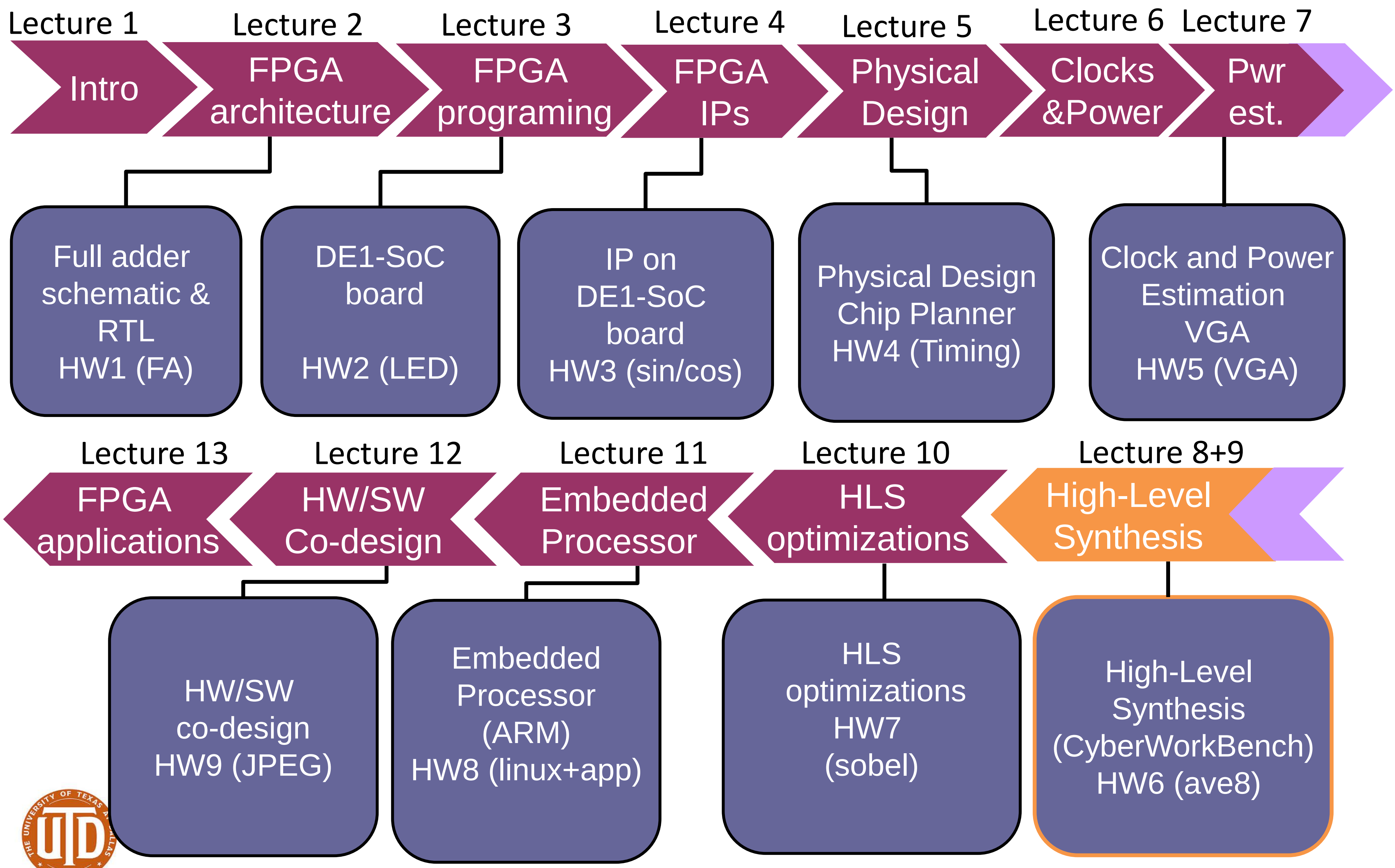
# Lecture 9 – NEC CyberWorkBench

Benjamin Carrion Schaefer
Associate Professor
Department of Electrical and Computer Engineering

## Objectives of this Lecture

- Review of commercial HLS tools
- NEC's CyberWorkBench
- Tools and flow
  - Parsing
  - Synthesizing
  - RTL generation
  - Results analysis (QOR, schematic, plot graph)
  - Verification
    - Cycle-accurate simulations
    - RTL simulations

# Lecture + Homework Synchronization



# Commercial HLS Tool

ASIC + FPGA ----------------------------------------

- Cadence – Stratus (C,C++, SystemC)
- Siemens/Mentor Graphics – Catapult (C++, SystemC)
- NEC – CyberWorkBench (C, BDL, SystemC)
- ~~Synopsys – Synphony (C, SystemC)~~

