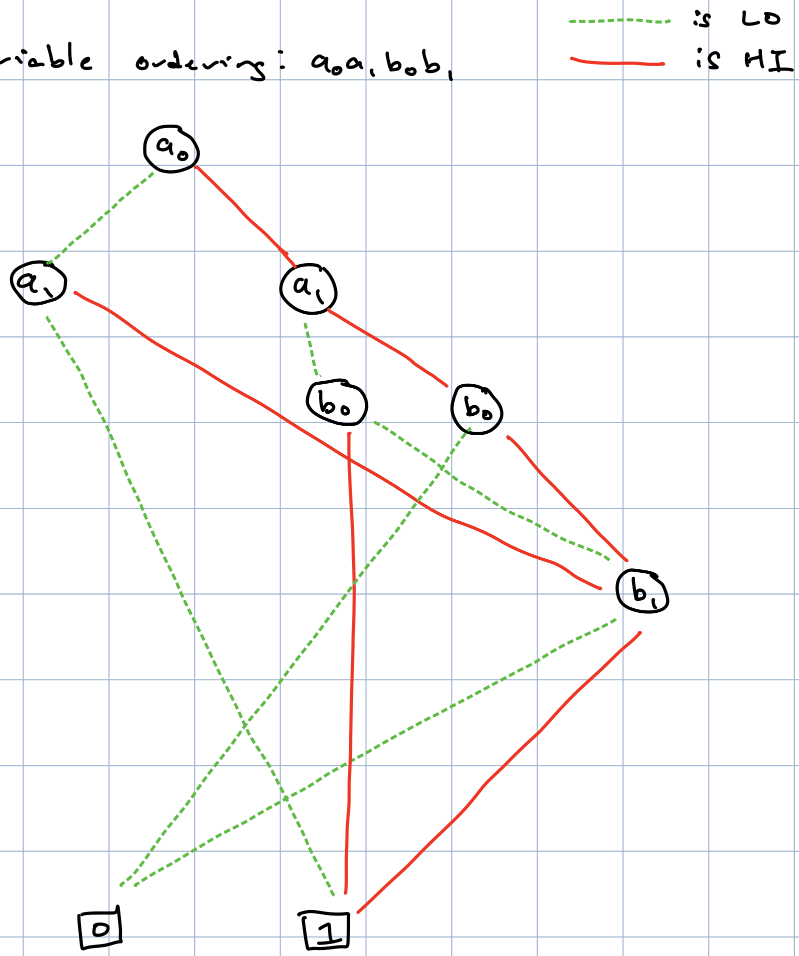
Avery Peiffer

1. BDDs: Static Variable Ordering

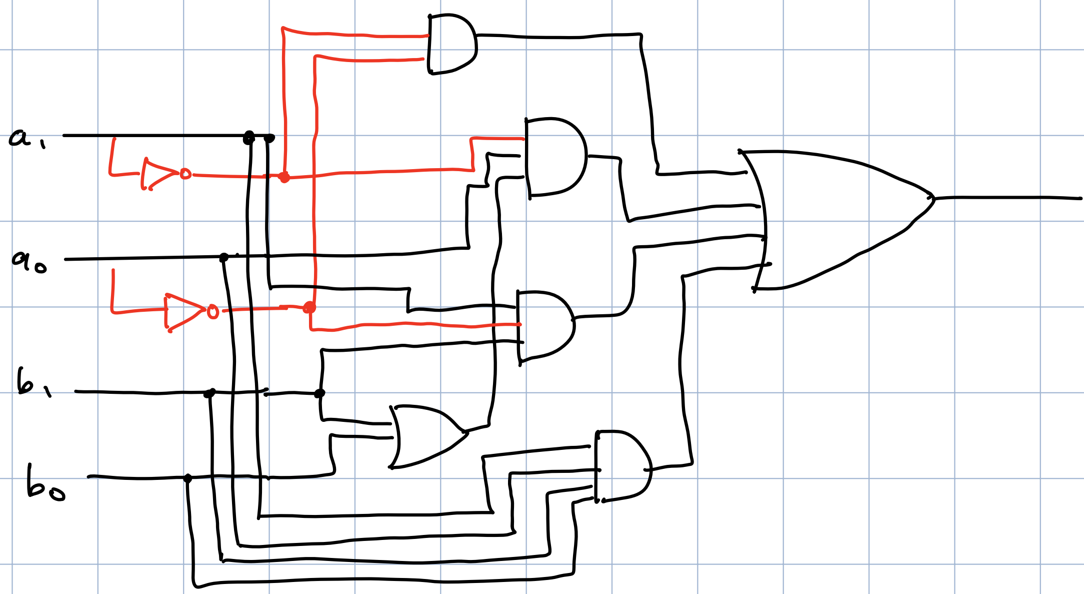
$a_1 a_0 <= b_1 b_0 \longrightarrow z = 1$

