

Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Verilog Simulation Demo

(Refer Slide Time: 00:24)

How to compile/run/simulate

- Commercial tools
 - Mentor's Modelsim/Quarta
 - Cadence Incisive
 - Synopsys – VCS
 - Xilinx Vivado
- Free
 - icarus (both Verilog and VHDL)
 - GHDL (only for VHDL)
- Online platform
 - edaplayground

Sometime we would also like to see here, the next question would be that how will we compile, how will we run or how we will simulate, so what are these three different things? What does each of them mean? Compile means because it is a high level language, so we need to compile it we need a compiler and after compilation the execution will be formed, executable would be formed and that executable once we will run that run is also called simulation because we are simulating the hardware behavior.

So how do we compile, run and simulate the hardware model? Now because the language is specific, the software which would compile, run and simulate are different basically more specific to those languages. Now there are multiple of these commercial tools which are usually there are couple of EDA companies electronic design automation companies, they build, they design the software and we can use these software to compile, run and simulate.

So, these big three or four EDA companies Mentor's Graphics has a ModelC which is quite popular Verilog compiler and Verilog simulation tool and Cadence has Incisive, Synopsys has VCS and Xilinx has Vivado. So, we would be using Xilinx Vivado at some stage of time whenever you will come back in the campus, because Vivado not only would be able to compile and simulate, but it would also be able to synthesize.

So after using synthesis you can burn that design onto FPGA ports. So once you are back so then we can have this FPGA ports experiments. The other these commercial tools will have many more features like it help you quite a lot in debugging and visualization. On the other hand, there are free software which can be used for compilation and simulation. So, in case of Verilog Icarus is a popular one which can be used and for GHDL is another one which is only for VHDL. There are some online platforms also like edaplayground. So edaplayground help you to write this Verilog code on to a online browser where you can compile, you can simulate all the online.

(Refer Slide Time: 03:18)

Viewing waveforms

- Free HDL simulator does not offer GUI and waveform view
- Waveform need to explicitly dumped, see in waveform viewer

```
initial
  begin
    $dumpfile("ha.vcd");
    $dumpvars;
  end
```

This half-adder source code is available at:
<https://www.edaplayground.com/x/3Vua#>

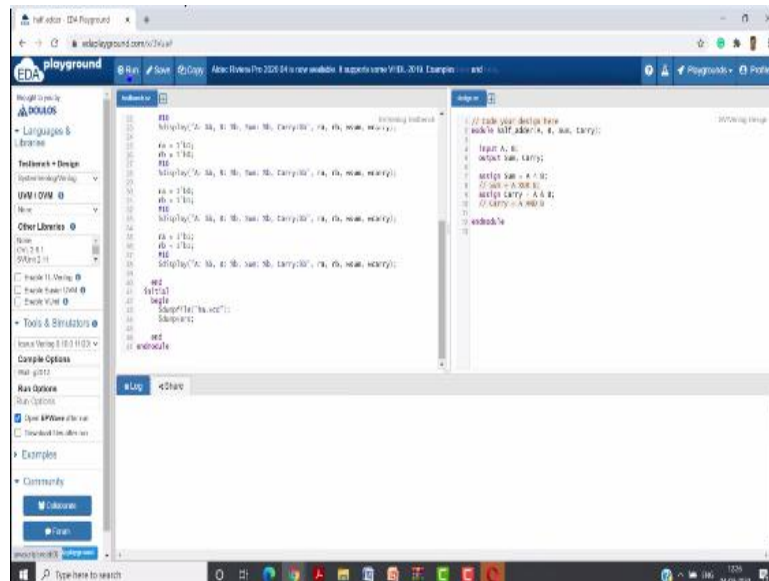


So, now during this course we will be using free HDL simulators Icarus most of the time. So this Icarus software you can install on to your computers I will give a quick demo on to using my computer. So, because we would be compiling it using a Icarus software it is a command line tool. Now to view the waveform, we have to explicitly dump these waveforms and then see in a waveform viewer. So, to see or to add to dump the waveforms we have to add additional initial module in our testbench in that initial module we will write initial begin dump file and then the name of the file and then one particular command is sufficient dumpvars.

So, this dumpvars whatever variable it finds in your whole module testbench it will dump all of them in that vcd file, vcd is value code dump. So this would be a dump file which can be used to see our waveforms. So, now let us quickly look at the demonstration so I will give you two demonstration one based on edaplayground and the second one based on your own

installation of Icarus. So, let us say this code I am directly using the code which I have written and let us see that in the browser.

