Embedded Systems

Course Project Report :

Group Members : Abhijeet Verma (B19CSE001)
AdityaRaje Ashok devade (B19CSE005)

Project Topic : Enumeration of design behaviors from RTL-based datapath-intensive designs

What have we done?

- Developed a Verilog code Parser (900+ lines C code) from scratch which parses Datapath designs or datapath components of a design only . By datapath I mean how input data passes through complex connections of Gates and operations to give the final output .
- **Input :** Give a single verilog file to the parser
- **Output :** Parser outputs line by line explanation of datapath operations along with design overview of circuit and each module in the verilog file .
- **Limitation** : The parser cannot take multiple verilog files at once and connect them together to give an overall circuit . Meaning the whole RTL design should be in a single verilog file and should not broken in various components.

Detailed TestCases Analysis

Testcase 1 :

Input file :

rtl > half_adder.v

```
module half_adder(input a,b, output sum, carry);

    assign sum = a ^ b;
    assign carry = a & b;

endmodule
```

Output :

```
bomberabhi17@GeNeSis:/mnt/d/abhijeet/desktop/verilog-parser-master$ ./veri "halfadder.v"

 ---------------MODULE half_adder  DESIGN OVERVIEW ---------------
Total inputs: 2
a b

Total outputs: 2
sum carry

Total wires: 0


 Total gates: 0


Total registers: 0

 --------------- LINE BY LINE ANALYSIS ---------------
Performing wire assignments as :  sum = a ^ b
Performing wire assignments as :  carry = a & b

--------------- CIRCUIT halfadder.v  DESIGN OVERVIEW ---------------
 Overall Circuit size: 4
Total primary inputs: 2
a b

Total outputs: 2
sum carry

Total gates: 0


This is how Input is converted to Output through Datapath (Combination of Gates)
Following connections of wires are also Performed using Assign Keyword
 sum = a ^ b
 carry = a & b
```

Testcase 2 :

Input file :

fulladder.v

```
1   module full_adder(a, b, cin,sum, cout);
2   input a,b,cin;
3   output sum,cout;
4   wire sum1,carry1,carry2;
5   assign sum1 = a ^ b;
6   assign carry1 = a & b;
7   assign sum = cin ^ sum1;
8   assign carry2 = cin & sum1;
9   assign cout = carry1 | carry2;
0   endmodule
1
```

Output :

```
bomberabhi17@GeNeSis:/mnt/d/abhijeet/desktop/verilog-parser-master$ ./veri "fulladder.v"

 ---------------MODULE full_adder  DESIGN OVERVIEW ---------------
Total inputs: 3
a b cin

Total outputs: 2
sum cout

Total wires: 3
sum1 carry1 carry2

 Total gates: 0


Total registers: 0

 --------------- LINE BY LINE ANALYSIS ---------------
Performing wire assignments as :  sum1 = a ^ b
Performing wire assignments as :  carry1 = a & b
Performing wire assignments as :  sum = cin ^ sum1
Performing wire assignments as :  carry2 = cin & sum1
Performing wire assignments as :  cout = carry1 | carry2

--------------- CIRCUIT fulladder.v  DESIGN OVERVIEW ---------------
 Overall Circuit size: 8
Total primary inputs: 3
a b cin

Total outputs: 2
sum cout

Total gates: 0


This is how Input is converted to Output through Datapath (Combination of Gates)
Following connections of wires are also Performed using Assign Keyword
 sum1 = a ^ b
 carry1 = a & b
 sum = cin ^ sum1
 carry2 = cin & sum1
 cout = carry1 | carry2
```

Testcase 3 :

