Embedded Systems
Professor Thomas Bräunl
Associate Lecturer Kieran Quirke-Brown

Tutorial 1 – Number Representation

1. Convert the following numbers:

GIVEN		CONVERT TO	
Decimal	77	Binary, 8-bit signed	0100 1101
Decimal	-100	Binary, 8-bit signed	1001 1100
Decimal	99	Hex, 8-bit signed	63
Decimal	-23	Hex, 8-bit signed	E9
Binary, signed	1101 1100	Hex, 8-bit signed	DC
Binary, signed	1101 1100	Decimal	-36
Binary fixed pt.	11.0101	Decimal	3.3125
Decimal	0.85	Binary fixed pt. (4 deci.)	0.1101
Decimal FP	-16.25	IEEE FP	1 1000 0011 0000 0100
IEEE FP	0 0111 1111 1110000 00000000 00000000	Decimal FP	+; exp=0; 1.111 = 1.875
IEEE FP	1 1000 0010 0110000 00000000 00000000	Decimal FP	-; exp=3; 1.011 = -11

2. Negate the following 8-bit numbers using 2's complement:

1011 1100 0100 0011 → 0100 0100

1000 0000 0111 1111 → 1000 0000 (overflow, can't negate -128)

3. Convert the following FP numbers to IEEE FP format (only up to 4 binary FP digits):

Given	Sign, Exponent	FP Bit sequence
0	sign = 0; exp = 0	0 0000 0000 0000 0000 0000…
–1	sign = 1; exp = 127	0 0111 1111 0000 0000 0000…
+1.1	sign = 0; exp = 127	0 0111 1111 0001 0000 0000…
–65.75	sign = 1; exp = 133	1 1000 0101 0000 0111 0000…

4. Using kmaps find the equations for the following outputs and state transitions, draw the relevant circuit.

D2	D1	D0	A	B
0	0	0	0	1
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	0	0
1	1	0	1	0

Solution:

First we look at the transition states, we know we need 7 states to represent all the changes of A and B. We determine the number of flip flops we need by working out the smallest power of two that is greater than 7. In this case 2 to the power of 3 gives us 8 which is greater than 7 and therefore the number of flipflops we need to represent the entire system is 3. Starting from 000 we count to 6 in binary (0-6) for our seven states. Each current state represented by Q2-0 needs to transition to the next line which we will represent as D2-0 (next state). Now we can define a kmap for each of the D states to build our combinatorial circuit. We also include any other possible states but mark them with an X as don't cares as we shouldn't be getting into those states.

Q2	Q1	Q0	D2	D1	D0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	1	X	X	X

D2 Kmap

	Q1Q0	00	01	11	10
Q2					
0		0	0	1	0
1		1	1	X	0

We can then write an expression for when D2 is 1 by grouping the 1's (and don't cares if needed) into the largest groupings of a power of 2 in either a straight line or a square.

D2 = Q2Q1' + Q1Q0

Note the apostrophe indicates active low (when Q1 is 0 then Q1' is 1).

D1 Kmap

	Q1Q0	00	01	11	10
Q2					
0		0	1	0	1
1		0	1	X	0

D1 = Q1’Q0 + Q2’Q1Q0’

D0 Kmap

	Q1Q0	00	01	11	10
Q2					
0		1	0	0	1
1		1	0	X	0

Note in this case we can wrap the top right most 1 with the top left 1. Since we are using the top left 1 twice (once for yellow and once for green) we indicated it with blue.

D0 = Q1’Q0’ + Q2’Q0’

We now have the foundations of our state circuit from here we just need to work out the circuits for A and B. Again we can just create a Kmap from using the state table and the original table to the question.

A Kmap

	Q1Q0	00	01	11	10
Q2					
0		0	1	0	0
1		1	0	X	1

A = Q2Q0’ + Q2’Q1’Q0

B Kmap

	Q1Q0	00	01	11	10
Q2					
0		1	1	1	0
1		1	0	X	0

B = Q1’Q0’ + Q2’Q0

Now we have all the equations we can implement it in Retro as below. The LEDs for each output are to confirm their current state.