(Refer Slide Time: 04:58)

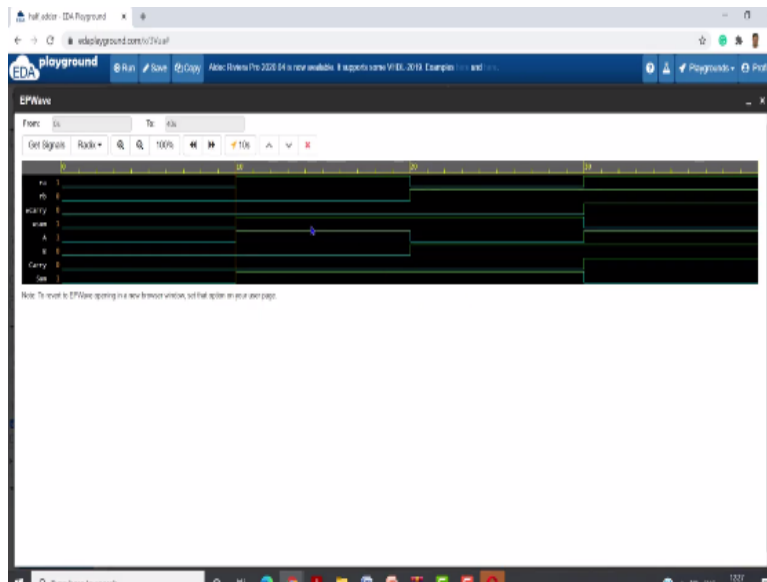


So, let us have a quick understanding of this edaplayground. You can see on one side you are writing your design. Design is being saved as design dot sv because system Verilog sv is the extension for system Verilog because system Verilog is a superset of Verilog so that is a by default edaplayground take it like a dot sv extension and then we have written the same code here and this code similarly testbench dot sv is also written.

So this is also the similar thing what we have written you will see two initial modules one initial modules have all the stimulus part, the other stimulus, the other initial module dumps the variables. So on the left page you can see that whether your design is of Verilog system Verilog or VHDL so because we are using Verilog and system Verilog. So there is no UVM, OVM and there is no other libraries which are used.

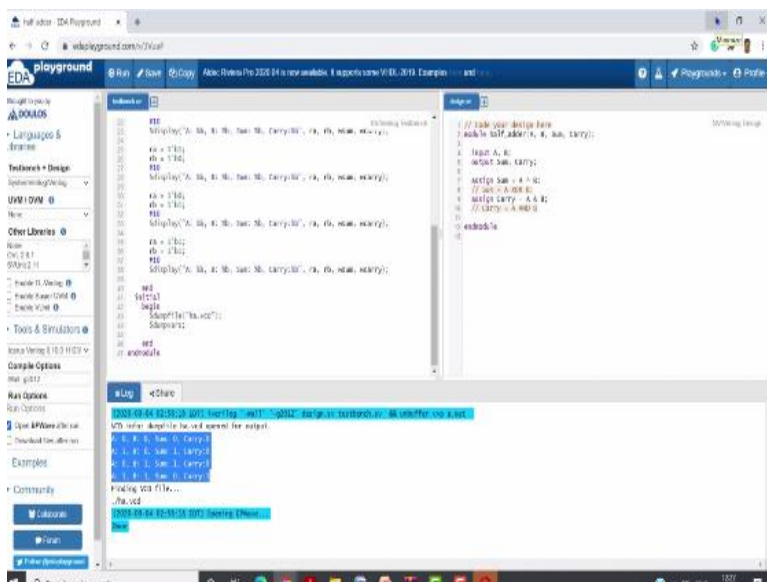
So, here in tools and simulators you can see there are various commercial software and then also free simulator. So in free simulator we would be using the latest version of Icarus and we cannot use this commercial software because all of them are paid and requires some license fees. So, let us use the latest version of Icarus as the software and then after doing all of these thing then we can also say that after running we would like to open in EP wave so that we can see the waveforms. So after all of these things are done we can save it and we can run it.

(Refer Slide Time: 07:00)



So, after running you will see two things. One, that you will see a waveform viewer. In this waveform viewer you have four signals ra rb w carry w sum ab carry and sum and you can see the input and output of all of them at point, you see that exactly at 10 nanosecond when we have changed our input, our output got changed immediately. So this is how what is the meaning of our assigned statement that it changed our output immediately. So, similarly at any particular point of time now we can check what is the input and percent corresponding to that input what is the output. So now we can close this.