Input file :

c17.v

```
1   module c17 (N1,N2,N3,N6,N7,N22,N23);
2
3   input N1,N2,N3,N6,N7;
4
5   output N22,N23;
6
7   wire N10,N11,N16,N19;
8
9   nand NAND2_1 (N10, N1, N3);
0   nand NAND2_2 (N11, N3, N6);
1   nand NAND2_3 (N16, N2, N11);
2   nand NAND2_4 (N19, N11, N7);
3   nand NAND2_5 (N22, N10, N16);
4   nand NAND2_6 (N23, N16, N19);
5
6   endmodule
```

Output :

```
bomberabhi17@GeNeSis:/mnt/d/abhijeet/desktop/verilog-parser-master$ ./veri "c17.v"

 ---------------MODULE c17  DESIGN OVERVIEW ---------------
Total inputs: 5
N1 N2 N3 N6 N7

Total outputs: 2
N22 N23

Total wires: 4
N10 N11 N16 N19

 Total gates: 6
NAND2_1 NAND2_2 NAND2_3 NAND2_4 NAND2_5 NAND2_6

Total registers: 0

 --------------- LINE BY LINE ANALYSIS ---------------
 NAND2_1 = nand ( N10, N1, N3, )
 wire 7 : N10
 wire 0 : N1
 wire 11 : NAND2_1
 wire 2 : N3
 NAND2_2 = nand ( N11, N3, N6, )
 wire 8 : N11
 wire 12 : NAND2_2
 wire 3 : N6
 NAND2_3 = nand ( N16, N2, N11, )
 wire 9 : N16
 wire 1 : N2
 wire 13 : NAND2_3
 NAND2_4 = nand ( N19, N11, N7, )
 wire 10 : N19
 wire 14 : NAND2_4
 wire 4 : N7
 NAND2_5 = nand ( N22, N10, N16, )
 wire 5 : N22
 wire 15 : NAND2_5
 NAND2_6 = nand ( N23, N16, N19, )
 wire 6 : N23
 wire 16 : NAND2_6

--------------- CIRCUIT c17.v  DESIGN OVERVIEW ---------------
 Overall Circuit size: 17
Total primary inputs: 5
N1 N2 N3 N6 N7

Total outputs: 2
N22 N23

Total gates: 6


This is how Input is converted to Output through Datapath (Combination of Gates)
c->wire[0]->type: I, ID: 7,  name: N10,
Inputs (0):
Outputs (1): 11
c->wire[1]->type: I, ID: 0,  name: N1,
Inputs (0):
Outputs (1): 11
c->wire[2]->type: nand, ID: 11,  name: NAND2_1,
Inputs (2): 7 0
Outputs (1): 2
c->wire[3]->type: I, ID: 2,  name: N3,
Inputs (1): 11
Outputs (0):
c->wire[4]->type: I, ID: 8,  name: N11,
Inputs (1): 13
Outputs (1): 12
c->wire[5]->type: nand, ID: 12,  name: NAND2_2,
Inputs (2): 8 2
Outputs (1): 3
c->wire[6]->type: I, ID: 3,  name: N6,
Inputs (1): 12
Outputs (0):
c->wire[7]->type: I, ID: 9,  name: N16,
Inputs (1): 15
Outputs (1): 13
c->wire[8]->type: I, ID: 1,  name: N2,
Inputs (0):
Outputs (1): 13
c->wire[9]->type: nand, ID: 13,  name: NAND2_3,
Inputs (2): 9 1
Outputs (1): 8
c->wire[10]->type: I, ID: 10,  name: N19,
Inputs (1): 16
Outputs (1): 14
c->wire[11]->type: nand, ID: 14,  name: NAND2_4,
Inputs (2): 10 8
Outputs (1): 4
c->wire[12]->type: I, ID: 4,  name: N7,
Inputs (1): 14
Outputs (0):
c->wire[13]->type: I, ID: 5,  name: N22,
Inputs (0):
Outputs (1): 15
c->wire[14]->type: nand, ID: 15,  name: NAND2_5,
Inputs (2): 5 7
Outputs (1): 9
c->wire[15]->type: I, ID: 6,  name: N23,
Inputs (0):
Outputs (1): 16
c->wire[16]->type: nand, ID: 16,  name: NAND2_6,
Inputs (2): 6 9
Outputs (1): 10
```

