The Cryptol Language

A pure, declarative functional language built on top of haskell

No side effects

Functions depend only on their inputs, not internal state

Observe combinational circuits are just mappings from input to output

Can model bit vectors in hardware description languages

Can also model sequential circuits
Cryptol Language

A simple function call:

\( (f \text{ where } f \ x = \ x+1) \ 10 \)

An anonymous function call:

\( (\x = \ x+1) \ 10 \)

A function definition:

\[
f \ x = a+b
\text{ where }
\begin{align*}
    a &= 2\times x; \\
    b &= x-1;
\end{align*}
\]

Defining a number:

\( y = 3\times a \text{ where } a=0x1234; \)
Cryptol Language

A 17 bit number declaration and assignment:

```
x:[17];
x = 32;
```

A function declaration:

```
swab [a b c d] = [d c b a];
```

A function definition:

```
f x = a+b
where {
    a = 2*x;
    b = x-1;
};
```

Defining a variable:

```
y = 3*a where a=0x1234;
```
Cryptol Language

Sequences: all elements same type and size

[1 2 3 4 5 6 7 8 9 10];

this is of type [10][4] - 10 (size) 4 bit words (shape)

[1 .. 10];

another way to express the above sequence

[1 3 .. 10]

the above is same as [1 3 5 7 9]

[1 ..];

an infinite sequence
Cryptol Language

Index positions 2,0 of sequence, then 1 from right

\[
x = [10 -- 1]; \\
y = x@2; \\
z = x@0; \\
q = x!1; \\
r = x@@[0 1 2]; \\
r = x!![0 1 2];
\]

finally index 3 elements from left and right

\[
a = 0x15; \\
b = a@@[0 .. 3]; \\
c = 0xfe@0;
\]

Little endian - 0 bit is least significant
Cryptol Language

A tuple and its type

(True, 121, "abc")
(Bit, [12], [3][8])

finally index 3 elements from left and right

a = 0x15;
b = a@[0 .. 3];
c = 0xfe@0;

Little endian - 0 bit is least significant
Cryptol Language

A polynomial:

<|x^8 + x^5 + x + 1|>

arithmetic ops:

pmult, pdiv, pmod

Little endian - 0 bit is least significant