(Refer Slide Time: 07:53)



And if we close this then we see that in the low or in the test, in the prompt you see that these variables have been displayed or basically this print statements have been given A equal to 0 B equal to 0 sum equal to 0 comma carry equal to 0. So, all of these things are written in a same way the way format we have defined. So this is how we can use edaplayground, so you can sign up in this edaplayground and you can use. The other alternative you can install this Icarus on to your own computer and then can work.

(Refer Slide Time: 08:34)

```

[neera@localhost icarus]$ git
git clone git://github.com/steveicarus/iverilog.git
Cloning into 'iverilog'...
remote: Enumerating objects: 143, done.
remote: Counting objects: 100% (143/143), done.
remote: Compressing objects: 100% (100/100), done.
remote: Total 59107 (delta 80), reused 79 (delta 43), pack-reused 58964
Receiving objects: 100% (59107/59107), 23.67 MiB | 3.08 MiB/s, done.
Resolving deltas: 100% (47351/47351), done.
[neera@localhost icarus]$ ls
iverilog
[neera@localhost icarus]$ cd iverilog/
[neera@localhost iverilog]$ ls

```

```

aLocal.m4      elab_expr.cc      ivl_target_priv.h  netlist.cc       PDelays.cc        PScope.h         t-dll.txt
AStatement.cc elab_lval.cc      ivl_target.txt    netlist.h        PDelays.h         PSpec.cc         tgt-blif
AStatement.h  elab_net.cc      lexor_keyword.gperf netlist.txt      PEvent.cc        PSpec.h         tgt-fpga
asyncc.cc     elaborate_analog.cc lexor_keyword.h  netmisc.cc       PEvent.h         PTask.cc        tgt-null
Attrib.cc     elaborate_analog.cc lexor_lex         netmisc.h        PExpr.cc         PTask.h         tgt-pal
Attrib.h      elab_scope.cc    libisc           net_modulo.cc    PExpr.h          PUDP.cc        tgt-pcb
attributes.txt elab_sig_analog.cc libveriuser     net_nex_input.cc pform_analog.cc  PUDP.h         tgt-sizer
autoconf.sh  elab_sig.cc      link_const.cc    net_nex_output.cc pform.cc         PWire.cc        tgt-stub
BUS5.txt     elab_type.cc     load_module.cc   netparray.cc     pform_class_type.cc PWire.h         tgt-verilog
cadpli       eml.cc           lpm.txt          netparray.h      pform_disciplines.cc Statement.h       tgt-vhdl
check_conf   eval_attrib.cc   main.cc          net_proc.cc      pform_dump.cc    README.txt      tgt-vlog95
compiler.h   eval.cc          Makefile.in      netqueue.cc     pform.h          scripts         tgt-vvp
config_guess eval_tree.cc     mingw-cross.txt  netqueue.h      pform_package.cc solaris         util.h
config.h.in  examples        mingw.txt        netscalar.cc    pform_pclass.cc  Statement.cc    va_math.txt
config.sub   exposeodes.cc   minstalldirs    netscope.h      pform_string_type.cc Statement.h       verilog.spec
configure.in expr_synt.cc    Module.cc        net_scope.cc    pform_struct_type.cc svector.h       verinum.cc
constants.vams functor.cc       Module.h         netstruct.cc    pform_types.cc  sv_vpi_user.h  verinum.h
COPYING     functor.h       net_analog.cc   netstruct.h     pform_types.h   swift.txt      verilreal.cc
COPYING.lesser functor.h        net_assign.cc   net_tran.cc     PFunction.cc    symbol_search.cc verilreal.h
cpcheck_sup glossary.txt    net_class.cc    nettypes.cc     PGate.cc        sync.cc        veriuser.h
cprop.cc    HName.cc       netclass.h      nettypes.h      PGate.h         syn_rules.y    version.base.h
cygwin.txt  HName.h        netclass.h      net_udp.cc      PGenerate.cc    synth2.cc     version.c
design_dump.cc ieee1364-notes.txt netdarray.cc   netvector.cc    PGenerate.h     synth.cc       vhd1pp
developer-quick-start.txt INSTALL        netdarray.h    netvector.h     p_ll_types.h.in sys_funcs.cc  vpi
discipline.cc install-sh      net_design.cc  ndangle.cc     PModport.cc    target.cc     vpi_modules.cc
discipline.h  iverilog-vpi.man.in netenum.cc     parse_api.h     PModport.h     target.h      vpi.txt
disciplines.vams iverilog-vpi.sh netenum.h      parse_api.h     PNamedItem.cc  t-dll-analog.cc vpi_user.h
dosify.c     ivl_alloc.h    net_event.cc   parse_misc.cc   PNamedItem.h   t-dll-api.cc  vvp
driver       ivl_assert.h  net_expr.cc    parse_misc.h    PPackage.cc    t-dll.cc     xilinx-hint.txt
dup_expr.cc  ivl_def       net_func.cc    parse.y         PPackage.h     t-dll-expr.cc
ivlpp       net_func_eval.cc net_func_eval.cc PClass.cc      property_qual.h t-dll.h

