ES 4: Number systems (and more VHDL)

(sorry)

Steven Bell

26 September 2024

Tufts UNIVERSITY

What's the point?

Any useful computer will need to deal with negative numbers

If you know how negative numbers are represented:

a) You can explain why the computer gets "interesting" results

b) You can build a circuit that operates on negative numbers (like a computer!)

By the end of class today, you should be able to:

- Convert between hexadecimal and binary
- Convert between 2's complement binary and decimal
- Explain why we prefer 2's complement to sign-magnitude
- Explain the VHDL types: bit, std_logic, integer, unsigned, signed.

Hexadecimal

Is just a shorthand for binary numbers,
because no one can read 0001011110010101

0x 1 7 9 5

h (for hex)

0000	0	
0001	1	
0010	2	
⋮		
1000	8	
1001	9	
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Practice!

Write the following binary numbers in hex:

5 10 12 3

0101_1010 1100_0011 1011_1110_1110_1111

5 A C 3 B E E F

Write these hexadecimal numbers in binary:

FF 80 DEAD

1111 1111 1000 0000 1101 1110 1010 1101

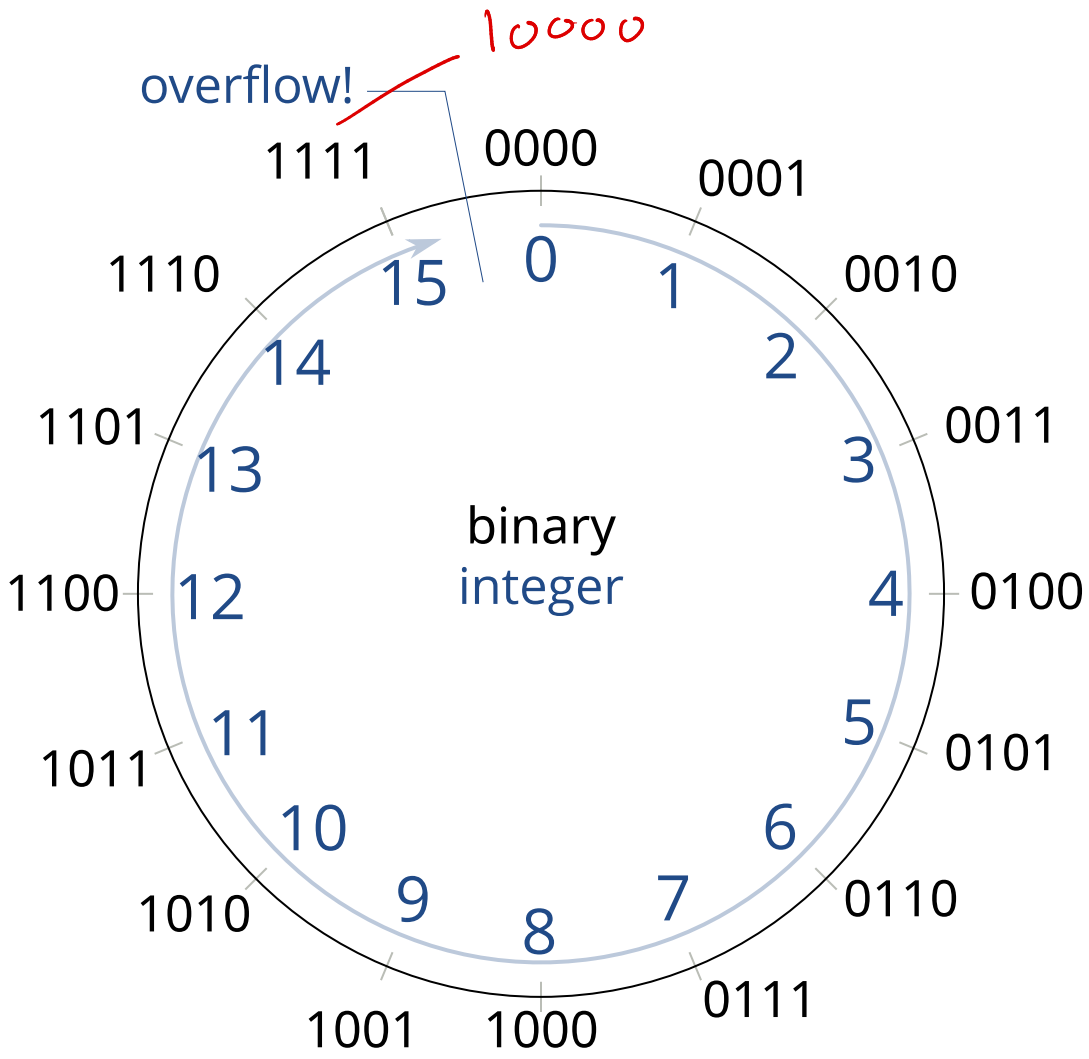
10

There are 10 kinds of people in the world.
Those who understand binary,
and those who don't.