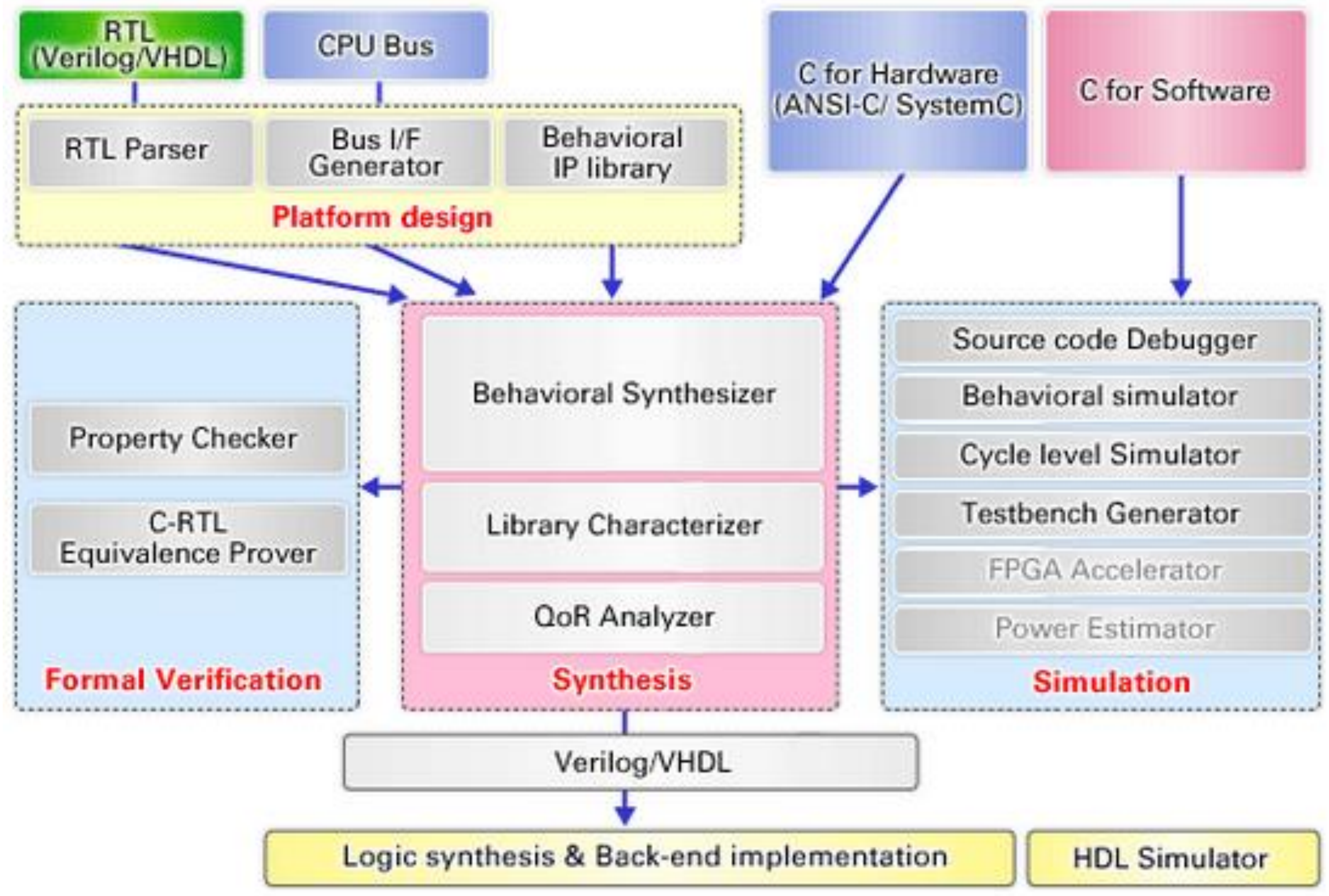
FPGA ------------------------------------------------

- Xilinx – Vivado HLS (C,C++)
- Intel – HLS compiler (c,C++)

NEC
All-in-C
CyberWorkBench™
Pioneering C-based LSI Design
CyberWorkBench®
Pioneer of C-based LSI Design

# CyberWorkBench Overview

- More info at www.cyberworkbench.com

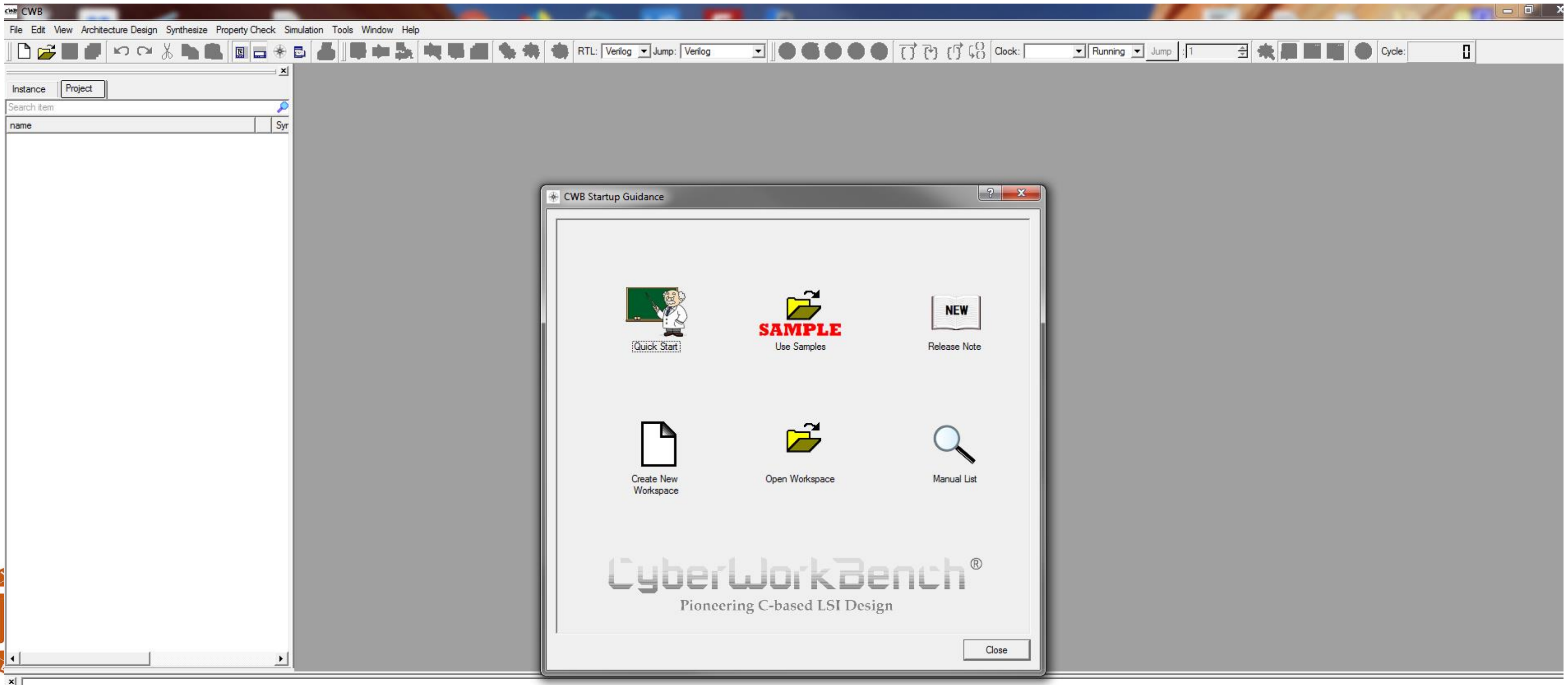


# Integrated Development Environment (IDE)

- Facilitates the use of the program
- Integrates all the separate tools (24) the conform CWB
- Allows to easy analyze and interpreted the results

%cwb &

cwb 6.1


CWB
File Edit View Architecture Design Synthesize Property Check Simulation Tools Window Help
RTL: Verilog Jump: Verilog
Clock: Running Jump Cycle:
Instance Project
Search item
name
CWB Startup Guidance
Quick Start
SAMPLE
Use Samples
NEW
Release Note
Create New Workspace
Open Workspace
Manual List
CyberWorkBench®
Pioneering C-based LSI Design
Close

