SKIRCH Circuit Simulator

Language Reference Manual

Jeffrey Sinckler/jcs2137 and Brian Hunter/bmh2130

11/3/2010

First iteration of the reference manual for the SKIRCH Circuit Simulator Programming Language. Written solely for educational purposes, specifically for Programming Languages and Translators.

SKIRCH LANGUAGE REFERENCE MANUAL

Lexical Conventions

Identifiers in Skirch will be similar to C. They must begin with a letter and are followed by numbers and letters. Identifiers are used to define variable names as well as function names.

```
Binary1 input = 0;
```

Keywords in Skirch are also similar to C. If-else statements, while loops, and return statements are all present in Skirch.

```
Binary1 input = circuit_function(1);
if(input == 1)
        input ^ 0;
else
        input – 0;
```

Comments in Skirch will use C conventions as well. Open: /* Close: */

```
/*Begin function*/
empty_circuit(Binary1 x)
{
}
/*End function*/
```

Skirch is a free-form language where white space is only used to separate tokens.

Constants/Literals

Skirch will use only binary types for literals. Inputs into circuits will have to use this binary type that will be defined in our library. Also, there will be a separate binary type for input into multiplexers.

Variables and Naming

Variables of the before mentioned binary types can be named and used in functions. These variables can be used with C style.

```
Binary1 input1 = 0;
Binary1 input2 = 1;
```

```
Binary1 output = circuit_function(input1, input2);
```

Expression Precedence

Basic gates will take precedence over larger gates implemented by the user. Among basic gates, precedence is equal and goes from left to right. Skirch is meant to simulate a circuit, so the input is fed in from the left and recalculated at every gate.

Type Specifiers

binary1
binary2
binary3
binaryGroup2
binaryGroup3
integer

Represents binary numbers with different amounts of bits. Circuits should only be able to take binary1 values, while multiplexers can take binary 2 and binary3 values. The group types contain dual or triple inputs. These types are accessible by calling their elements, first, second, and third as shown in later sections. Further, there is currently no support for input sets larger than three.

Declarators

Identifiers
Declarators
Declarators() for functions
{identifier, identifier...} for multiple inputs.

If and while statements

These statements will be implemented just as they are in C. This will allow users to repeatedly run their circuits on their inputs or create branched circuits based on values from a particular gate.

```
Binary1 input = 0;
while(i <= 10)
{
        circuit_function(input);
}
```

Function Declaration

In Skirch, functions that are declared are only allowed to be instructions for running a circuit. They will be defined in a similar fashion to C, and the return type will only be the number tuple that is returned. The function will have a name, parameters, and a definition.

```
Binary1 and_gate_function(Binary1 inp1, Binary1 inp2)
{
        return inp1 ^ inp2;
}
BinaryGroup2 split_and_or_function(BinaryGroup2 group1)
{
        BinaryGroup2 result;
        result.first = group1.first ^ 0;
        result.second = group1.second o 1;
        return result;
}
```

Multiple Inputs

Skirch will provide a type that groups together input values. Some circuits require multiple inputs and also return multiple inputs. Users will be able to chain together not only functions with single inputs, but with multiple inputs. Users will be able to write functions and pass in a tuple value and get a tuple value as the result of the circuit function. This can subsequently be passed into another circuit function that takes similar input types. This is implemented using the BinaryGroup2 and BinaryGroup3 types.

Standard Library

The Skirch standard library provides basic operators for the following gates:
AND '^'
OR 'o'
NOR 'n'
NOT '-'
NAND 'a'

Skirch also provides library functions for the basic flip flops:

RS Flip-flop
JK Flip-flop
D Flip-flop

These gates and flip flop functions are of course available to the user when he writes his own circuit functions.

Skirch also provides basic functions that allow users to run a circuit a certain number of times, using the resulting output as the input for the next iteration. There are two versions of this method. The first version only prints when the function runs through to the end. The second function prints the output at every iteration. The user is able to specify the number of times that he wants the iteration to run.

Skirch also provides function testing. There is a method deticated to running test iterations for every value of input and printing out the values. Users will be able to use this function to easily test their circuits and see if the desired output is reached.