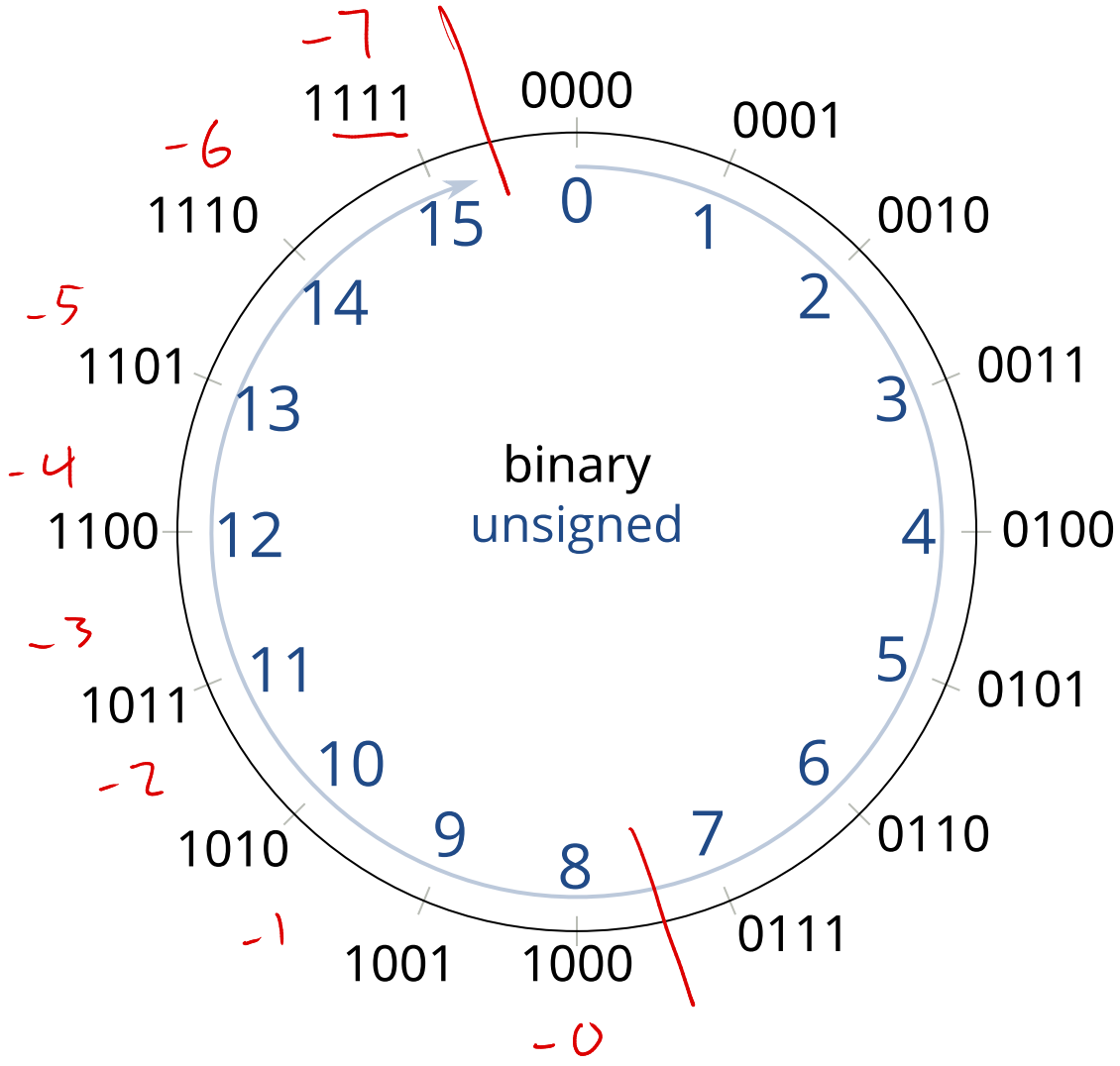
There are $\frac{10}{8}$ kinds of people in the world.