a_1	a_0	b_1	b_0	z
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

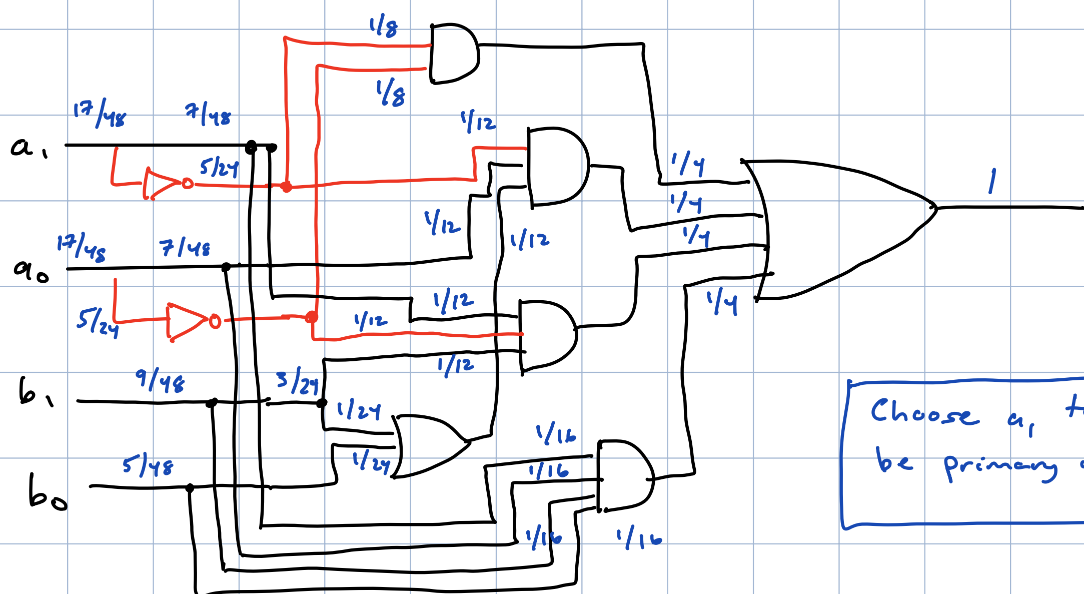
(i) Bad variable ordering: $a_0 a_1 b_0 b_1$



