VerilogParser – Guide

VerilogParser is a C++ library that can be used to convert a Verilog gate level netlist file into a graph. The output graph can be used to display information about the circuit and simulate the circuit's outputs.

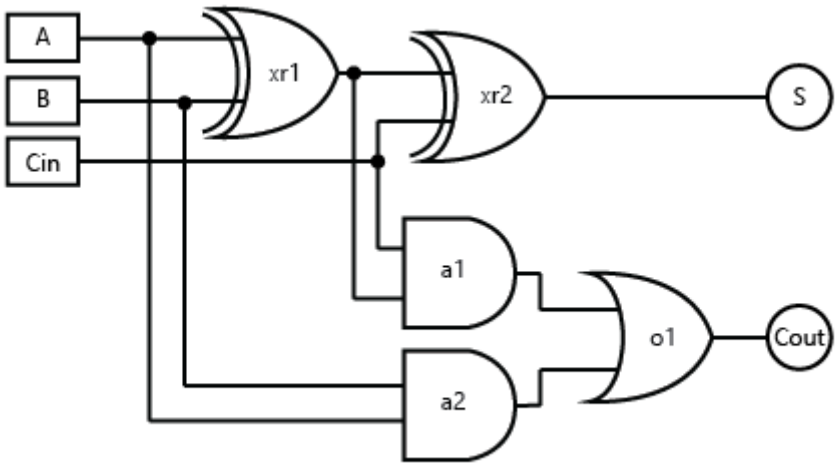
The nodes represent the gates in the circuit as well as the IO ports.
The edges represent the wires connecting these nodes.

For example the following Verilog module represents this circuit:

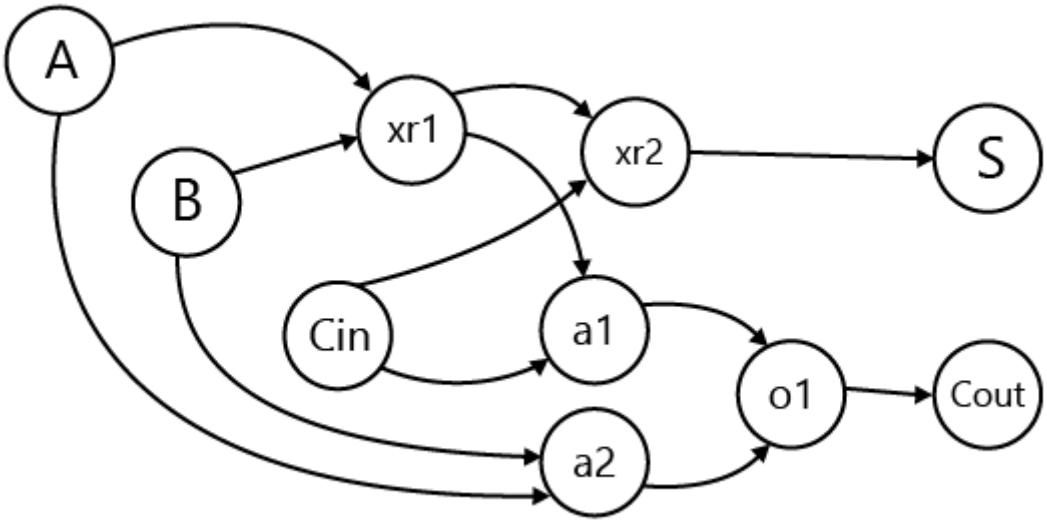
```
module FullAdder (A, B, S, Cin, Cout);
  // IO ports
  input A;
  input B;
  input Cin;
  output Cout;
  output S;

  // Wires
  wire w0;
  wire w1;
  wire w2;
  wire w3;

  // Gates
  xor xr1 (w1, A, B);
  xor xr2 (S, w1, Cin);
  and a1 (w2, w1, Cin);
  and a2 (w3, A, B);
  or o1 (Cout, w2, w3);
endmodule
```



This circuit will be converted into a graph as shown below.