## CWB Setup

1. **Setup of CWB**

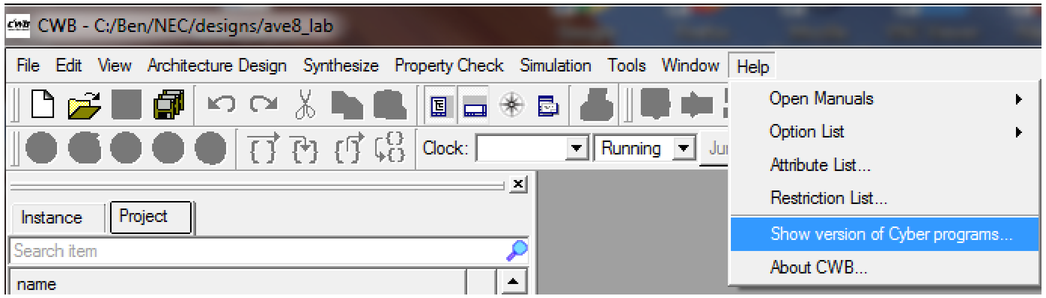
   **Edit ~/.bashrc file (e.g. % vi ~/.bashrc) to set the path to CWB and point to the license server)**

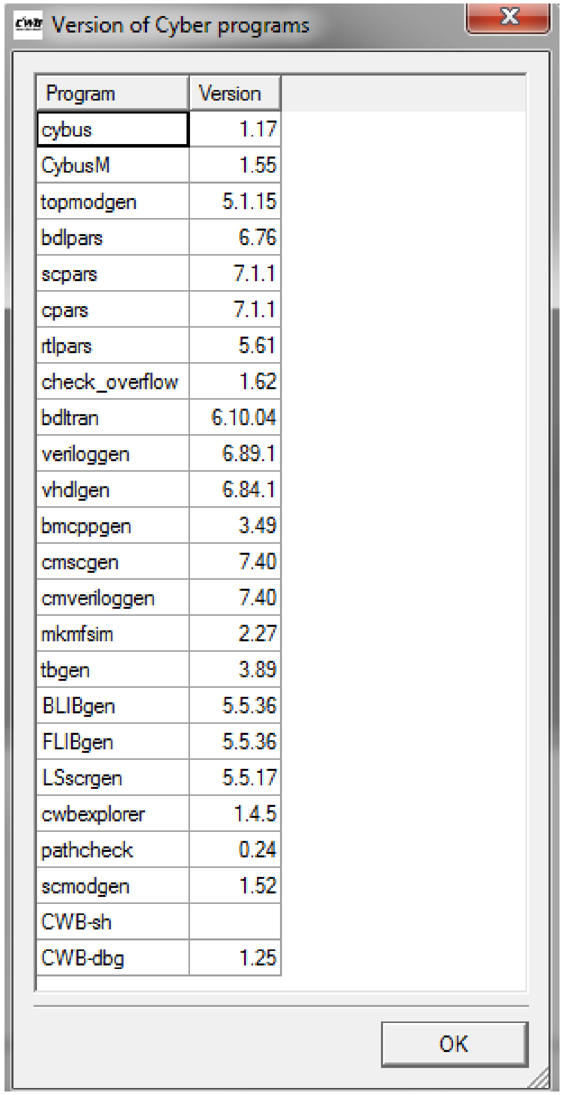
   ```
   # NEC CWB
   export CYBER_PATH=/proj/cad/cwb-6.1
   export CYBER_ADMIN_PATH=${CYBER_PATH}/linux_x86_64/admin

   export CYBER_SYSTEMC_HOME=${CYBER_PATH}/osci
   export CYBER_LIB=${CYBER_PATH}/lib
   export LD_LIBRARY_PATH=${CYBER_PATH}/lib:${LD_LIBRARY_PATH}

   export CYLMD_LICENSE_FILE=27010@legolas1.utdallas.edu

   export PATH=$PATH:${CYBER_PATH}/bin
   export PATH=$PATH:${CYBER_ADMIN_PATH}/bin
   ```

2. **Check and open CWB:**

   ```
   % /proj/cad/cwb-6.1/bin
   %which cwb
   % cwb &
   ```

# CyberWorkBench in a Nutshell


CWB - C:/Ben/NEC/designs/ave8_lab
File Edit View Architecture Design Synthesize Property Check Simulation Tools Window Help
Open Manuals
Option List
Attribute List...
Restriction List...
Show version of Cyber programs...
About CWB...
Clock:
Running
Instance
Project
Search item
name

