In-semester Test 1

- The test will be 1 hour in duration
- No Internet
- The test will consist of **three** questions
- You will be provided with stub .hdl files for each question

Before the test

- You will be given access to the required files.
- Extract the .zip
- You will have a directory named 'elXXxxx' that contains all the required files.
- Re-name the directory using your Leeds University username e.g. 'el17xyz'

During the test

- The only files you need to modify are **q1.hdl**, **q2.hdl** and **q3.hdl**
- Do not change the names of the input or output pins. Only write code in the PARTS: area of the .hdl files
- Once you have implemented the circuit, use the Hardware Simulator to test your implementation. Ensure you use a range of inputs to thoroughly test your implementation for correct behaviour.

After the test

- Ensure that you have saved all your HDL files
- Check that you have re-named the directory using your Leeds University username
- Create a .zip of the directory and submit in the 'Submit My Work' section of the XJEL2665 module page in Minerva.
- Check the submitted file is the correct one.

Once you have left the examination venue there is no opportunity to re-submit.

Question 1

Design a circuit that implements the following truth table in *q1.hdl*

a	b	c	out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

8 marks total

Question 2

In *q2.hdl* implement an 8-bit comparator. The circuit has a two 8-bit inputs. The output should be a 1 if the inputs are the same. The output should be 0 if the inputs are different

For example, if the inputs were

1	1	1	0	0	1	0	0
1	1	1	0	0	1	0	0

then the output would be 1 as the inputs are the same.

If the inputs were

1	0	0	0	1	1	1	0
1	1	1	0	0	1	0	0

then the output would be 0 as the inputs are different.

8 marks total

Question 3

In *q3.hdl* implement a 2's complement generator. The circuit has a one 8-bit input and one 8-bit output. The output should be the 2's complement of the input.

For example, if the input was

0	0	0	0	0	0	1	0

i.e. decimal 2, the output would be

1	1	1	1	1	1	1	0

i.e. -2 in 2's complement.

8 marks total

API

Chip Name:	`Nand`
Inputs:	`a, b`
Outputs:	`out`
Function:	`If a=b=1 then out = 0 else out = 1`
Comment:	`This gate is considered primitive and thus there is no need to implement it`

Chip Name:	`Not`
Inputs:	`in`
Outputs:	`out`
Function:	`If in=0 then out=1 else out=0`
Comment:	

Chip Name:	`And`
Inputs:	`a, b`
Outputs:	`out`
Function:	`If a=b=1 then out=1 else out=0`
Comment:	

Chip Name:	`Or`
Inputs:	`a, b`
Outputs:	`out`
Function:	`If a=b=0 then out=0 else out=1`
Comment:	

Chip Name:	`Xor`
Inputs:	`a, b`
Outputs:	`out`
Function:	`If a!=b then out=1 else out=0`
Comment:	

Chip Name:	`Not16`
Inputs:	`in[16]`
Outputs:	`out[16]`
Function:	`For i=0..15 out[i]=Not(in[i])`
Comment:	

Chip Name:	`And16`
Inputs:	`a[16], b[16]`
Outputs:	`out[16]`
Function:	`For i=0..15 out[i]=And(a[i],b[i])`
Comment:	

Chip Name:	`Or16`
Inputs:	`a[16], b[16]`
Outputs:	`out[16]`
Function:	`For i=0..15 out[i]=Or(a[i],b[i])`
Comment:	

Chip Name:	`Or8Way`
Inputs:	`in[8]`
Outputs:	`out`
Function:	`Out=Or(in[0],in[1],…,in[7])`
Comment:	

Chip Name:	`DMux`
Inputs:	`in, sel`
Outputs:	`a, b`
Function:	`If sel=0 then {a=in, b=0} else {a=0, b=in}`
Comment:	

Chip Name:	`Mux`
Inputs:	`a, b, sel`
Outputs:	`out`
Function:	`If sel=0 then out=a else out=b`
Comment:	

Chip Name:	`Mux16`
Inputs:	`a[16], b[16], sel`
Outputs:	`out[16]`
Function:	`If sel=0 then for i=0..15 out[i]=a[i] else for i=0..15 out[i]=b[i]`
Comment:	

Chip Name:	Mux4Way16
Inputs:	a[16], b[16], c[16], d[16], sel[2]
Outputs:	out[16]
Function:	If sel=00 then out=a else if sel=01 then out=b else if sel=10 then out=c else if sel=11 then
Comment:	The assignment operations mentioned above are all 16-bit. For example, "out=a" means "for i=0..15 out[i]=a[i]".

Chip Name:	Mux8Way16
Inputs:	a[16],b[16],c[16],d[16],e[16],f[16],g[16],h[16],sel[3]
Outputs:	out[16]
Function:	If sel=000 then out=a else if sel=001 then out=b else if sel=010 out=c ... else if sel=111 then out=h
Comment:	The assignment operations mentioned above are all 16-bit. For example, "out=a" means "for i=0..15 out[i]=a[i]"

Chip Name:	DMux4Way
Inputs:	in, sel[2]
Outputs:	a, b, c, d

Function:	If sel=00 then {a=in, b=c=d=0}
	else if sel=01 then {b=in, a=c=d=0}
	else if sel=10 then {c=in, a=b=d=0}
	else if sel=11 then {d=in, a=b=c=0}

Comment:	

Chip Name:	DMux8Way
Inputs:	in, sel[3]
Outputs:	a, b, c, d, e, f, g, h

Function:	If sel=000 then
	{a=in, b=c=d=e=f=g=h=0}
	else if sel=001 then
	{b=in, a=c=d=e=f=g=h=0}
	else if sel=010 then
	...
	else if sel=111 then
	{h=in, a=b=c=d=e=f=g=0}

Comment:	

Chip Name:	HalfAdder
Inputs:	a, b
Outputs:	sum, carry

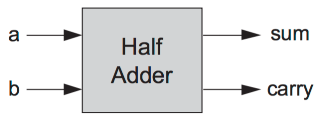
Function:	sum = LSB of a + b
	carry = MSB of a + b

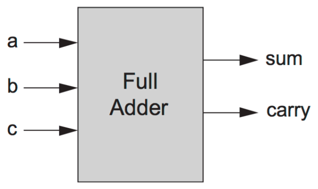
Comment:	

Chip Name:	FullAdder
Inputs:	a, b, c
Outputs:	sum, carry

Function:	sum = LSB of a + b + c
	carry = MSB of a + b + c

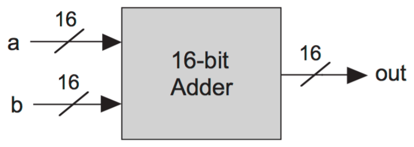
Comment:	





Chip Name:	Add16
Inputs:	a[16], b[16]
Outputs:	out[16]
Function:	out = a + b
Comment:	Integer 2's complement addition Overflow and carry are neither detected nor handled

Chip Name:	Inc16
Inputs:	in[16]
Outputs:	out[16]
Function:	out = in + 1
Comment:	Integer 2's complement addition Overflow and carry are neither detected nor handled


16
a
16
b
16-bit Adder
16
out