Those who understand hexadecimal,
and those who don't.

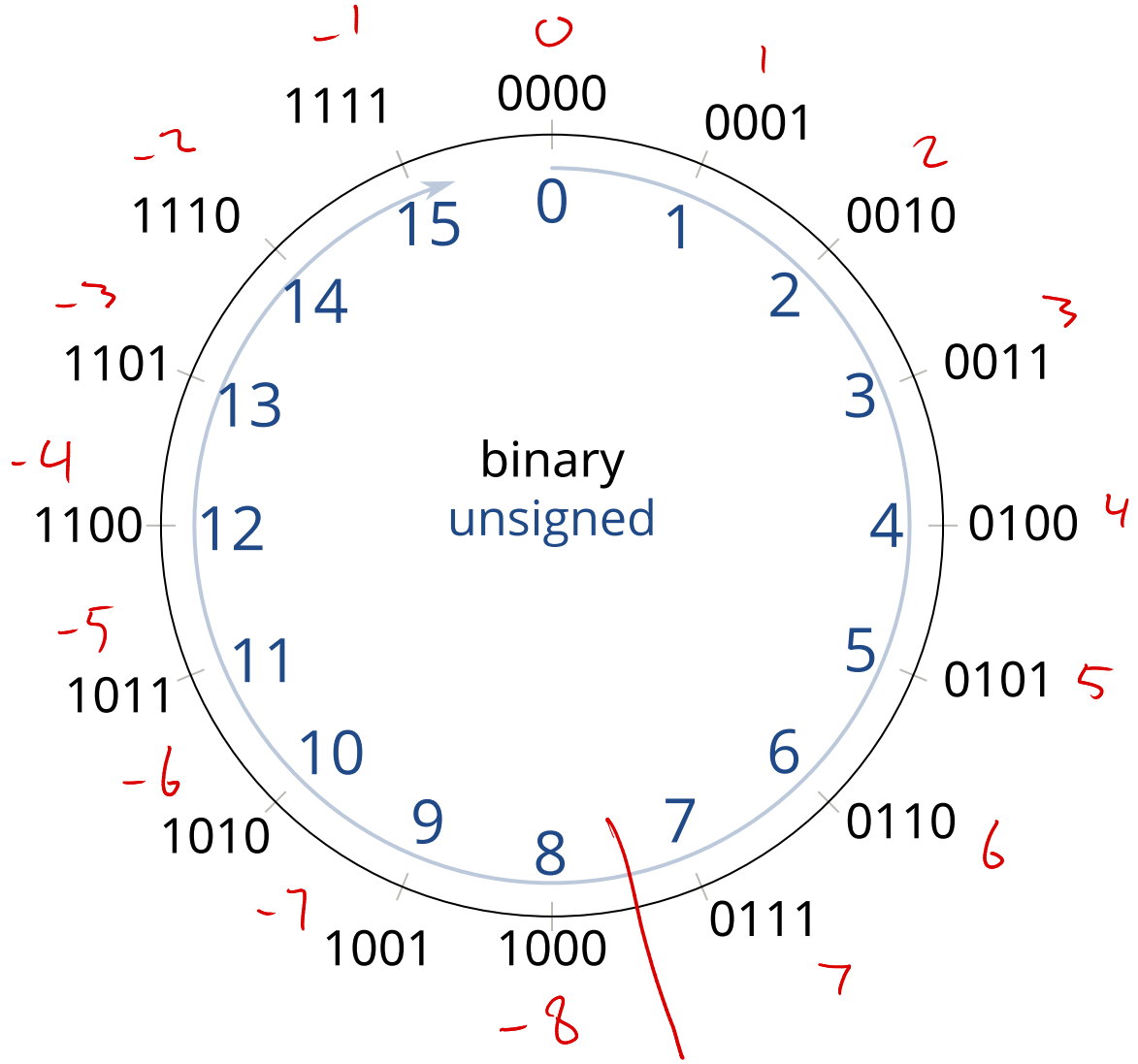
Unsigned numbers

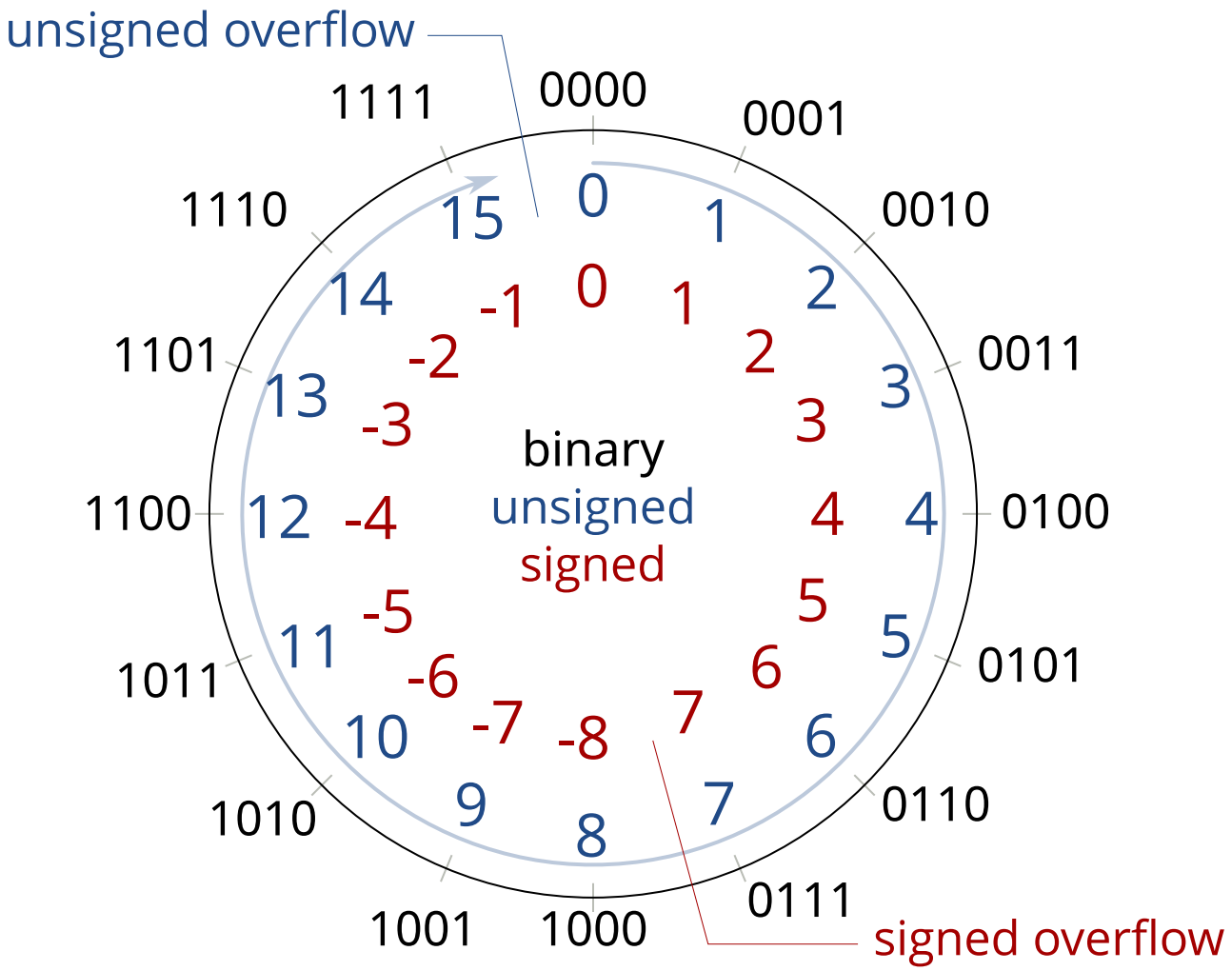


Sign-magnitude



Two's complement




unsigned overflow
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
2
3
4
5
6
7
-8
-7
-6
-5
-4
-3
-2
-1
binary
unsigned
signed
signed overflow

To write a negative number in 2's complement:

-6

- **Write the positive number** in binary

6: 0110

- **Flip all the bits** (1→0, 0→1)

1001

- **Add 1** (with all the appropriate carries)

1010

To convert negative 2's complement to decimal,

1100

- **Flip all the bits** (1→0, 0→1)

0011

- **Add 1** (with all the appropriate carries)

0100

- Write the number in decimal

-4

Wait, shouldn't this be reversed?