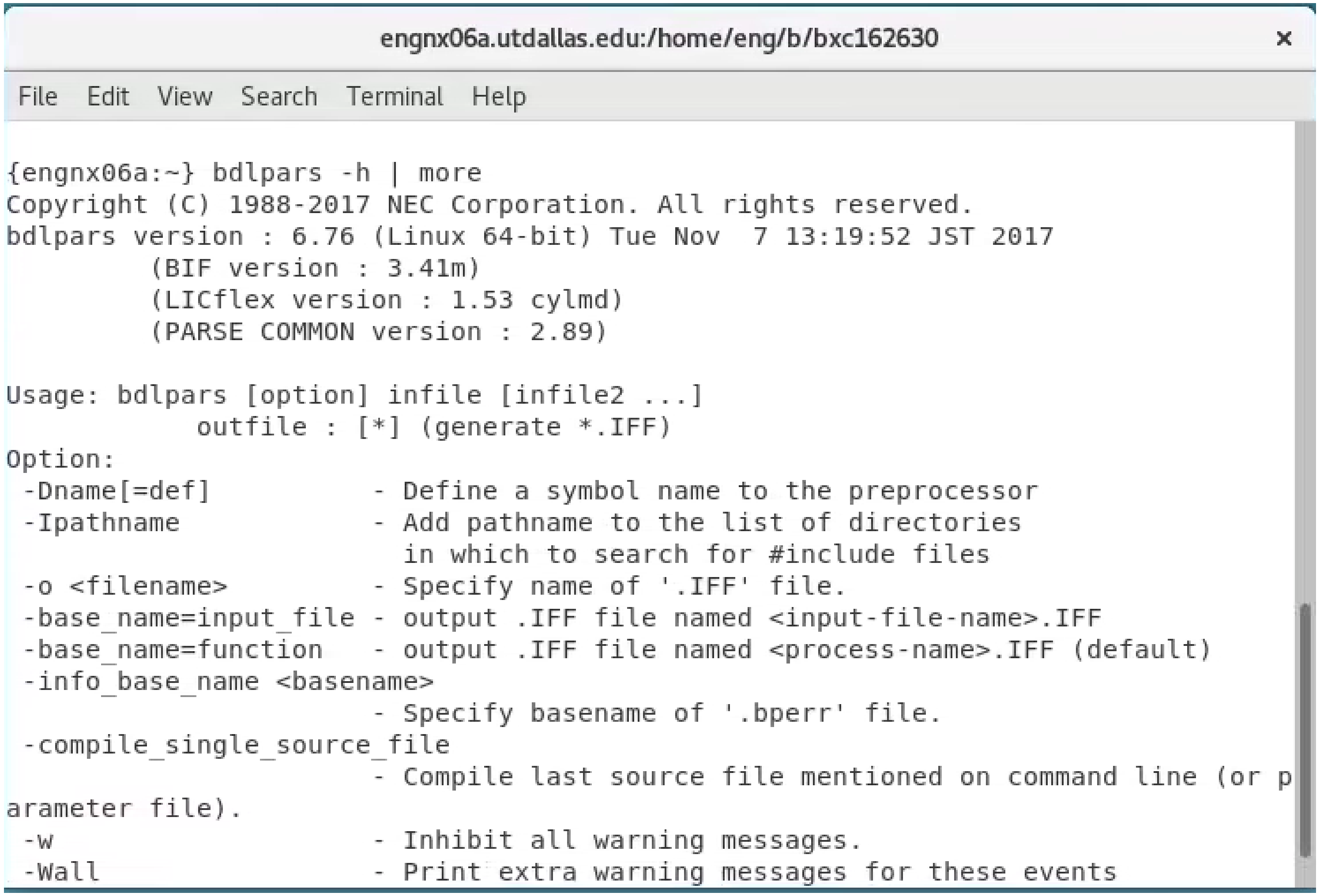


| Program | Version |
|---|---|
| cybus | 1.17 |
| CybusM | 1.55 |
| topmodgen | 5.1.15 |
| bdlpars | 6.76 |
| scpars | 7.1.1 |
| cpars | 7.1.1 |
| rtlpars | 5.61 |
| check_overflow | 1.62 |
| bdltran | 6.10.04 |
| veriloggen | 6.89.1 |
| vhdlgen | 6.84.1 |
| bmcppgen | 3.49 |
| cmscgen | 7.40 |
| cmveriloggen | 7.40 |
| mkmfsim | 2.27 |
| tbgen | 3.89 |
| BLIBgen | 5.5.36 |
| FLIBgen | 5.5.36 |
| LSscrgen | 5.5.17 |
| cwbexplorer | 1.4.5 |
| pathcheck | 0.24 |
| scmodgen | 1.52 |
| CWB-sh | |
| CWB-dbg | 1.25 |

- Composed of multiple programs
- IDE (GUI) just calls them with different options
- Each tool has its own manual and help function
- E.g.
  - cpars – Parses C code
    %cpars –h
  - bdltran – Main synthesizer
    %bdltran –h
  - veriloggen – RTL generation backend
    %veriloggen –h
  - tbgen – Testbench generator
    %tbgen –h
  - cmscgen – cycle-accurate model generator
    %cmscgen -h

# CWB from Command line

- Log in to UTD Linux server (no-machine)


engnx06a.utdallas.edu:/home/eng/b/bxc162630

```
{engnx06a:~} bdlpars -h | more
Copyright (C) 1988-2017 NEC Corporation. All rights reserved.
bdlpars version : 6.76 (Linux 64-bit) Tue Nov  7 13:19:52 JST 2017
         (BIF version : 3.41m)
         (LICflex version : 1.53 cylmd)
         (PARSE COMMON version : 2.89)

Usage: bdlpars [option] infile [infile2 ...]
            outfile : [*] (generate *.IFF)
Option:
 -Dname[=def]           - Define a symbol name to the preprocessor
 -Ipathname             - Add pathname to the list of directories
                          in which to search for #include files
 -o <filename>          - Specify name of '.IFF' file.
 -base_name=input_file - output .IFF file named <input-file-name>.IFF
 -base_name=function    - output .IFF file named <process-name>.IFF (default)
 -info_base_name <basename>
                        - Specify basename of '.bperr' file.
 -compile_single_source_file
                        - Compile last source file mentioned on command line (or p
arameter file).
 -w                     - Inhibit all warning messages.
 -Wall                  - Print extra warning messages for these events
```

# Generated Files in CWB's HLS Process

- Input : foo.c (ANSI C), foo.bdl (BDL), foo.sc/cpp (SystemC)

1. **Parse code** → syntactical errors?

