ECE/CS 314 Spring 2004 **Section 5**

Combinational Logic

From lecture, we have seen the design of a 1-bit full adder. Now, let's try to add a carry-in bit to the input:

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The equations can be derived immediately as:

Sum = $\underline{a}\ \underline{b}\ c + \underline{a}\ b\ \underline{c} + a\ \underline{b}\ \underline{c} + a\ b\ c$
Cout = $\underline{a}\ b\ c + a\ \underline{b}\ c + a\ b\ \underline{c} + a\ b\ c$

To simplify the equations, Karnot Maps should be used:

Sum:

	$\underline{b}\ \underline{c}$	$\underline{b}\ c$	$b\ c$	$b\ \underline{c}$
$\underline{a}$	0	1	0	1
a	1	0	1	0

Cout:

	$\underline{b}\ \underline{c}$	$\underline{b}\ c$	$b\ c$	$b\ \underline{c}$
$\underline{a}$	0	0	1	0
a	0	1	1	1

While Sum cannot be reduced further, Cout can be reduced to:

Cout = $a\ c + a\ b + b\ c$

Espresso

Espresso is a logic minimization tool; it is installed on the BSD cluster. To demonstrate it, use the truth table from above. Place the table in a text file as follows (Let's call it adder.raw):

```
.i 3
.o 2
000 00
001 10
010 10
011 01
100 10
101 01
110 01
111 11
.e
```

Don't-cares are indicated as '-' (rather than an 'x' from ECE230).

You can run espresso on this file by executing:

```
espresso –s adder.raw > adder.red
```

The reduced truth table will be saved in adder.red:

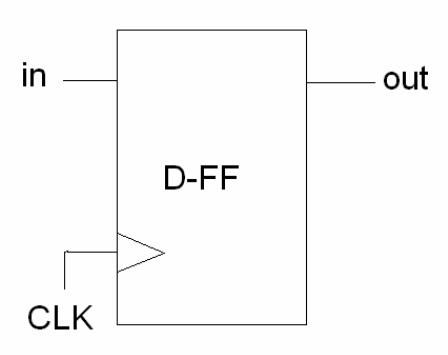
```
.i 3
.o 2
.p 7
100 10
010 10
001 10
111 10
-11 01
1-1 01
11- 01
.e
```

To extract the equations, look at each column on the right (output) side. A '1' indicates that the min-term on the left is included in the reduced equation. For example, the Sum column (1st column on the right side) has '1's at the last row (11-), which means that the min-terms is a·b.

(Note: If an equation has more than one min-term, the result is just the "sum" of them)

Sequential Logic

State holding element: D Flip-flop



On a positive edge of the clock signal, the D Flip-flop passes the input value to the output.

Now, let's implement a 2-bit counter:

Reset	S1	S0	S1'	S0'
0	0	0	0	1
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

The equations of S0' and S1' can be derived as:

S1' = $\underline{\text{Reset}}\ \underline{\text{S1}}$ S0 + $\underline{\text{Reset}}$ S1 $\underline{\text{S0}}$
S0' = $\underline{\text{Reset}}\ \underline{\text{S1}}\ \underline{\text{S0}}$ + $\underline{\text{Reset}}$ S1 $\underline{\text{S0}}$

And here’s a gate diagram of the circuit: