Summer Fellowship Report

On

Implementation of INOUT Ports

Submitted by

Vadisa Yamini

Under the guidance of

Prof.Kannan M. Moudgalya
Chemical Engineering Department
IIT Bombay

June 26, 2020

Acknowledgment

I would like to express my gratefulness to the FOSSEE team for making me a part of this project and giving me this chance to work on an interesting topic .This journey with the project was a great experience and I gained a lot of knowledge on my way.It also gave me a fruitful experience of work from home during the time of lock-down which I might not have got from anywhere else.

I would like to thank the eSim team for helping me and providing all the resources required to work with. I specially would like to thank my mentors Mr. Rahul Paknikar, Mrs. Gloria Nandihal and Mr. Saurabh Bansode for always helping and supporting me when ever I got struck.The guidance and steps provided by them helped me improve my work throughout the project. I would also specially like to thank Mr. Ashutosh Jha for sharing his knowledge and clearing my doubts.His inputs helped me to go through the project.

This process of learning and experimenting things is a whole new experience and am sure that ill take these points along with me in my career and carry this memorable experience throughout my life.

Contents

Chapter 1

Introduction

eSim is a free/libre and open source EDA tool for circuit design, simulation, analysis and PCB design developed by FOSSEE, IIT Bombay. It is an integrated tool built using free/libre and open source software such as KiCad, Ngspice, NGHDL and GHDL.

1.1 NGHDL

NGHDL is a mixed mode circuit simulator developed by FOSSEE, using NgSpice and GHDL. In **NGHDL**, NgSpice is used to simulate the analog components and GHDL is used to simulate the digital components, where the analog and digital components are communicating through socket.As in ngspice its difficult to write our own code models for digital circuits and many people being familiar with VHDL language is the main reason for introducing NGHDL.

1.2 INOUT Ports

NGHDL helps the user to design a digital system easily by writing a VHDL code for its behaviour. Ports are a part of the block interface in VHDL.The 3 main important port types in VHDL are IN, OUT and INOUT.Signal can read a value from a port of mode **in**, but is not allowed to assign a value to it.In **out** it is allowed to make signal assignments, but it is not legal to read from it.**Inout** is a bi-directional port. Both assignments to such a port and reading from it are allowed.

1.3 Motivation

In the process of circuit simulation using NGHDL many models require INOUT port for its functionality along side with IN and OUT ports.For circuits requiring simulations with a microcontrollers INOUT ports play even more important role.

The current version of NGHDL only supports IN and OUT port directions. Thus the importance of INOUT ports in a circuit simulation drives this project forward.

Chapter 2

Literature Survey

2.1 Ngspice

As the NGHDL is developed by using Ngspice and GHDL ,understanding of addition of codemodel in Ngspice is important for understanding and making further improvements in NGHDL .

The ngspice simulator already includes XSPICE libraries of predefined models and node types that span the analog and digital domains. XSPICE:

1.The **Interface Specification File** is a text file that describes, in a tabular format, information needed for the code model to be properly interpreted by the simulator when it is placed with other circuit components into a SPICE deck. This information includes such things as the parameter names, parameter default values, and the name of the model itself

2.The **Model Definition File** contains a C programming language function definition. This function specifies the operations to be performed within the model on the data passed to it by the simulator.

2.2 NGHDL Working

In a simulation including NGHDL model,when simulator gives input to the model Ngspice looks for that model and actually calls GHDL to get the results.NGHDL creates a dummy model for ngspice and Generation of C files for simulation.When a VHDL code is updated in NGHDL cfunc.mod and ifspec.ifs files for ngspice are created.Then the model file is generated containing testbench ,VHDL code, ghdlserver.c,startserver.sh and other VHDL and vhpi packages files.

While simulation NgSpice runs cfunc.c of the respective model which acts as a client. This client starts server shell script (`start_server.sh`) which executes Testbench file linked with the Server file (**ghdlserver.c**).Testbench acts as a controller of

NGHDL Server and commands the server file to connect and interact with the client.

In simple,the inputs from simulator are taken by cfunc.mod and are given to VHDL testbench which is linked with ghdlserver.c.The digital operation is done according to the logic in VHDL code uploaded and the results are send back to cfunc.mod through testbench which in final is given to eeschema in eSim.