   %cpars/bdlpars/scpars foo.c→ foo.IFF

2. **Synthesize code** → Allocation, Scheduling and Binding

   %bdltran foo.IFF –c1000 –s –lfl asic45.FLIB –lb asic45.BLIB

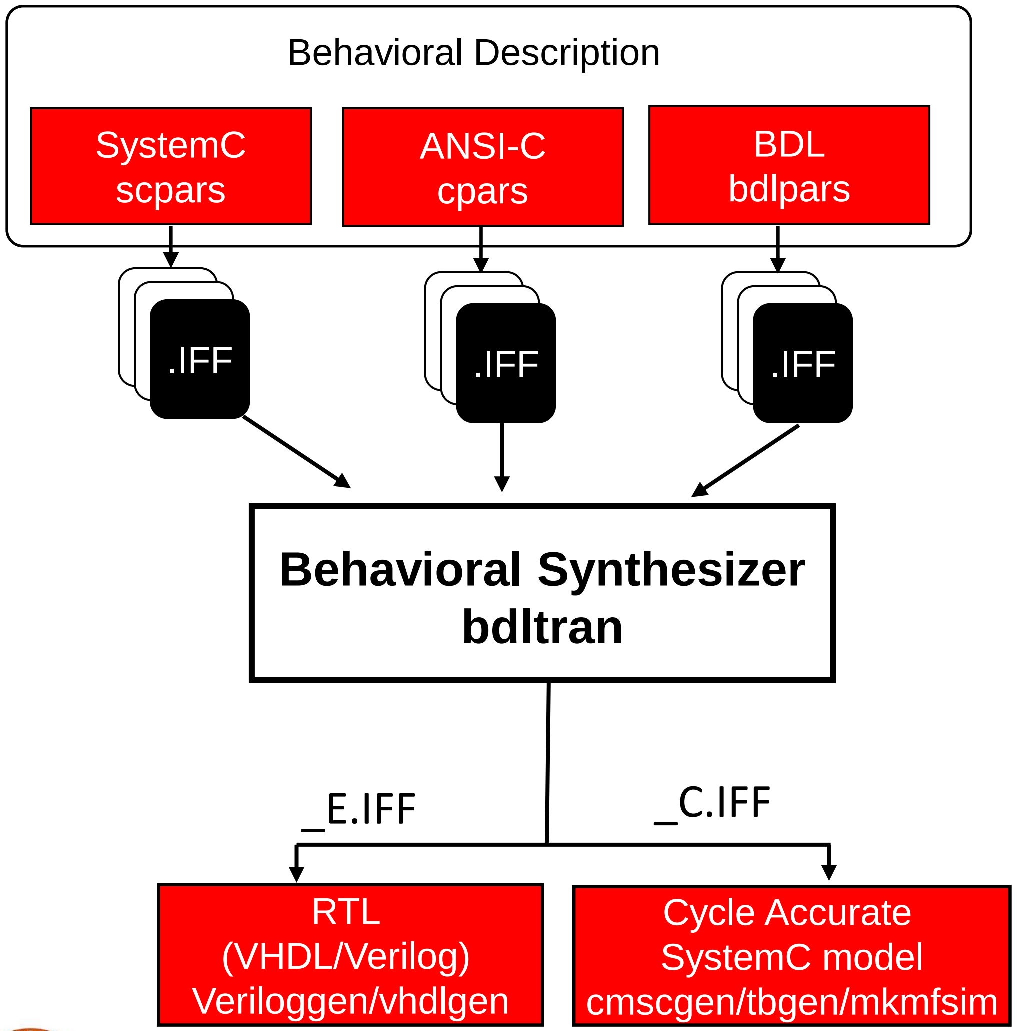
   → foo_C.IFF (result of scheduling phase), foo_E.IFF (result of binding phase). Clock units 1/100ns →1000 x 1/100ns=10ns

3. **RTL generation**

   %veriloggen foo_E.IFF **or**

   %vhdlgen foo_E.IFF → foo_E.v, foo_E.vhd

# CWB's Tool flow and Files Overview

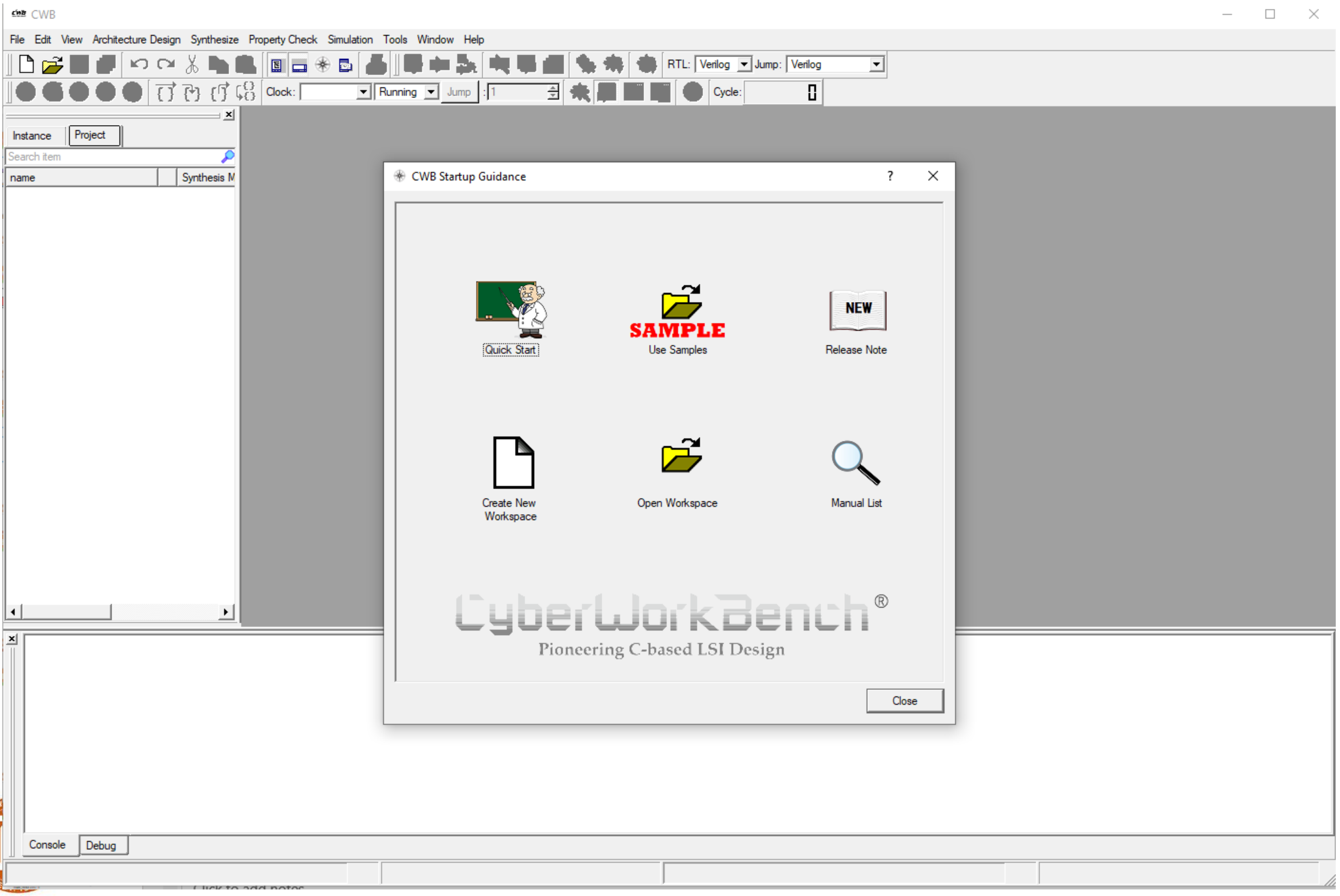


- **Inputs**
  - SystemC, ANSI-C and BDL
- **Outputs**
  - Optimized RTL Circuit (VHDL/Verilog)
  - SystemC Cycle Accurate Simulation Model
    - Fast cycle-accurate simulation model for single and multiple processes for full SoC simulation