Note here the numbers after input and outputs are actually the IDs of the wires which act as the input/output. This gives us the overall design of circuit

Testcase 4 :

Input file :

```
module simple (inp1,inp2,tau2015_clk,out);
// Start PIs
input inp1;
input inp2;
input tau2015_clk;

// Start POs
output out;

// Start wires
wire n1;
wire n2;
wire n3;
wire n4;
wire inp1;
wire inp2;
wire tau2015_clk;
wire out;

// Start cells
nand u1 ( inp1, inp2, n1 );
and f1 ( n2, tau2015_clk, n3 );
inv u2 ( n3, n4 );
inv u3 ( n4, out );
xor u4 ( n1, n3, n2 );

endmodule
```

Output :

```
bomberabhi17@GeNeSis:/mnt/d/abhijeet/desktop/verilog-parser-master$ ./veri "simple.v"

 ---------------MODULE simple  DESIGN OVERVIEW ---------------
Total inputs: 3
inp1 inp2 tau2015_clk

Total outputs: 1
out

Total wires: 8
n1 n2 n3 n4 inp1 inp2 tau2015_clk out

 Total gates: 5
u1 f1 u2 u3 u4

Total registers: 0

 ---------------- LINE BY LINE ANALYSIS ---------------
 u1 = nand ( inp1, inp2, n1, )
 wire 0 : inp1
 wire 1 : inp2
 wire 12 : u1
 wire 4 : n1
 f1 = and ( n2, tau2015_clk, n3, )
 wire 5 : n2
 wire 2 : tau2015_clk
 wire 13 : f1
 wire 6 : n3
 u2 = not n3 wire 14 : u2
 wire 7 : n4
 u3 = not n4 wire 15 : u3
 wire 3 : out
 u4 = xor ( n1, n3, n2, )
 wire 16 : u4

---------------- CIRCUIT simple.v  DESIGN OVERVIEW ---------------
 Overall Circuit size: 17
Total primary inputs: 3
inp1 inp2 tau2015_clk

Total outputs: 1
out

Total gates: 5


This is how Input is converted to Output through Datapath (Combination of Gates)
c->wire[0]->type: I, ID: 0,  name: inp1,
Inputs (0):
Outputs (1): 12
c->wire[1]->type: I, ID: 1,  name: inp2,
Inputs (0):
Outputs (1): 12
c->wire[2]->type: nand, ID: 12,  name: u1,
Inputs (2): 0 1
Outputs (1): 4
c->wire[3]->type: I, ID: 4,  name: n1,
Inputs (1): 12
Outputs (0):
c->wire[4]->type: I, ID: 5,  name: n2,
Inputs (1): 16
Outputs (1): 13
c->wire[5]->type: I, ID: 2,  name: tau2015_clk,
Inputs (0):
Outputs (1): 13
c->wire[6]->type: and, ID: 13,  name: f1,
Inputs (2): 5 2
Outputs (1): 6
c->wire[7]->type: I, ID: 6,  name: n3,
Inputs (1): 13
Outputs (0):
c->wire[8]->type: inv, ID: 14,  name: u2,
Inputs (1): 6
Outputs (1): 7
c->wire[9]->type: I, ID: 7,  name: n4,
Inputs (1): 14
Outputs (0):
c->wire[10]->type: inv, ID: 15,  name: u3,
Inputs (1): 7
Outputs (1): 3
c->wire[11]->type: I, ID: 3,  name: out,
Inputs (1): 15
Outputs (0):
c->wire[12]->type: xor, ID: 16,  name: u4,
Inputs (2): 4 6
Outputs (1): 5
```

Note here the numbers after input and outputs are actually the IDs of the wires which act as the input/output. This gives us the overall design of circuit