```

```
Precompiling vhdpp/lexor_keyword.gperf
[neera]@localhost iverilog$ ls
acc_user.h          dup_expr.cc        ivl_target.h      net_link.cc       PDelays.cc        PSpec.cc          tgt_fpga
aclocal.m4          elab_anet.cc      ivl_target_priv.h net_list.cc       PDelays.h         PSpec.h           tgt-null
AStatement.h       elab_expr.cc      ivl_target.txt    netlist.cc       PEvent.cc         PTask.cc          tgt-pal
AStatement.h       elab_lval.cc      lexor_keyword.cc  netlist.txt      PEvent.h         PTask.h           tgt-pcb
asyncc.cc          elab_net.cc       lexor_keyword.gperf netmisc.cc       PExpr.cc          PUDP.cc           tgt-sizer
Attrib.cc          elaborate_analog.cc lexor_keyword.h   netmisc.h        PExpr.h          PUDP.h            tgt-stub
Attrib.h           elab_scope.cc     lexor.lex         net_modulo.cc    pform_analog.cc  PWire.cc          tgt-verilog
attributes.txt      elab_sig_analog.cc libveruser        net_nex_input.cc pform_class_type.cc PWire.h           tgt-vhdl
autoconf.sh        elab_sig.cc       link_const.cc     net_nex_output.cc pform_class_type.cc QUICK_START.txt   tgt-vlog95
autom4te.cache     elab_sig.cc       load_module.cc    netnarray.cc     pform_disciplines.cc README.txt         tgt-vvp
BUGS.txt           elab_type.cc      lpm.txt           netnarray.h      pform_dump.cc    scripts            util.h
cadpli             emit.cc           main.cc           net_proc.cc      pform.h           solaris            va_math.txt
check_conf         eval_attrib.cc    makefile.in       net_queue.cc     pform_package.cc Statement.cc       verilog.spec
compiler.h         eval.cc           mingw-cross.txt   netqueue.h       pform_pclass.cc  Statement.h        verilum.cc
config_guess       eval_tree.cc      mingw-cross.txt   netscalar.cc     pform_string_type.cc svector.h         verilum.h
config.h.in        examples          mkinstalldirs    net_scope.cc     pform_struct_type.cc sv_vpi_user.h     verilreal.cc
config.sub         exposenodes.cc   Module.cc         net_struct.cc    pform_types.h    swift.txt         verilreal.h
configure.in        expr_synth.cc    Module.h          netstruct.h      pform_types.h    symbol_search.cc  veruser.h
constants.vams     functor.cc        named.h           net_tran.cc      PGate.cc         sync.cc           version_base.h
COPYING            functor.h         net_analog.cc    netypes.cc       PGate.h          synth2.cc         vhdpp
COPYING.lesser    glossary.txt     net_assign.cc    netypes.h        PGenerate.cc     sys_funcs.cc     vpi
cppcheck.sup       HName.cc         netclass.cc      net_udp.cc       PGenerate.h      sys_funcs.h       vpi_modules.cc
cprop.cc           HName.h          netclass.h       netverctor.cc   p_ll_types.h.in  target.cc         vpi.txt
kypwin.txt         ieee1364-notes.txt netarray.cc       netvector.h      PModport.cc     t-dll-analog.cc  vpi_user.h
design_dump.cc      INSTALL          net_design.cc    netarray.h       nodangle.cc      PModport.h       vvp
developer-quick-start.txt install-sh       iverilog-vpi.man.in netenum.cc       pad_to_width.cc PNamedItem.cc    t-dll-api.cc    xilinx-hint.txt
discipline.cc      iverilog-vpi.sh netenum.h        netenum.cc       parse_api.h     PNamedItem.h     t-dll.cc
discipline.h       ivl_alloc.h      netevent.cc     netenum.h        parse_misc.cc   PPackage.cc      t-dll-expr.cc  t-dll.h
disciplines.vams  ivl_assert.h     net_expr.cc     netevent.h       parse.y          PPackage.h       t-dll.h
dosify.c           ivl_def          net_func.cc     net_expr.h       property_qual.h PPropertyQual.h  t-dll-proc.cc  t-dll.txt
driver             ivl_def          net_func.cc     net_func.h       PClass.h        PScope.h         t-dll.txt
driver-vpi         ivlpp            net_func_eval.cc net_func.h       PClass.h        PScope.h         t-dll.txt
[neera]@localhost iverilog$ ./config
```

```
attributes.txt      elaborate_analog.cc lexor.lex         net_modulo.cc    pform.cc          PWire.h           tgt-verilog
autoconf.sh        elaborate.cc       libveruser       net_nex_input.cc pform_class_type.cc QUICK_START.txt   tgt-vhdl
autom4te.cache     elab_scope.cc     link_const.cc    net_nex_output.cc pform_class_type.cc README.txt         tgt-vlog95
BUGS.txt           elab_sig_analog.cc libveruser        netnarray.cc     pform_dump.cc    scripts            tgt-vvp