# Input to Synthesis Process

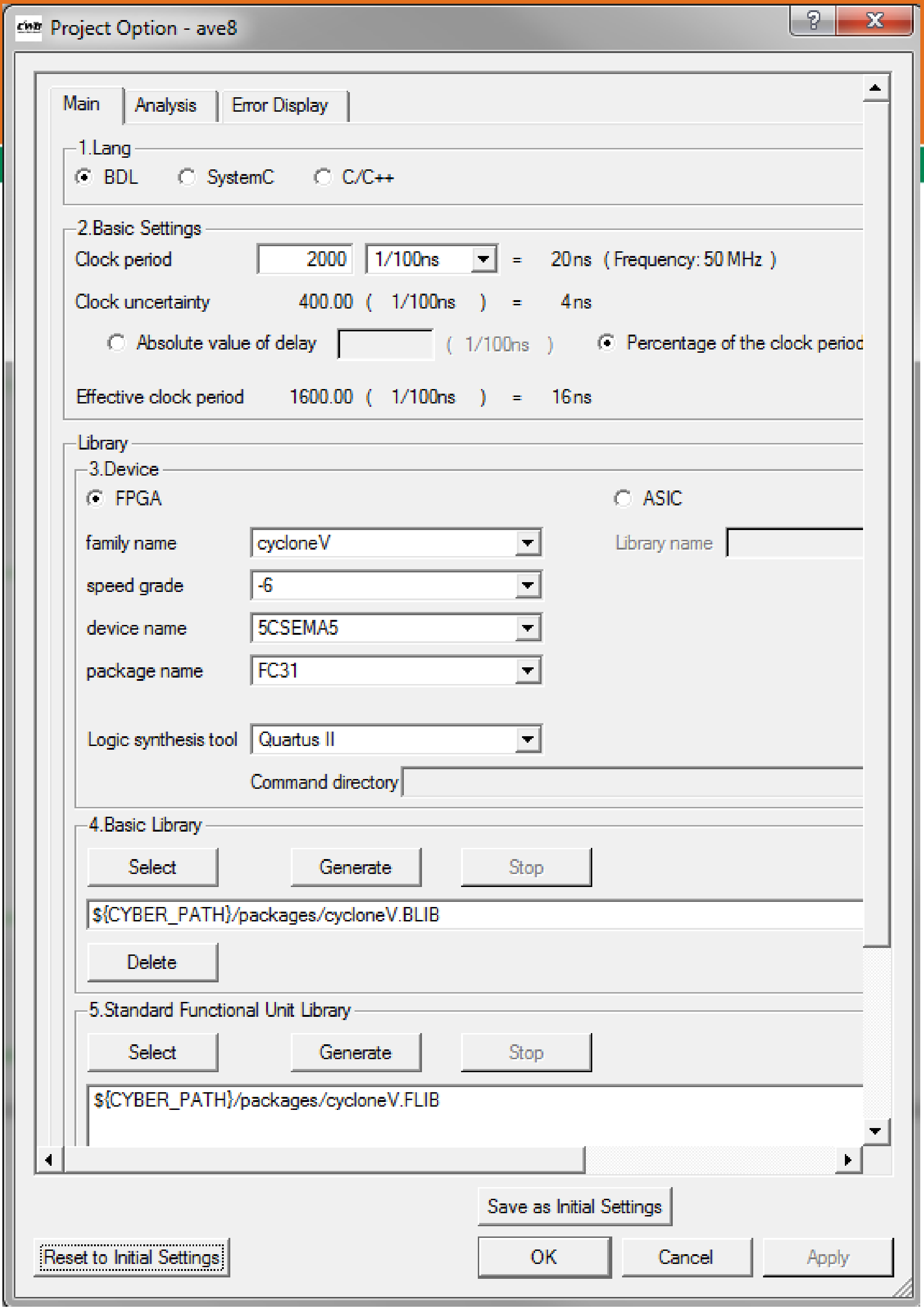
- C/BDL/SystemC file
- Technology libraries
  - .FLIB : contains area and delay of FUs
  - .BLIB : contains area and delay of basic operators
- Synthesis options
  - Target synthesis frequency (-c1000 = 10ns=100Mhz)
  - Synthesis mode (automatic -s, manual -sN, pipelined)
  - Other options , e.g., synthesize arrays as memories or register, unroll loops or not.

# Creating a new Project



# IDE Project

- Technology libraries and target synthesis frequency specified in IDE when project is generated
- Can be modified in synthesis option dialog window


Project Option - ave8

Main | Analysis | Error Display

1.Lang
[✓ BDL] SystemC C/C++

2.Basic Settings
Clock period 2000 1/100ns = 20ns (Frequency: 50 MHz)
Clock uncertainty 400.00 ( 1/100ns ) = 4ns
Absolute value of delay ( 1/100ns ) [✓ Percentage of the clock period]
Effective clock period 1600.00 ( 1/100ns ) = 16ns

Library
3.Device
[✓ FPGA] ASIC
family name cycloneV
Library name
speed grade -6
device name 5CSEMA5
package name FC31
Logic synthesis tool Quartus II
Command directory

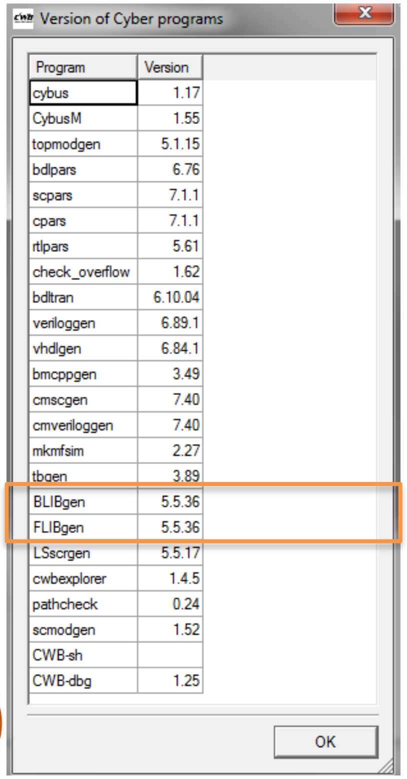
4.Basic Library
Select Generate Stop
${CYBER_PATH}/packages/cycloneV.BLIB
Delete