It turns out the result is the same either way. Try it!

To write a negative number in 2's complement:

4-bit -8

1000

0111 flip

1000 = 8

Write the positive number in binary

Flip all the bits (1→0, 0→1)

Add 1 (with all the appropriate carries)

To convert negative 2's complement to decimal,

Flip all the bits (1→0, 0→1)

Add 1 (with all the appropriate carries)

Write the number in decimal

Wait, shouldn't this be reversed?

It turns out the result is the same either way. Try it!

Practice!

Write the following numbers in 8-bit 2's complement:

-16, -1, -127

0 0 0 1 0 0 0 0 16
1 1 1 0 1 1 1 1 flip!
1 1 1 1 0 0 0 0 +1

Find the decimal value of these 2's complement numbers:

positive! = 79

01001111, 11111110, 10000000

-2 -128

1 0 1 1 0 0 0 0
+1: 1 0 1 1 0 0 0 1
128 32 16 1 = 177

Practice!

Write the following numbers in 8-bit 2's complement:

-16, -1, -127

0 0 0 0 0 0 0 1

1 1 1 1 1 1 1 0 flip

1 1 1 1 1 1 1 1 +1

Find the decimal value of these 2's complement numbers:

01001111, 11111110, 10000000

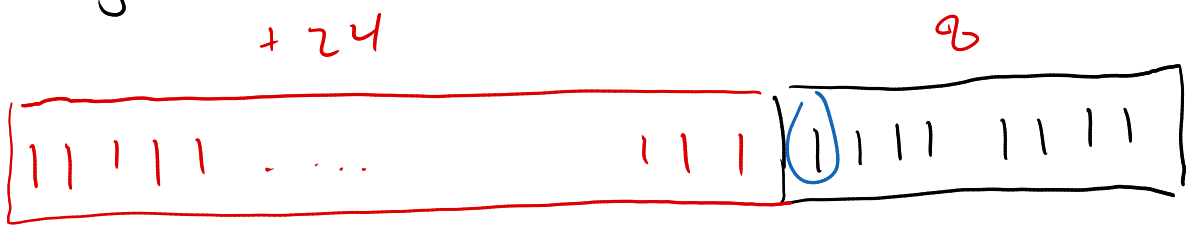
What's the point?

Practically all real systems use 2's complement

```
char x = 100; // A char is 8 bits

printf("x is %d\n", x); // Print it as a signed integer

x = x + 30; // Add 30... what will the value be?

printf("x + 30 is %d\n", x); // Print as a signed integer

printf("x + 30 (unsigned) is %u\n", x); // as unsigned integer
```

numbers.c, twoscomp.c

Sign extension



What's the point?

Practically all real systems use 2's complement

So why did we waste all that time with sign-magnitude?

Because sign-magnitude seems "obvious",
and 2's complement... "weird numbers are weird"

Floating-point numbers do use sign-magnitude

Numbers in VHDL

Types

`std_logic` Basic logic type, can take values 0, 1, X, Z (and others)

`std_logic_vector (n downto m)` Ordered group of std_logic

`unsigned (n downto m)` Like std_logic_vector, but preferred for numerically meaningful signals