(ii) $a_1'a_0' + a_1'a_0[b_0+b_1] + a_1a_0'b_1 + a_1a_0b_1b_0$

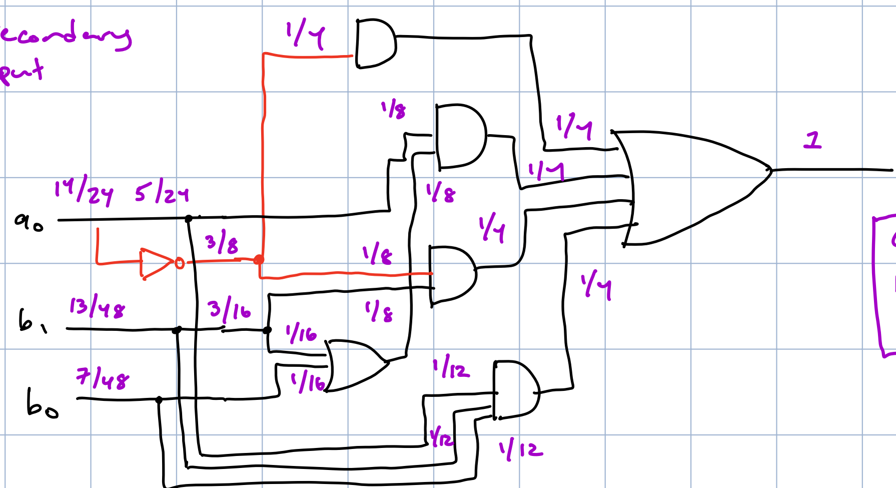


(iii) Assign weight 1 to primary output and propagate back



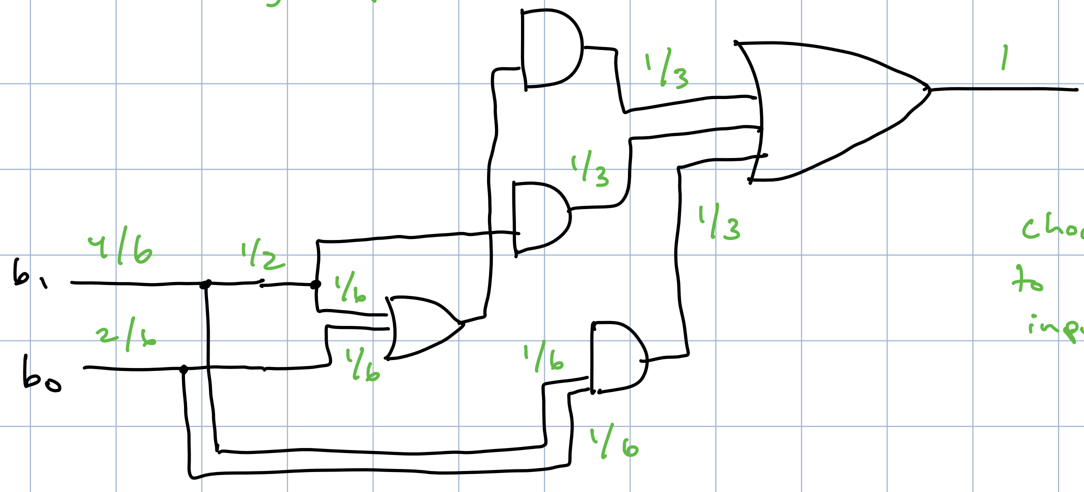
Choose a_1 to be primary input

Find secondary input



Choose a_0 to be secondary input

Choose tertiary input



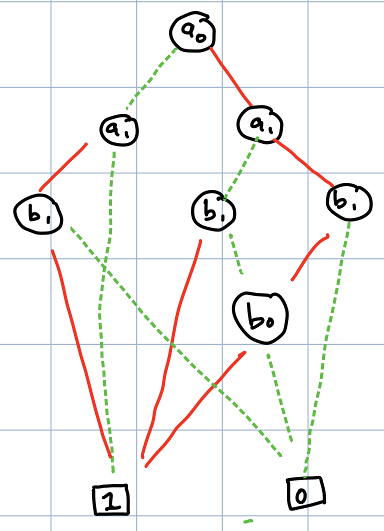
Choose b_1 to be tertiary input

New ordering: $a_1 a_0 b_1 b_0$

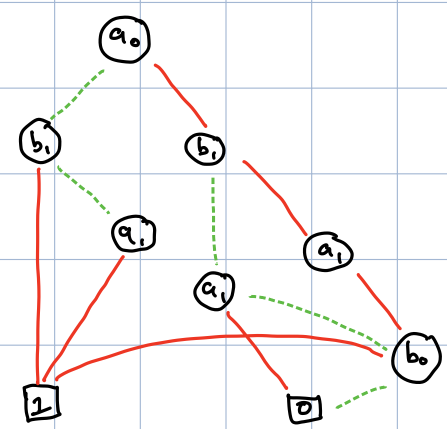
2. BDDs: Dynamic Variable Ordering

Sift b_1 up to the top

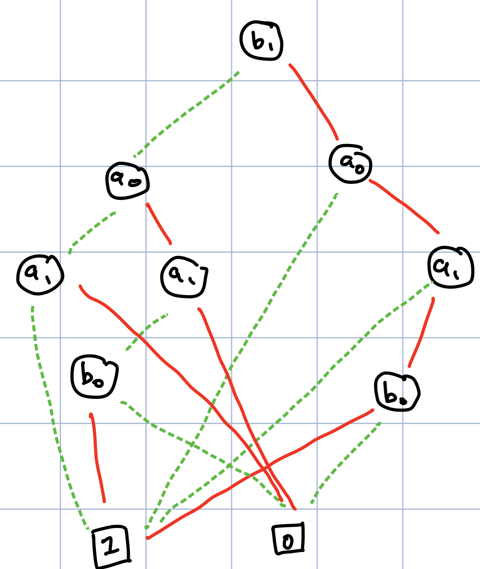
1. $a_0 a_1 b_1 b_0$



2. $a_0 b_1 a_1 b_0$



3. $b_1 a_0 a_1 b_0$



It appears that location 2 is best for b_1, as not many connections are necessary, compared to the other methods.

3. Derived Operators for BDDs

```
bool depends (bdd f, var x) {
    return (! iszero (EXOR (cofactor(f, x, (bool) 0), cofactor (f, x, (bool) 1)));
}

bdd univquant (bdd f, var x) {
    return ( AND (cofactor (f, x, (bool) 0), cofactor (f, x, (bool) 1 ));
}

bool opposite (bdd f, bdd g) {
    return ( OR (NOT(f), g));
}

bdd exchange (bdd f, var a, var b) {
    bdd A = var2func(a);
    bdd B = var2func (b);
    bdd F_a = cofactor (f, a, 1);
    bdd F_ap = cofactor (f, a, 0);
    bdd F_a_b = cofactor (F_a, b, 1);
    bdd F_a_bp = cofactor (F_a, b, 0);
    bdd F_ap_b = cofactor (F_ap, b, 1);
    bdd F_ap_bp = cofactor (F_ap, b, 0);
    return ITE(A, ITE (B, F_a_b, F_ap_b), ITE(B, F_a_bp, F_ap_bp));
}

bdd compose (bdd f, bdd g, var x) {
    bdd F_x = cofactor (f, x, 1);
    bdd F_xp = cofactor (f, x, 0);
    return ITE (g, F_x, F_xp);
}