5.Standard Functional Unit Library
Select Generate Stop
${CYBER_PATH}/packages/cycloneV.FLIB

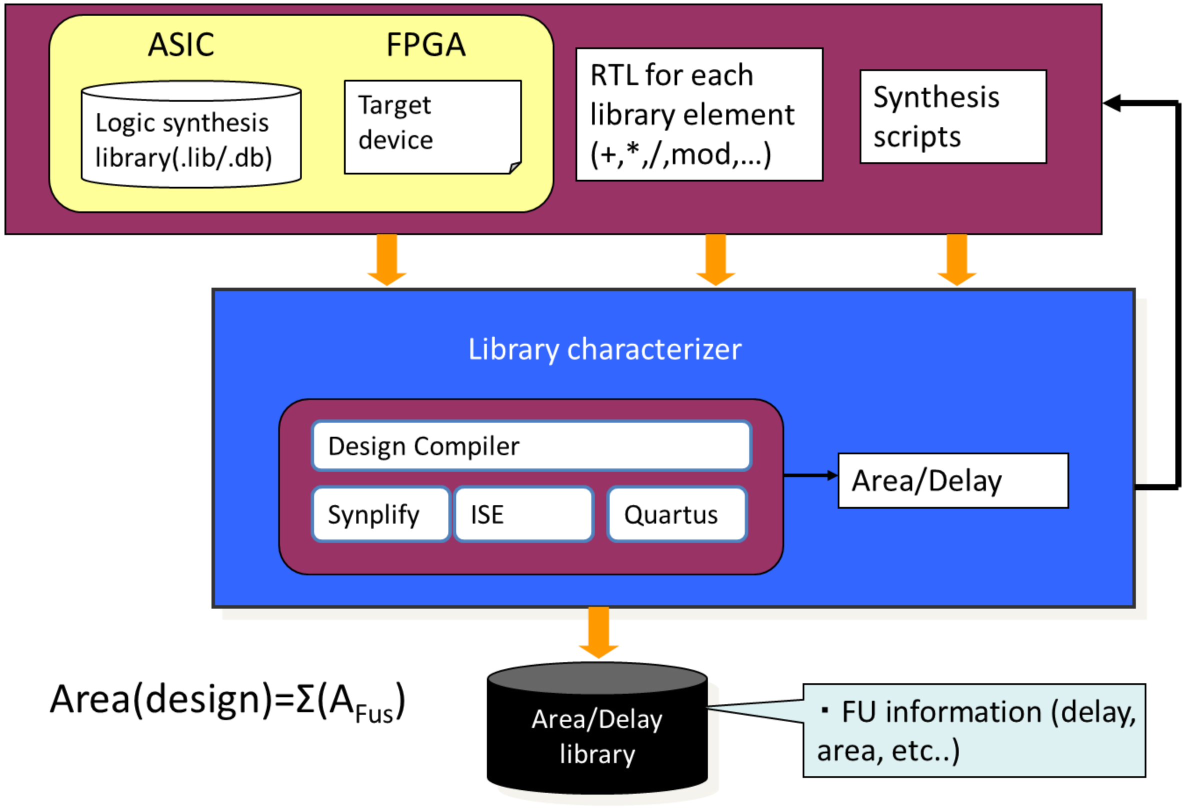
Save as Initial Settings
Reset to Initial Settings OK Cancel Apply

# FLIB/BLIB files

- Generic ASIC files given for reference → Need to call library generator with fab's technology library
- FPGA files given, but can be re-generated too with library generator
- CWB's FLIB and BLIB library generators
  - FLIBgen
  - BLIBgen



| Program | Version |
|---|---|
| cybus | 1.17 |
| CybusM | 1.55 |
| topmodgen | 5.1.15 |
| bdlpars | 6.76 |
| scpars | 7.1.1 |
| cpars | 7.1.1 |
| rtlpars | 5.61 |
| check_overflow | 1.62 |
| bdltran | 6.10.04 |
| veriloggen | 6.89.1 |
| vhdlgen | 6.84.1 |
| bmcppgen | 3.49 |
| cmscgen | 7.40 |
| cmveriloggen | 7.40 |
| mkmfsim | 2.27 |
| tbgen | 3.89 |
| BLIBgen | 5.5.36 |
| FLIBgen | 5.5.36 |
| LSscrgen | 5.5.17 |
| cwbexplorer | 1.4.5 |
| pathcheck | 0.24 |
| scmodgen | 1.52 |
| CWB-sh | |
| CWB-dbg | 1.25 |