Testcase 6 :

Input file :

last.v

```
1   module c3_path (nx1,nx3,nx2,nx4,nx33,nx44,nx12);
2
3   // Start PIs
4   input nx1;
5   input nx3;
6   input nx2;
7   input nx4;
8
9   // Start POs
10  output nx33;
11  output nx44;
12  output nx12;
13
14  // Start wires
15  wire nx1;
16  wire nx3;
17  wire nx33;
18  wire nx44;
19  wire nx12;
20  wire nx2;
21  wire nx4;
22
23  // Start cells
24  not NOT_X1 inst_2 ( nx44, nx4 );
25  inv INV_X1 inst_1 ( nx33, nx3 );
26  nand NAND2_X1 inst_0 ( nx12, nx2, nx1 );
27
28  endmodule
```

Output :

```
bomberabhi17@GeNeSis:/mnt/d/abhijeet/desktop/verilog-parser-master$ ./veri "last.v"

 ---------------MODULE c3_path  DESIGN OVERVIEW ---------------
Total inputs: 4
nx1 nx3 nx2 nx4

Total outputs: 3
nx33 nx44 nx12

Total wires: 7
nx1 nx3 nx33 nx44 nx12 nx2 nx4

 Total gates: 3
NOT_X1 INV_X1 NAND2_X1

Total registers: 0

 ---------------- LINE BY LINE ANALYSIS ---------------
 NOT_X1 = not inst_2 wire 0 : inst_2
 wire 5 : nx44
 wire 14 : NOT_X1
 wire 3 : nx4
 INV_X1 = not inst_1 wire 0 : inst_1
 wire 4 : nx33
 wire 15 : INV_X1
 wire 1 : nx3
 NAND2_X1 = nand ( inst_0, nx12, nx2, nx1, )
 wire 0 : inst_0
 wire 6 : nx12
 wire 2 : nx2
 wire 16 : NAND2_X1
 wire 0 : nx1

---------------- CIRCUIT last.v  DESIGN OVERVIEW ---------------
 Overall Circuit size: 17
Total primary inputs: 4
nx1 nx3 nx2 nx4

Total outputs: 3
nx33 nx44 nx12

Total gates: 3


This is how Input is converted to Output through Datapath (Combination of Gates)
c->wire[0]->type: I, ID: 0,  name: inst_2,
Inputs (0):
Outputs (1): 14
c->wire[1]->type: I, ID: 5,  name: nx44,
Inputs (0):
Outputs (1): 14
c->wire[2]->type: not, ID: 14,  name: NOT_X1,
Inputs (2): 0 5
Outputs (1): 3
c->wire[3]->type: I, ID: 3,  name: nx4,
Inputs (1): 14
Outputs (0):
c->wire[4]->type: I, ID: 0,  name: inst_1,
Inputs (0):
Outputs (1): 15
c->wire[5]->type: I, ID: 4,  name: nx33,
Inputs (0):
Outputs (1): 15
c->wire[6]->type: inv, ID: 15,  name: INV_X1,
Inputs (2): 0 4
Outputs (1): 1
c->wire[7]->type: I, ID: 1,  name: nx3,
Inputs (1): 15
Outputs (0):
c->wire[8]->type: I, ID: 0,  name: inst_0,
Inputs (0):
Outputs (1): 16
c->wire[9]->type: I, ID: 6,  name: nx12,
Inputs (0):
Outputs (1): 16
c->wire[10]->type: I, ID: 2,  name: nx2,
Inputs (0):
Outputs (1): 16
c->wire[11]->type: nand, ID: 16,  name: NAND2_X1,
Inputs (3): 0 6 2
Outputs (1): 0
c->wire[12]->type: I, ID: 0,  name: nx1,
Inputs (1): 16
Outputs (0):
```

Note here the numbers after input and outputs are actually the IDs of the wires which act as the input/output. This gives us the overall design of circuit