cadpli             elab_sig.cc       load_module.cc   netnarray.h      pform.h           solaris            util.h
check_conf         elab_type.cc      lpm.txt          net_proc.cc      pform_package.cc stamp-config-h   va_math.txt
compiler.h         emit.cc           main.cc          netqueue.cc     pform_pclass.cc  stamp_pli_types-h verilog.spec
config_guess       eval_attrib.cc    Makefile.in      netqueue.h       pform_string_type.cc Statement.cc       verilum.cc
config.h.in        eval.cc           mingw-cross.txt  netscalar.cc     pform_struct_type.cc Statement.h        verilum.h
config.log         examples          mkinstalldirs   net_scope.cc     pform_types.h    svector.h         verilreal.cc
config.status      exposenodes.cc   Module.cc        net_struct.cc    PFunction.cc     swift.txt         verilreal.h
config.sub         expr_synth.cc    Module.h         netstruct.h      PFunction.cc     symbol_search.cc  version_base.h
configure.in        functor.cc        named.h          net_tran.cc      PGate.h          sync.cc           version.c
constants.vams     functor.h         net_analog.cc   netypes.h        PGenerate.cc     synth2.cc         vhdpp
COPYING            glossary.txt     net_assign.cc   netypes.h        PGenerate.h      sys_funcs.cc     vpi_modules.cc
COPYING.lesser    HName.cc         netclass.cc     net_udp.cc       PGenerate.h      sys_funcs.h       vpi.txt
cppcheck.sup       HName.h          netclass.h       netvector.cc    p_ll_types.h.in  sys_funcs.cc     vpi_user.h
cprop.cc           HName.h          netclass.h       netvector.h      PModport.cc     target.cc         vvp
kypwin.txt         ieee1364-notes.txt netarray.cc       netvector.h     nodangle.cc      PModport.h       t-dll-analog.cc
design_dump.cc      INSTALL          net_design.cc    netarray.h       pad_to_width.cc PNamedItem.cc    t-dll-api.cc    xilinx-hint.txt
developer-quick-start.txt install-sh       iverilog-vpi.man.in netenum.cc       parse_api.h     PNamedItem.h     t-dll.cc
discipline.cc      iverilog-vpi.sh netenum.h        netenum.cc       parse_misc.cc   PPackage.cc      t-dll-expr.cc  t-dll.h
discipline.h       ivl_alloc.h      netevent.cc     netevent.h       parse.y          PPackage.h       t-dll.h
disciplines.vams  ivl_assert.h     net_expr.cc     net_expr.h       property_qual.h PPropertyQual.h  t-dll-proc.cc  t-dll.txt
dosify.c           ivl_def          net_func.cc     net_func.h       PClass.h        PScope.h         t-dll.txt
[neera]@localhost iverilog$ make
mkdir dep
Using git-describe for VERSION_TAG
g++ -DHAVE_CONFIG_H -I. -Ilibmisc -Wall -Wextra -Wshadow -g -O2 -MD -c main.cc -o main.o
mv main.d dep/main.d
g++ -DHAVE_CONFIG_H -I. -Ilibmisc -Wall -Wextra -Wshadow -g -O2 -MD -c asyncc.cc -o asyncc.o
mv asyncc.d dep/asyncc.d
g++ -DHAVE_CONFIG_H -I. -Ilibmisc -Wall -Wextra -Wshadow -g -O2 -MD -c design_dump.cc -o design_dump.o
```

```
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-null'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/tgt-stub'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-stub'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/tgt-vvp'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-vvp'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/tgt-vhdl'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-vhdl'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/tgt-vlog95'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-vlog95'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/tgt-pcb'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-pcb'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/tgt-blif'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-blif'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/tgt-sizer'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/tgt-sizer'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/driver'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/home/neera/labs/icarus/iverilog/driver'
make[1]: Entering directory '/home/neera/labs/icarus/iverilog/driver'
./mkinstalldirs "/usr/local/bin" \
"/usr/local/include/iverilog" \
"/usr/local/lib/ivl" \
"/usr/local/lib/ivl/include" \
"/usr/local/share/man" \
"/usr/local/share/man/man1"
/usr/bin/install -c -m 644 iverilog-vpi.man "/usr/local/share/man/man1/iverilog-vpi.1"
/usr/bin/install: cannot remove '/usr/local/share/man/man1/iverilog-vpi.1': Permission denied
make: *** [install.man] Error 1
[neera]@localhost iverilog$ sudo make install
```

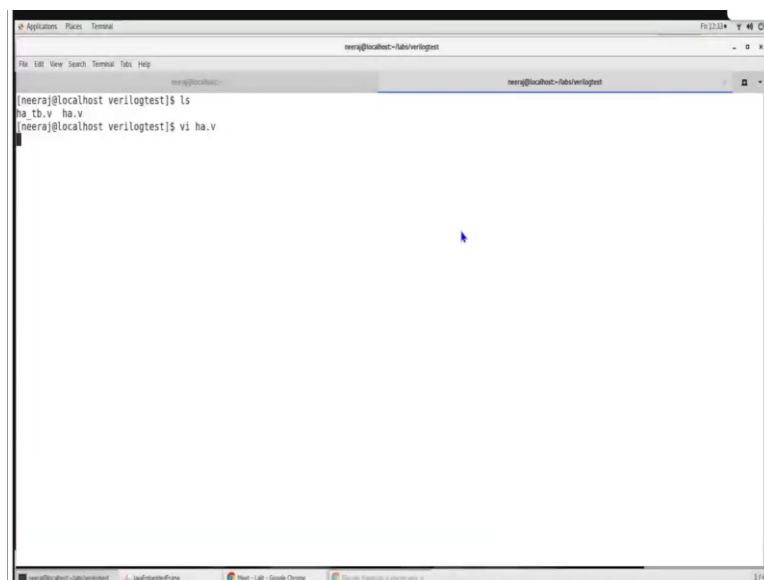
So let us see that part of demo as well. So, I have created one directory here and then I will start with the installation part. So, during the installation first thing I have to do is I have to get it from the GitHub. So the statement command is `gitclone github isteve icarus iverilog dot git`. So, after getting this git then I see that one directory has been created iverilog and in this directory there are so many files.

We have to see how do we start configuring. There is one particular sh file which is called `autoconf dot sh`. So, first we will run this `autoconf dot sh` and after this we see that there is another file which has been created which is `configure`. Now we will run this `configure` file. So after running this `configure` a `make` file would be generated. So we see again now a `make` file has been generated.

So this `make` file we can use to make or to compile this whole software. It will take couple of seconds. Yes, we see now it is done and now we can install this iverilog into our computer. So, for installing we can write `make install` so when we say `make install` it says either because permission is denied. So, there are two alternative for this, either if we use, if we have our route permission on to our system then we do `sudo make install`.

Otherwise we define the directory where it need to be defined, it need to be installed and that directory should have permissions. So, after installing now we can see that this has been installed and now we can start using it. So, if for using it I have again downloaded the same file which I have created the same file.

(Refer Slide Time: 11:38)




```
1 module half_adder(A, B, Sum, Carry);
2
3   input A, B;
4   output Sum, Carry;
5
6   // Logic
7   // Sum = A XOR B;
8   // Carry = A AND B;
9   assign Sum = A ^ B;
10  assign Carry = A & B;
11
12 endmodule
```

ha.v 11L, 178C 6.3 All

So you see ha dot v is the half adder module which defines the of data flow module of our half adder.

(Refer Slide Time: 11:51)

```
[neera@localhost verilogtest]$ ls
ha_tb.v ha.v
[neera@localhost verilogtest]$ vi ha.v
[neera@localhost verilogtest]$ vi ha
```

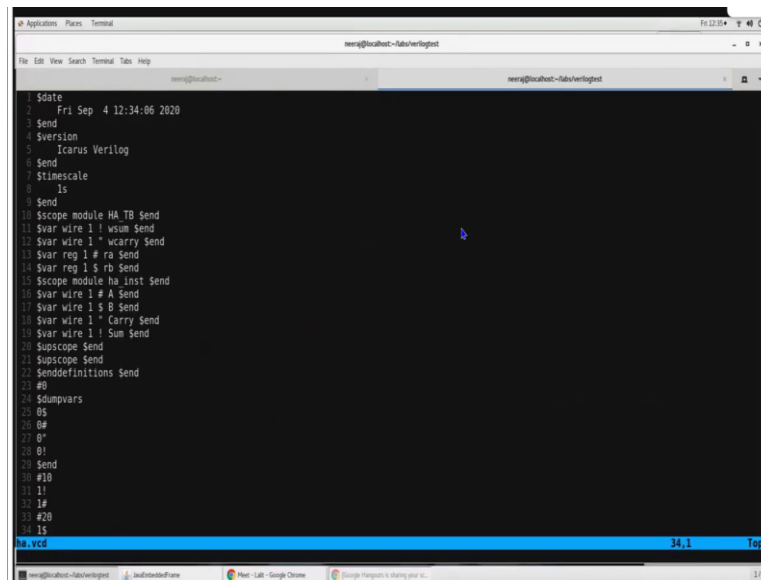


```
11 B(rb),
12 .Sum(wsum),
13 .Carry(wcarry));
14
15 initial
16 begin
17     ra = 1'b0;
18     rb = 1'b0;
19     #10
20     $display($time, " A: %b, B: %b, Sum: %b, Carry:%b", ra, rb, wsum, wcarry);
21
22     ra = 1'b1;
23     rb = 1'b0;
24     #10
25     $display($time, " A: %b, B: %b, Sum: %b, Carry:%b", ra, rb, wsum, wcarry);
26
27     ra = 1'b0;
28     rb = 1'b1;
29     #10
30     $display($time, " A: %b, B: %b, Sum: %b, Carry:%b", ra, rb, wsum, wcarry);
31
32     ra = 1'b1;
33     rb = 1'b1;
34     #10
35     $display($time, " A: %b, B: %b, Sum: %b, Carry:%b", ra, rb, wsum, wcarry);
36
37 end
38 initial
39 begin
40     $dumpfile("ha.vcd");
41     $dumpvars;
42
43 end
44 endmodule
ha_tb.v 15,1 Bot
ha_tb.v' 44L, 789C
```

And we have end ha dot testbench. We have the testbench which provides the stimulus, which instantiate the half adder and it also dumps our vcd files.

(Refer Slide Time: 12:04)

```
[neera@localhost verilogtest]$ ls
ha_tb.v ha.v
[neera@localhost verilogtest]$ vi ha.v
[neera@localhost verilogtest]$ iverilog ha.v ha_tb.v
[neera@localhost verilogtest]$ ls
a.out ha_tb.v ha.v
[neera@localhost verilogtest]$ ./a.out
VCD info: dumpfile ha.vcd opened for output.
10 A: 0, B: 0, Sum: 0, Carry:0
20 A: 1, B: 0, Sum: 1, Carry:0
30 A: 0, B: 1, Sum: 1, Carry:0
40 A: 1, B: 1, Sum: 0, Carry:1
[neera@localhost verilogtest]$ ls
a.out ha_tb.v ha.v ha.vcd
[neera@localhost verilogtest]$ vi
```



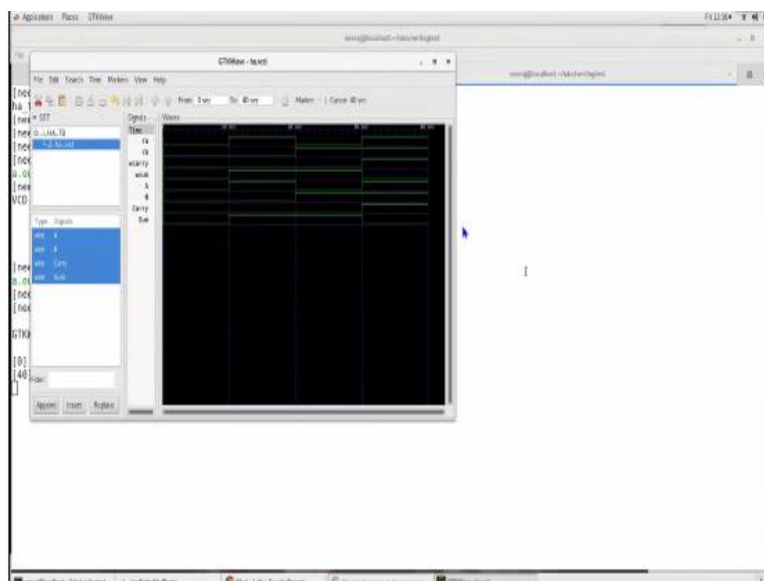
```
neeraj@localhost:~/Icarus/verilog$
$date
Fri Sep 4 12:34:06 2020
$send
$version
Icarus Verilog
$send
$timescale
1s
$send
$scope module HA_TB $send
$var wire 1 ! wsum $send
$var wire 1 * wcarry $send
$var reg 1 # ra $send
$var reg 1 $ rb $send
$scope module ha_inst $send
$var wire 1 # A $send
$var wire 1 $ B $send
$var wire 1 * Carry $send
$var wire 1 ! Sum $send
$upscope $send
$upscope $send
$senddefinitions $send
#0
$dumpvars
#0
#0*
#0!
$send
#10
#1!
#1#
#20
#1$
ha.vcd 34,1 Top
```

So now given this half adder files so first thing is we have to do is we have to compile them and for compiling the command would be iverilog. So iverilog then we list all the files which we need to compile and after compilation step then you see one a dot out has been created. So this is the executable which has been created. If we run this executable, then all the execution has happened. So, basically all the stimulus is given to our testbench stimulus which is generally in our testbench is given to our design under test module and you see that at every time it says that at what other values of sum and carry.

So, it looks correct A equal to 0, B equal to 1, sum should be 1 and carry should be 0. So in the other similar way so all the combinations are giving us correct result. So, let us see the output along with that so this output whatever we are writing in dollar display that would be given on the prompt or basically would be generated here itself because we have also written dumpvars and we have generated vcd file so this vcd file is here. So, if we would like to see this vcd file this is actually a text file, but we can see this text file only using some particular editor or some particular tools.

(Refer Slide Time: 13:44)

```
neera@localhost:~$ cd /home/neera/verilogtest/
neera@localhost:~/verilogtest$ ls
ha_tb.v  ha.v
neera@localhost:~/verilogtest$ vi ha.v
neera@localhost:~/verilogtest$ vi ha_tb.v
neera@localhost:~/verilogtest$ iverilog ha.v ha_tb.v
neera@localhost:~/verilogtest$ ls
a.out  ha_tb.v  ha.v
neera@localhost:~/verilogtest$ ./a.out
VCD info: dumpfile ha.vcd opened for output.
      10 A: 0, B: 0, Sum: 0, Carry:0
      20 A: 1, B: 0, Sum: 1, Carry:0
      30 A: 0, B: 1, Sum: 1, Carry:0
      40 A: 1, B: 1, Sum: 0, Carry:1
neera@localhost:~/verilogtest$ ls
a.out  ha_tb.v  ha.v  ha.vcd
neera@localhost:~/verilogtest$ vi ha.vcd
neera@localhost:~/verilogtest$ gtkwave ha.vcd
```



So gtk wave is a popular tool which can be used to see this vcd file. So let us see this vcd file using gtk wave. If gtk wave is not installed on to your system, first you have to install gtk wave and then you can use that to run or to see these particular waveforms. sSo gtk wave has been started. In gtk wave, I can see there are, these are the four signals which are there in HA dot TB and I can use any of them, I can append them in my signal waveforms.

So, here I can see these signals that for initial 0 to 10 nanosecond all the values were 0 and after 10 nanosecond values has been changed ra and rb has been changed so the our output is also changed. On the other hand, if I want to see my signals inside ha dot inst that means that particular instance of half adder then I can select those signals also and then append them in my signal viewer or gtk wave viewer. So, there also I can see the similar signal. So, you can

use this gtk wave to view the waveform and you can use dollar display to print all the outputs and this is how we will see how we can write in Verilog.

So, in summary in today's module what we have seen that how we can write a very simple program like half adder in Verilog and the learning style because Verilog is going to be quite a complete language, extensive language which has lot of syntax. So, what we will do is we will pick and choose small syntax like this we have figured out couple of these things today and so we will use these things and we will create our modules using this.

And we will have similar demos or similar explanations whenever we introduce our new syntax in Verilog and then we can practice it by creating our own modules and that is how we would be learning a Verilog. So, it should be learning by doing rather than treating it like a language where we will first define, we learn all the syntax and then we apply. So, rather than that we would see that in what application we can use that syntax and then we will learn accordingly. Thank you very much.