2.3 INOUT port in VHDL

Inout port in VHDL is implemented by using tristate logic. You can read and write this signal. But that use a three-states buffer. There is always a enable signal or pin which decides the functioning of INOUT pin.(zero value decides out case and value 1 decides in or vice versa based on the example).In some cases there can be more then 1 enable signals.Thus here when the port needs to be used as output it can be directly assigned value.If the port needs to be used the input then the port first should be given value of high impedance Z and then its value can be read to other signals.Assigning the value Z while using it as input is the logic behind tristate buffer which is used in inout implementation.

Chapter 3

Problem Statement

To Implement INOUT ports in NGHDL in such a way it supports all practical circuits and simulates optimal results successfully.

3.1 Approach

To implement INOUT Ports it is important to identify the changes required to be added to the current files and then include those in the process of NGHDL.

As the NGHDL module doesnt agree VHDL file with INOUT port while uploading required changes are made after uploading.This method shows result as the functioning of NGHDL is by ngspice and ghdl which support INOUT ports individually.

This process is mainly divided into 3 steps/case.
Case 1:
Port is declared as IN and modified as OUT in the DUT files
Case 2:
Port is declared as OUT and modified as IN in the DUT files
Case 3:
Port used as both IN and OUT in same project.

Chapter 4

Implementation

4.1 Case 1

Port is declared as IN and modified as OUT in the DUT files.
Steps involved:

1. Upload the VHDL file in NGHDL with tristate logic used for inout port but still defined as IN in the VHDL code.(let the port name be bi).
2. Changing the model files for simulation: The files which are required to be changed after uploading are: VHDL codefile, VHDL testbench, `Connection_info.txt`, cfunc.mod, ifspecs.ifs.

   In VHDL code,in entity the port declaration is changed to *inout*.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;
ENTITY shu IS
    PORT(
        inp    : IN STD_LOGIC;
        bi     : inout STD_LOGIC;
        outo   : OUT STD_LOGIC);
END shu;

ARCHITECTURE min OF shu IS
SIGNAL  a   : STD_LOGIC;  -- DFF that stores
SIGNAL  oe  : STD_LOGIC := '1';
SIGNAL  b   : STD_LOGIC ;  -- DFF that stores
BEGIN
    PROCESS (oe,a,bi,inp)
        BEGIN
        a <= inp;
        IF( oe = '0') THEN
        bi <= 'Z';
        b <= bi ;
        ELSif( oe = '1') Then
        bi <= a;
        b <= '1';
        END IF;
    END PROCESS;
        outo <= b;
END min;
```

Figure 4.1

   In VHDL testbench the port declaration is changed to INOUT and also changes are made in the VHPI part where the testbench is linked with ghdlserver.c .

Here function takes input from server and gives it to VHDL design by `Vhpi_Get_Port_Value` and takes output values from digital design and send it through server to client cfunc.mod by `Vhpi_Set_Port_Value` function.Thus the part `Vhpi_Get_Port_Value` of bi (inout port) is deleted and `Vhpi_Get_Port_Value` is added as shown below.

```
begin
    while true loop
        wait until clk_s = '0';

        obj_ref := Pack_String_To_Vhpi_String("inp");
        Vhpi_Get_Port_Value(obj_ref,inp_v,1);
        inp <= To_Std_Logic(inp_v);

        wait for 1 us;

        bi_v := Pack_String_To_Vhpi_String(To_String(bi));
        obj_ref := Pack_String_To_Vhpi_String("bi");
        Vhpi_Set_Port_Value(obj_ref,bi_v,1);

        outo_v := Pack_String_To_Vhpi_String(To_String(outo));
        obj_ref := Pack_String_To_Vhpi_String("outo");
        Vhpi_Set_Port_Value(obj_ref,outo_v,1);
        report "test3 - "& std_logic'image(bi);
        report "Iteration - "& integer'image(count) severity note;
        count := count + 1;
    end loop;
```

Figure 4.2

In `Connection_info.txt` file located in model folder ,the bi pin is defined as OUT instead of IN.
In Ngspice side,in ifspecs.ifs which acts as interface,bi direction is changed to INOUT and description is also changed as inout pin.

```
PORT_TABLE:
Port_Name:  bi
Description:    "inout port bi"
Direction:  inout
Default_Type:   d
Allowed_Types:  [d]
Vector: yes
Vector_Bounds:  [1 1]
Null_Allowed:   no
```

Figure 4.3

In cfunc.mod bi is added as new output and from all the functions taking input values from the schematic bi is removed.