```

4. Reverse FSM Reachability Analysis

$$R_{-(k+1)}(p,q) = R_{-k}(p,q) + (\exists x) R_{-k}[p^+(p,q,x), q^+(p,q,x)]$$

(i) This formula makes sense because it is essentially the "forward" formula applied with a change in perspective. If $R_{-k}(p,q)$ is reachable, then the existential quantifier represents all of the states that could reach that state. So, in that sense, we are able to "go back in time."

(ii) $R_0(P,q) = \{\text{state } B\} = p'q$

$$R_{-1}(p,q) = R_0(p,q) + \exists x R_0 \underbrace{[p^+(p,q,x), q^+(p,q,x)]}_{\delta(p,q,x,p^+,q^+)}$$

$$\delta(p,q,x,p^+,q^+) = [p^+ \oplus (pq' + p'x)]' \cdot [q^+ \oplus (p'q + p'x')]'$$

$$= p'q' + p'q$$

$$R_{-1}(p,q) = p'q + p'q' + p'q \longrightarrow p'q + p'q' \longrightarrow \text{States } A \text{ and } B$$

Yes, this result makes sense. In order to reach State B in one clock cycle, the machine must already be at either State A or State B. This aligns with what we see in the FSM diagram.

5. YBDD (Writeup)

(i)

Bits in adder	Size (right ordering)		Size (wrong ordering)	
4	21	13	55	43
8	45	25	1007	759
16	93	49	Did not finish	
64	381	193	Did not finish	

Table values are reported as the size of the final sum bit and final carry bit

(ii) Ling Adder output

I first declared p and g for all i. I then found $P_{63} = P_6 P_5 P_4 P_3$, $H_1 = g_7 + g_6 + P_6 g_5 + P_6 P_5 g_4$, and $H_0 = g_3 + g_2 + P_2 g_1 + P_2 P_1 g_0$. I then used these results to calculate $h_8 = H_1 + P_{63} H_0$, which was used to calculate $c_7 = P_7 h_8$. This c7 is then used to calculate s7, which is compared against s7_true using the XOR operator.