`signed (n downto m)`

`integer` Poor for synthesis, but constants are integers by default

Literals

`'0', '1', 'X', 'Z'`

`"00001010", x"0c"` 8-bit binary, hex

`9x"101"` 9-bit hex

`3b"101"` 3-bit binary

`7d"101"` 7-bit decimal

`5, 38, 10000000`

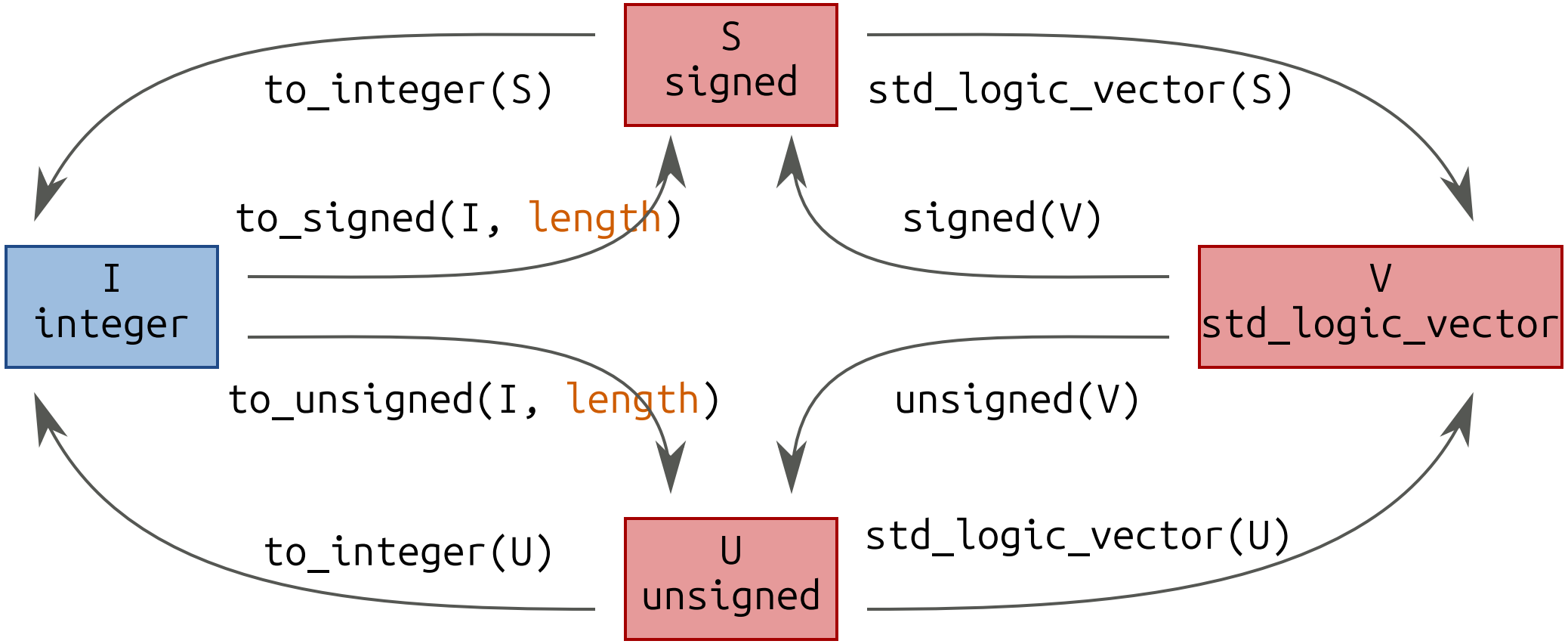
Converting numeric types

Numbers

Not good for synthesis!

Bit vectors

Have a defined bit representation
(length + ordering)



https://fpgatutorial.com/vhdl-types-and-conversions/

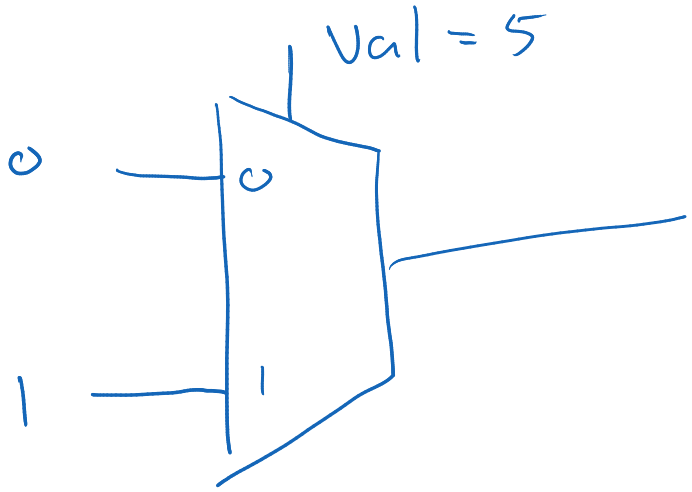
What's wrong with integers?

An integer defaults to 32 bits, which is a lot of hardware!

When/else

```
Y <= '1' when (val = x"5") else '0';
```

What does this build?



Practice!

Multiplexer, thermometer decoder, 4-bit ALU

For next time

1. Read the book (4.3, 4.9) and complete the pre-class quiz
2. Lab report 2 due this week
3. Lab report 3 due **in 2 weeks**
4. No prelab for lab 4 (yay!)