```
void cm_shu(ARGS)
{
    Digital_State_t *_op_bi, *_op_bi_old;
    Digital_State_t *_op_outo, *_op_outo_old;

            // Declaring components of Client
            FILE *log_client = NULL;
            log_client=fopen("client.log","a");
            int socket_fd, bytes_recieved;
            char send_data[1024];
            char recv_data[1024];
            char *key_iter;
            struct hostent *host;
            struct sockaddr_in server_addr;
            int sock_port = 5000+PARAM(instance_id);
            char temp_inp[1024];
```

Figure 4.4

In similar manner many functional changes are made in cfunc.mod with a main motive to remove it as a input and define it as a OUT port.

3. With changes made above appropriate schematic is drawn with bi used as OUT pin and optimal simulation results were observed.
   Schematic:

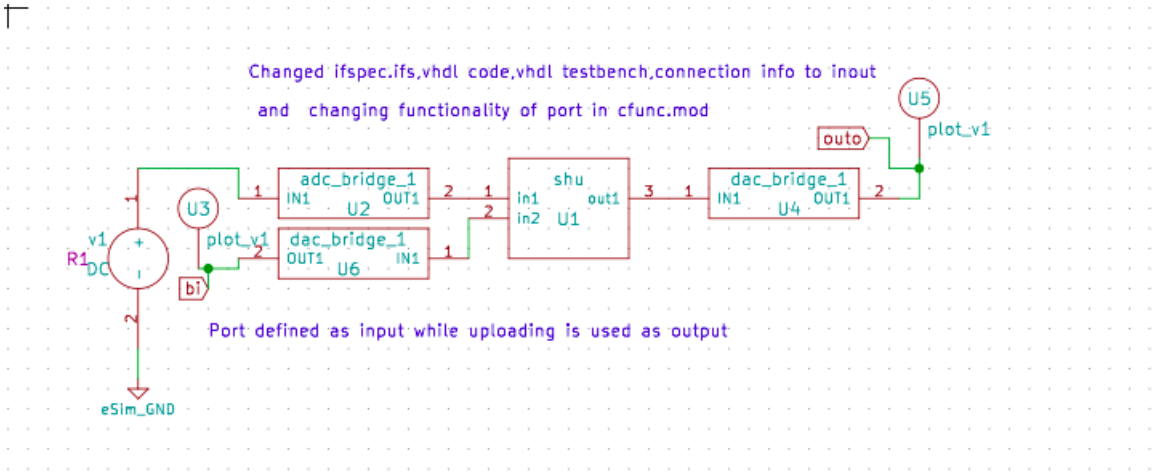

Figure 4.5

Results:

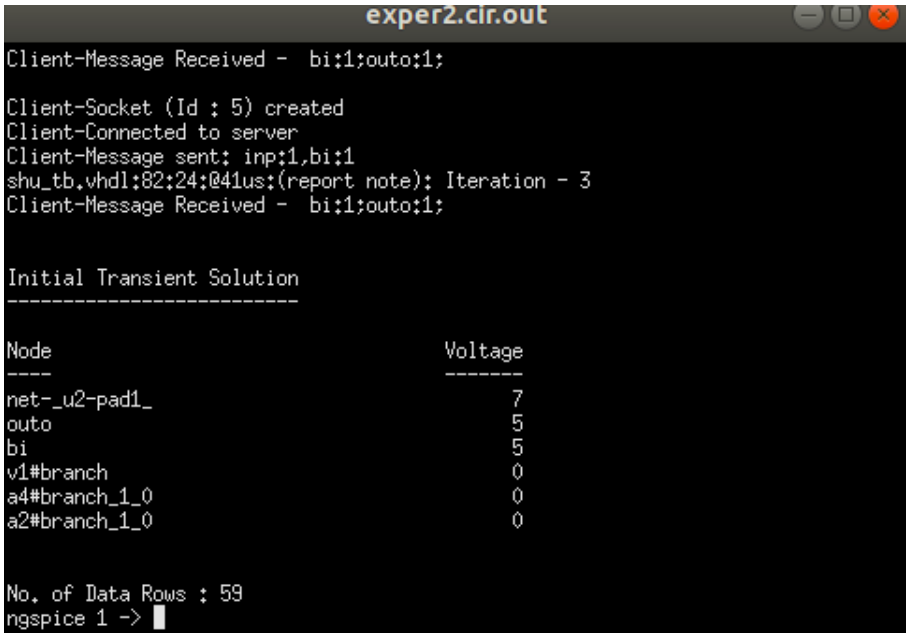

Figure 4.6

4.2 Case 2

Port is declared as OUT and modified as IN in the DUT files.

Steps involved:

1. Upload the vhdl file in NGHDL with tristate logic used for inout port but still defined as OUT in the code entity(let the port name be bidir)

2. Changing the model files for simulation: Similar to case 1,the files which are required to be changed after uploading are: VHDL codefile, VHDL testbench, `Connection_info.txt`, Cfunc.mod, ifspecs.ifs.
   In VHDL code, in entity the port declaration is changed to *inout*.

```
ENTITY inimay IS
    PORT(
        cl   : IN STD_LOGIC;
        oe   : IN STD_LOGIC;
        inp      : IN STD_LOGIC;
        bidir    : INOUT STD_LOGIC;
        outp     : OUT STD_LOGIC);
END inimay;
ARCHITECTURE maxpld OF inimay IS
SIGNAL  a   : STD_LOGIC;   -- DFF that stores
SIGNAL  b   : STD_LOGIC ;  -- DFF that stores
BEGIN                                       -- feedback value.
    PROCESS (oe,a,inp)           -- Behavioral representation
        BEGIN
        a <= inp;                  -- of tri-states.
        IF( oe = '1') THEN
            b <= inp;
            bidir <= a;
        ELSE
            bidir <= 'Z';
            b <= bidir;
        END IF;
    END PROCESS;
        outp <= b;
END maxpld;
```

Figure 4.7

In VHDL testbench, the port declaration is changed to INOUT and also changes are made in the VHPI part where the testbench is linked with ghdlserver.c similar to the case1.

Thus the part `Vhpi_Set_Port_Value` of bidir (inout port) is deleted and `Vhpi_Get_Port_Value` is added to the code as shown below:

```
obj_ref := Pack_String_To_Vhpi_String("oe");
Vhpi_Get_Port_Value(obj_ref,oe_v,1);
oe <= To_Std_Logic(oe_v);

obj_ref := Pack_String_To_Vhpi_String("inp");
Vhpi_Get_Port_Value(obj_ref,inp_v,1);
inp <= To_Std_Logic(inp_v);

obj_ref := Pack_String_To_Vhpi_String("bidir");
Vhpi_Get_Port_Value(obj_ref,bidir_v,1);
bidir <= To_Std_Logic(bidir_v);

wait for 1 us;
outp_v := Pack_String_To_Vhpi_String(To_String(outp));
obj_ref := Pack_String_To_Vhpi_String("outp");
Vhpi_Set_Port_Value(obj_ref,outp_v,1);
```

Figure 4.8

In `Connection_info.txt` file located in model folder ,the bidir pin is defined as IN instead of OUT.

In Ngspice side,in ifspecs.ifs which acts as interface,bidir direction is changed to INOUT and description is also changed as inout pin.

```
PORT_TABLE:
Port_Name:  bidir
Description:    "inout port bidir"
Direction:  inout
Default_Type:   d
Allowed_Types:  [d]
Vector: yes
Vector_Bounds:  [1 1]
Null_Allowed:   no
```

Figure 4.9

In cfunc.mod, bidir is removed as output and bidir is added as new input.Thus, obtaining the value of bidir in Ngspice.

```
char temp_oe[1024];
char temp_inp[1024];
char temp_bidir[1024];



    if(INIT)
    {
        /* Allocate storage for output ports and set the load for input ports */
        cm_event_alloc(0,1*sizeof(Digital_State_t));
cm_event_alloc(1,1*sizeof(Digital_State_t));
/* set the load for input ports. */
int Ii;
for(Ii=0;Ii<PORT_SIZE(cl);Ii++)
{
    LOAD(cl[Ii])=PARAM(input_load);
}
for(Ii=0;Ii<PORT_SIZE(oe);Ii++)
{
    LOAD(oe[Ii])=PARAM(input_load);
}
for(Ii=0;Ii<PORT_SIZE(inp);Ii++)
{
    LOAD(inp[Ii])=PARAM(input_load);
}
for(Ii=0;Ii<PORT_SIZE(bidir);Ii++)
{
```

Figure 4.10

Similarly changes are made throughout the code to ensure bidir functions as IN.

3. After all the changes made in the model file, schematic of new project is designed with using bidir as IN pin as shown below.

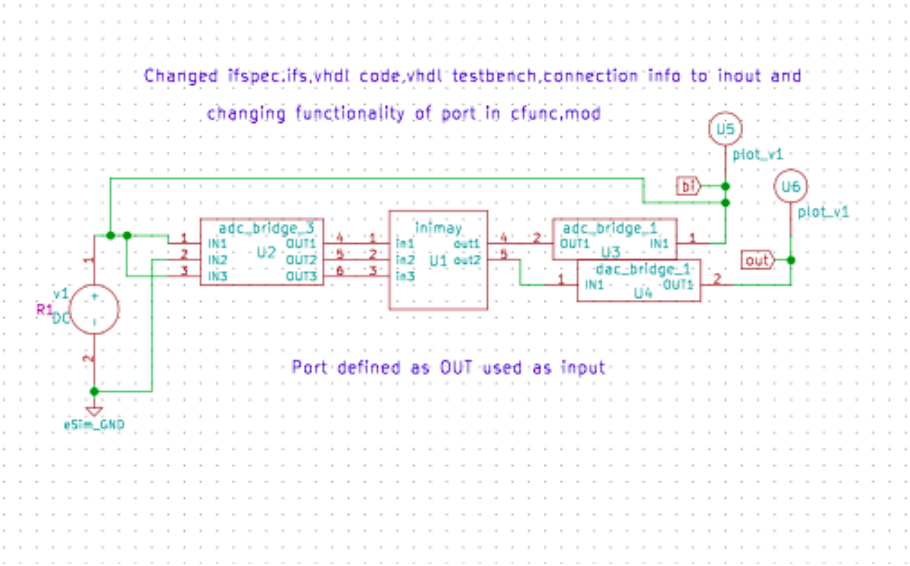

Figure 4.11

With all the steps mentioned above the project was simulated and obtained correct results:

```
Client-Message sent: cl:0,oe:0,inp:0,bidir:0
inimay_tb.vhdl:98:24:@21us:(report note): Iteration - 1
Client-Message Received -  outp:0;

Client-Socket (Id : 5) created
Client-Connected to server
Client-Message sent: cl:1,oe:0,inp:1,bidir:1
inimay_tb.vhdl:98:24:@31us:(report note): Iteration - 2
Client-Message Received -  outp:1;


Initial Transient Solution
--------------------------

Node                                   Voltage
----                                   -------
bi                                           7
out                                          5
v1#branch                                    0
a3#branch_1_0                                0


No. of Data Rows : 59
ngspice 1 ->
```

Figure 4.12

4.3 Case 3

In this case a port should act as both input and output in same simulation.
This case is most challenging due to 2 reasons:

- Schematic drawing-as ADC or DAC are unidirectional and inout port needs bidirectional component to convert analog to digital (while using as IN)and digital to analog signal(while using as OUT).
- Changes to make in files need to support the inout port. INOUT port direction is usually decided by a signal or a input port in the VHDL code.For example when the enable pin is 1, it should act as input and when enable is 0,it should act as output according to this information output should be generated.

Steps followed:

Similar to the above cases, change the VHDL code and ifspec.ifs to INOUT port.

In cfunc.mod,previously a port was either defined for functions of IN or OUT according to the use.In this case,define the pin for both in and out parts.

```
else
{
    _op_bi = (Digital_State_t *) cm_event_get_ptr(0,0);
    _op_bi_old = (Digital_State_t *) cm_event_get_ptr(0,1);
    _op_outo = (Digital_State_t *) cm_event_get_ptr(1,1);
    _op_outo_old = (Digital_State_t *) cm_event_get_ptr(1,2);
}
```

Figure 4.13

```
int Ii;
for(Ii=0;Ii<PORT_SIZE(inp);Ii++)
{
    LOAD(inp[Ii])=PARAM(input_load);
}

for(Ii=0;Ii<PORT_SIZE(bi);Ii++)
{
    LOAD(bi[Ii])=PARAM(input_load);
}
```

Figure 4.14

Similarly, many changes are made throughout the cfunc.mod to make bi function as both IN and OUT.

Next changes are made in `Connection_info.txt` file.The information in `Connection_info.txt` is to tell whether a pin is IN or OUT.This information is used by ghdlserver.c file.When used as both IN and OUT ,the pin can be seperately described as both

IN and OUT or a better solution is to jut define it as OUT. ghdlserver.c file only checks for OUT port in `connection_info.txt` file.Thus describing the pin as just OUT will make it work for both the cases.

Final changes are to be Done in VHDLtestbech file. Just like previous changes defining bi for both IN and OUT using `Vhpi_Get_Port_Value` and `Vhpi_Set_Port_Value` for inout port doesn't give optimal results. When,the set for `Vhpi_Set_Port_Value` of bi are commented the port behaves correctly as IN and by commenting `Vhpi_Get_Port_Value` , the port gives correct results as OUT.But commenting lines everytime while using a port as both IN and OUT in same schematic isn't possible. Thus the following methods were tried to overcome this problem:

IF ELSE logic:

Initially If else loops were tried in the manner shown below:

```
    process
        variable count : integer:=0;
        variable inp_v : VhpiString;
        variable bi_v : VhpiString;
        variable outo_v : VhpiString;
        variable obj_ref : VhpiString;
    begin
        while true loop
            wait until clk_s = '0';

            obj_ref := Pack_String_To_Vhpi_String("inp");
            Vhpi_Get_Port_Value(obj_ref,inp_v,1);
            inp <= To_Std_Logic(inp_v);

            IF( oe = '1') THEN
            obj_ref := Pack_String_To_Vhpi_String("bi");
            Vhpi_Get_Port_Value(obj_ref,bi_v,1);
            bi <= To_Std_Logic(bi_v);
            else
            wait for 1 us;
            bi_v := Pack_String_To_Vhpi_String(To_String(bi));
            obj_ref := Pack_String_To_Vhpi_String("bi");
            Vhpi_Set_Port_Value(obj_ref,bi_v,1);
            end if;
            wait for 1 us;
            outo_v := Pack_String_To_Vhpi_String(To_String(outo));
            obj_ref := Pack_String_To_Vhpi_String("outo");
            Vhpi_Set_Port_Value(obj_ref,outo_v,1);

            report "Iteration - "& integer'image(count) severity note;
            count := count + 1;
        end loop;
    end process;

end architecture;
```

Figure 4.15

Here if else are used to avoid contradiction between IN and OUT and a signal oe is defined in testbench for changing its functionality. But using if else loop gives result for only OUT condition. Also,defining signal called oe isn't a good idea as we cannot change oe value while simulation.There fore we end up by using it as only IN or OUT at a single simulation.

if else failed to achieve target even after trying different ways and changing its position (adding 2 whole loop process in if and else).

After this case ,next was to test the OUT problem by directly using bi as both IN and OUT as shown below:

```
while true loop
    wait until clk_s = '0';

    obj_ref := Pack_String_To_Vhpi_String("inp");
    Vhpi_Get_Port_Value(obj_ref,inp_v,1);
    inp <= To_Std_Logic(inp_v);

    obj_ref := Pack_String_To_Vhpi_String("bi");
    Vhpi_Get_Port_Value(obj_ref,bi_v,1);
    bi <= To_Std_Logic(bi_v);

    report "Iter - "& std_logic'image(bi);
    wait for 1 us;

    report "value - "& std_logic'image(bi);
    bi_v := Pack_String_To_Vhpi_String(To_String(bi));
    obj_ref := Pack_String_To_Vhpi_String("bi");
    Vhpi_Set_Port_Value(obj_ref,bi_v,1);

    outo_v := Pack_String_To_Vhpi_String(To_String(outo));
    obj_ref := Pack_String_To_Vhpi_String("outo");
    Vhpi_Set_Port_Value(obj_ref,outo_v,1);
```

Figure 4.16

Checking error by using it as INOUT simultaneously:

It was observed that even if both `Vhpi_Set_Port_Value` and `Vhpi_Get_Port_Value` are used the when the port is used as input the expected results are obtained.But when used as OUT its always zero.From this we can conclude that the main problem is with making OUT functionality work.

Making OUT port work: (from here all the cases pin is uses as OUT by making several changes in testbench)
It is clear by above statements that when used as OUT the data is getting lost and resulting in zero always. So,the next step was to detect where the data is getting lost. By checking files it was found that error is in VHDL file itself. VHDL code return X undefined value,thus resulting in 0 output.
'X' in simulation can mean drive contention. i.e. your test bench is driving a '0', and at the same time your unit under test is driving a '1'. To work correctly, a bidirectional pin should be assigned to 'Z' by the entity that is not currently controlling the wire. This means that when the UUT(unit under test: vhdl code) wants to use the pin for output, the test bench should assign 'Z' to the pin. When the test bench wants to drive the pin as an input to the UUT, then the UUT needs to assign 'Z' to the pin.
In our method,we assign port value Z when used as input in VHDL code (refer Figure 4.1) but in VHDL testbench while using as OUT and we arent assigning Z

to port value. Thus Z value is assigned to bi as shown below:

```
        wait until clk_s = '0';
        obj_ref := Pack_String_To_Vhpi_String("inp");
        Vhpi_Get_Port_Value(obj_ref,inp_v,1);
        inp <= To_Std_Logic(inp_v);

        obj_ref := Pack_String_To_Vhpi_String("bi");
        Vhpi_Get_Port_Value(obj_ref,bi_v,1);
        bi <=  'Z';

        wait for 1 us;

        bi_v := Pack_String_To_Vhpi_String(To_String(bi));
        obj_ref := Pack_String_To_Vhpi_String("bi");
        Vhpi_Set_Port_Value(obj_ref,bi_v,1);

        outo_v := Pack_String_To_Vhpi_String(To_String(outo));
        obj_ref := Pack_String_To_Vhpi_String("outo");
        Vhpi_Set_Port_Value(obj_ref,outo_v,1);
        report "test3 - "& std_logic'image(bi);
```

Figure 4.17

By using it as above case,the OUT functionality executes with right results. But it code doesn't work when we want to use it as input.Thus this idea is resumed and bi $\leq$ z is given after the wait statement.But it isnt of no use because the VHDL code executes it logic before that wait statement and returns out values later.

It was also observed later that `Vhpi_Get_Port_Value` for bi(bidirectional) isnt the main problem but bi $\leq$ `to_std_logic(bi_v)` is creating zero output. Only by commenting this line output shows right results.When bi $\leq$ `to_std_logic(bi_v)` isn't commented, bi gets zero value from cfunc.mod when its used as output as no input is given to this. When bi got value 0 from ngspice(schematic and cfunc.mod) it doesn't work as output and produces zero result.(In VHDL codes it actually gives X as output which in ngspice is shown as zero).
But by observations and many trials its found that even if 0 creates the problem after giving value 1 as input to bi in testbench gives correct results as OUT(checked for both the cases were results should be zero and one). When 1 is given to bi,initially VHDL value of X for output but by the end of 3 iterations it returns the desired value thus giving plots with optimal results.

```
wait until clk_s = '0';
obj_ref := Pack_String_To_Vhpi_String("inp");
Vhpi_Get_Port_Value(obj_ref,inp_v,1);
inp <= To_Std_Logic(inp_v);

obj_ref := Pack_String_To_Vhpi_String("bi");
Vhpi_Get_Port_Value(obj_ref,bi_v,1);
bi <=  '1';

wait for 1 us;

bi_v := Pack_String_To_Vhpi_String(To_String(bi));
obj_ref := Pack_String_To_Vhpi_String("bi");
Vhpi_Set_Port_Value(obj_ref,bi_v,1);
```

Figure 4.18: Assigning 1 as input

OR Logic with enable signal:

From above observations we can conclude that for a port to work as input it should take value given by server `(using To_STD_Logic(bi_v))` and to work as out put bi should be taking value one. Thus the following logic was used :

```
wait until clk_s = '0';
obj_ref := Pack_String_To_Vhpi_String("inp");
Vhpi_Get_Port_Value(obj_ref,inp_v,1);
inp <= To_Std_Logic(inp_v);

obj_ref := Pack_String_To_Vhpi_String("bi");
Vhpi_Get_Port_Value(obj_ref,bi_v,1);
bi <=  oe or To_Std_Logic(bi_v);

wait for 1 us;

bi_v := Pack_String_To_Vhpi_String(To_String(bi));
obj_ref := Pack_String_To_Vhpi_String("bi");
Vhpi_Set_Port_Value(obj_ref,bi_v,1);

outo_v := Pack_String_To_Vhpi_String(To_String(outo));
obj_ref := Pack_String_To_Vhpi_String("outo");
Vhpi_Set_Port_Value(obj_ref,outo_v,1);
```

Figure 4.19

With oe being the enable pin this method gives correct results for both IN and OUT. oe value is 0 when used as IN thus logical or assigns the value of input given by server to bi. Oe value is 1 when used as OUT thus the logic in VHDL code executes with out X and returns value to client (cfunc.mod) through server by `Vhpi_Set_Port_Value`.

Change of INOUT functionality by using a input pins:

Every time we need to change behaviour of INOUT pin in the above method mentioned oe value has to be changed while simulation which isnt possible. Also,usually inout pins direction is decided usually by one or more input pins(for example the condition can be when 2 inputs namely in1 and in2 are 1 then the pin is IN or else the pin is OUT). For VHDL codes where input pins decide direction of port,the testbench is edited as follows:

```
        Vhpi_Listen;
        wait for 2 us;
        Vhpi_Send;
    end loop;
    wait;
end process;

process
    variable count : integer:=0;
    variable inp_v : VhpiString;
    variable bi_v : VhpiString;
    variable outo_v : VhpiString;
    variable obj_ref : VhpiString;
begin
    while true loop
        wait until clk_s = '0';
        obj_ref := Pack_String_To_Vhpi_String("inp");
        Vhpi_Get_Port_Value(obj_ref,inp_v,1);
        inp <= To_Std_Logic(inp_v);

        obj_ref := Pack_String_To_Vhpi_String("bi");
        Vhpi_Get_Port_Value(obj_ref,bi_v,1);

        wait for 1 us;
        bi <=  inp or To_Std_Logic(bi_v);
        wait for 1 us;

        bi_v := Pack_String_To_Vhpi_String(To_String(bi));
        obj_ref := Pack_String_To_Vhpi_String("bi");
        Vhpi_Set_Port_Value(obj_ref,bi_v,1);

        outo_v := Pack_String_To_Vhpi_String(To_String(outo));
        obj_ref := Pack_String_To_Vhpi_String("outo");
        Vhpi_Set_Port_Value(obj_ref,outo_v,1);
--      report "test3 - "& std_logic'image(bi);
        report "Iteration - "& integer'image(count) severity note;
```

Figure 4.20

The 2 main changes that you can observe in the above method are:

- In 1st process where `Vhpi_send` and `Vhpi_listen` are used the wait time is increased to 2 us.
- In second process and additional wait 1 us step is added and assigning value to bidirectional port is done inside that.

Here inp is the input signal which decides the direction of bidirectional pin. Thus logical or operation is done with inp pin. But the value of inp is only identified in testbench after the wait statement. So,first bi¡= inp or `to_std_logic(bi_v)` line is given after wait statement. But this gives wrong results as VHDL code logic execution takes place before the wait statement itself. After wait statement only giving the output values obtained in VHDL code to client through server takes place. Thus an additional wait statement is added specially for inout pin. If we divide the functioning of code into 3 parts it can be explained as below:

- Before first wait statement the inputs are taken and given to digital design in VHDL code.
- After first wait statement the inout port input is taken based of input pin value.
- After bidirectional pin value is also assigned the digital logical in VHDL code gets executed and are given back to client through `Vhpi_Set_Port_Value` after 2nd wait statement.

Chapter 5

Conclusion

The implementation of INOUT port was done along with the following restrain:
After uploading the VHDL code the user has to mention the condition on which the behaviour of code depends in VHDL tesbench where bidirectional port is assigned value as follows:
bi $\leq$ (enable pin/condition) or `to_std_logic(bi_v)`
Once if the enable pin or condition is mentioned in the testbench then the project is good to go. We need not make any changes again before simulation or when changing its behaviour from IN to OUT and vice versa.

Bibliography

[1] Ngspice manual
URL:http://ngspice.sourceforge.net/docs/ngspice-31-manual.pdf

[2] Github FOSSEE NGHDL Repository
URL:https://github.com/fossee/nghdl

[3] Intel forums
URL:https://forums.intel.com/s/question/0D50P00003yyLhLSAU/problems-with-bidirectional-signals-in-vhdl-testbench-for-modelsim?language=en_US

[4] Xilinx forums on INOUT testbench
URL:https://forums.xilinx.com/t5/Synthesis/INOUT-port-in-VHDL/td-p/429116

[5] Stackoverflow website
URL:https://electronics.stackexchange.com/questions/458144/vhdl-what-will-happen-in-case-of-u-or-x-or-z-signal

[6] Stack exchange on testbench for INOUT VHDL
URL:https://electronics.stackexchange.com/questions/309960/testbench-for-inout-port-in-vhdl

[7] eSim official website.
URL:https://esim.fossee.in/

[8] Intel website
URL:https://www.intel.com/content/www/us/en/programmable/support/support-resources/design-examples/design-software/vhdl/v_bidir.html