Nodes:

Node index	Name	Type	IsInput	IsOutput
0	A	Input Port	Y	N
1	B	Input Port	Y	N
2	Cin	Input Port	Y	N
3	Cout	Output Port	N	Y
4	S	Output Port	N	Y
5	xr1	xor	N	N
6	xr2	xor	N	N
7	a1	and	N	N
8	a2	and	N	N
9	o1	or	N	N

Nodes Graph (Adjacency Matrix):

	A	B	Cin	Cout	S	xr1	xr2	a1	a2	o1
A	0	0	0	0	0	1	0	0	1	0
B	0	0	0	0	0	1	0	0	1	0
Cin	0	0	0	0	0	0	1	1	0	0
Cout	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0
xr1	0	0	0	0	0	0	1	1	0	0
xr2	0	0	0	0	1	0	0	0	0	0
a1	0	0	0	0	0	0	0	0	0	1
a2	0	0	0	0	0	0	0	0	0	1
o1	0	0	0	1	0	0	0	0	0	0

Usage Examples:

Example 1: Display circuit properties and IO ports

```
// Example 1
Circuit circuit;
circuit.parseFile("FullAdder.v");

// Display circuit properties
cout << "Module name:       " << circuit.getModuleName() << endl;
cout << "Number of nodes:   " << circuit.getNodesCount() << endl;
cout << "Number of inputs:  " << circuit.getInputNodesCount() << endl;
cout << "Number of outputs: " << circuit.getOutputNodesCount() << endl;
cout << "Number of gates:   " << circuit.getGatesCount() << endl;
cout << endl;


// Traverse the input nodes only
cout << "Input Nodes:" << endl;
for (int i = 0; i < circuit.getInputNodesCount(); i++) {
    cout << "  " << circuit.inputNode(i).getName() << endl;
}
cout << endl;

// Traverse the output nodes only
cout << "Output Nodes:" << endl;
for (int i = 0; i < circuit.getOutputNodesCount(); i++) {
    cout << "  " << circuit.outputNode(i).getName() << endl;
}
cout << endl;
```

Example 2: Display the adjacency matrix

```
// Example 2
Circuit circuit;
circuit.parseFile("FullAdder.v");

// Display the adjacency matrix
cout << " Adjacency matrix: " << endl << endl;
for (int i = 0; i < circuit.getNodesCount(); i++) {
    cout << "  ";

    for (int j = 0; j < circuit.getNodesCount(); j++) {
        cout << circuit[i][j];
    }
    cout << endl;
}

cout << endl;
```

Example 3: Display Nodes' properties

```
// Example 3
Circuit circuit;
circuit.parseFile("FullAdder.v");

// Display all the nodes in the circuit
cout << "All nodes: " << endl;
for (int i = 0; i < circuit.getNodesCount(); i++) {
    cout << "  " << circuit.node(i).getType() << " " << circuit.node(i).getName() << endl;
}
cout << endl;


// Access a certain node
Node X = circuit.node(6);            // access a node by its index
Node B = circuit.node("B");          // access a node by its name


// Display properties of a certain node
cout << "Node properties:" << endl;
cout << "  name:      " << X.getName() << endl;
cout << "  type:      " << X.getType() << endl;
cout << "  is input:  " << (X.isInputPort() ? "Yes" : "No") << endl;
cout << "  is output: " << (X.isOutputPort() ? "Yes" : "No") << endl;
cout << "  is gate:   " << (X.isGate() ? "Yes" : "No") << endl;
cout << "  trise:     " << X.getTRise() << endl;
cout << "  tfall:     " << X.getTFall() << endl;
cout << endl;


// Check the input connections of a certain node
cout << "Inputs: " << endl;
for (int i = 0; i < X.getInputsCount(); i++) {
    Node t = X.inputNode(i);
    cout << "  " << t.getType() << " " << t.getName() << endl;
}
cout << endl;


// Check the output connections of a certain node
cout << "Outputs: " << endl;
for (int i = 0; i < X.getOutputsCount(); i++) {
    Node t = X.outputNode(i);
    cout << "  " << t.getType() << " " << t.getName() << endl;
}
cout << endl;
```

Example 4: Depth first search graph traversal using the adjacency matrix (display the DFS visit order)

```
// Example 4
Circuit circuit;
circuit.parseFile("FullAdder.v");

// Depth first search
int order = 0;
vector<int> visitOrder(circuit.getNodesCount(), 0);

for (int i = 0; i < circuit.getInputNodesCount(); i++) {
    string nodeName = circuit.inputNode(i).getName();
    int nodeIndex = circuit.getNodeIndex(nodeName);

    stack<int> S;
    S.push(nodeIndex);
    while (!S.empty()) {
        int s = S.top();
        S.pop();

        if (visitOrder[s] < 1) {
            visitOrder[s] = ++order;

            for (int j = circuit.getNodesCount() - 1; j >= 0; j--)
                if (circuit[s][j] == 1)
                    S.push(j);
        }
    }
}

for (int i = 0; i < visitOrder.size(); i++) {
    cout << "Node:         " << circuit.node(i).getName() << endl;
    cout << "Visit order:  " << visitOrder[i] << endl;
    cout << " ---------------------------------- " << endl;
}
```