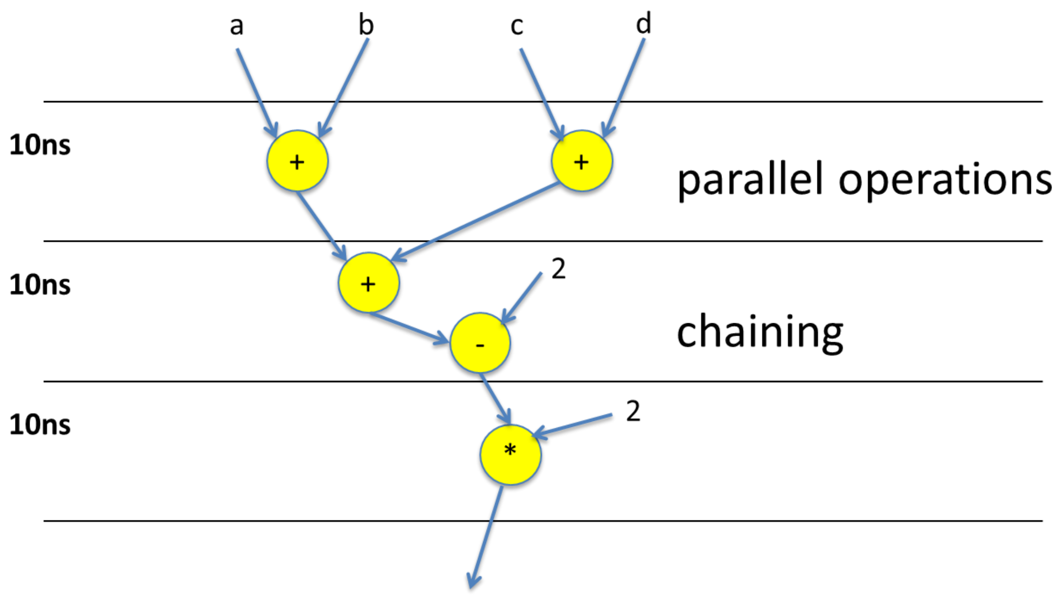


$$\text{Area(design)} = \Sigma(A_{\text{Fus}})$$

# Functional Units Chaining

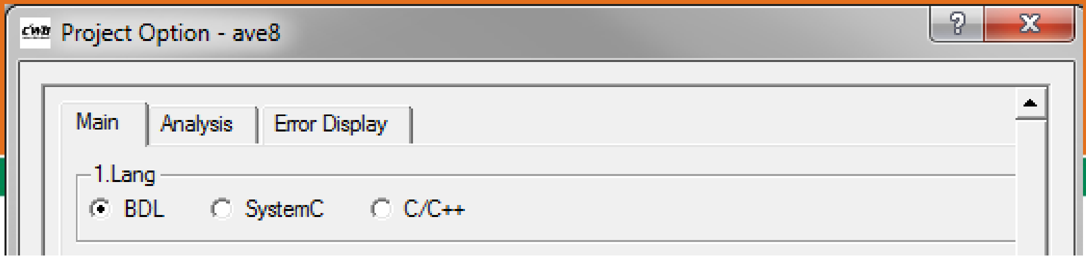
- Include chaining effect
- E.g.

```
#@VERSION{2.00}
#@LIB{CWBSTDFLIB}
#@LIBTYPE{STANDARD}
#@KIND{BASIC_OPERATOR}
#@UNIT 1/10ps
#@SYN_TOOL{DC}
@FLIB {
   NAME      add1u
   KIND       +
   BITWIDTH   1
   DELAY      900
   CHAIN_FROM add1u : 900
   CHAIN_FROM add1s : 900
   CHAIN_FROM add2u : 900
   CHAIN_FROM add2s : 900
   CHAIN_FROM add3u : 400
   CHAIN_FROM add3s : 0
   CHAIN_FROM add4u : 0
   CHAIN_FROM add4s : 0
   CHAIN_FROM add8u : 0
   CHAIN_FROM add8s : 0
   CHAIN_FROM add12u : 0
```



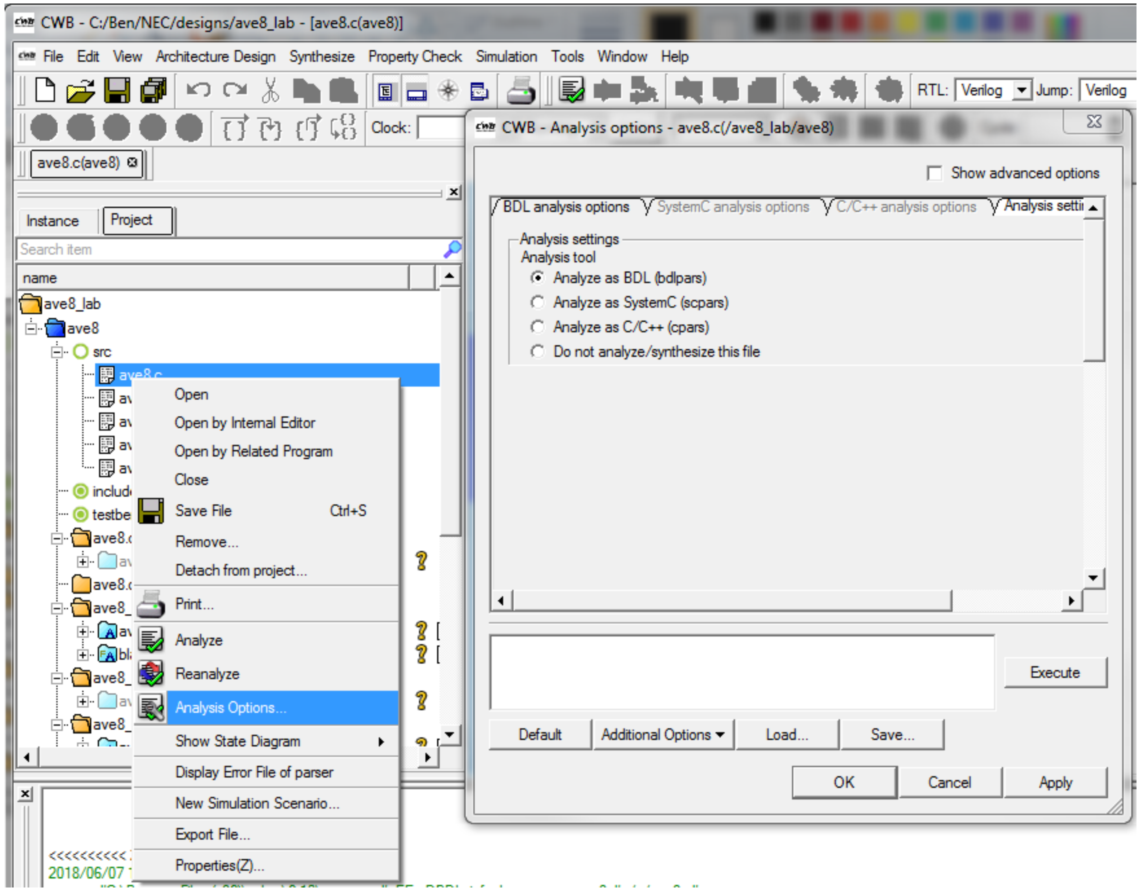
# Parser Selection

- CWB takes as inputs 3 languages → has 3 parsers
  - BDL → bdlpars
  - ANSI-C → cpars
    - // Cyber func=process
  - SystemC → scpars
- By default, the IDE uses the final extension to determine which parser to use → can be overwritten


Project Option - ave8

Main | Analysis | Error Display

1.Lang: [✓ BDL] SystemC C/C++


CWB - C:/Ben/NEC/designs/ave8_lab - [ave8.c(ave8)]

CWB - Analysis options - ave8.c(/ave8_lab/ave8)

Show advanced options

BDL analysis options | SystemC analysis options | C/C++ analysis options | Analysis setti

Analysis settings

Analysis tool

- [✓ Analyze as BDL (bdlpars)]
- Analyze as SystemC (scpars)
- Analyze as C/C++ (cpars)
- Do not analyze/synthesize this file

# Setting Pre-compiler Directives

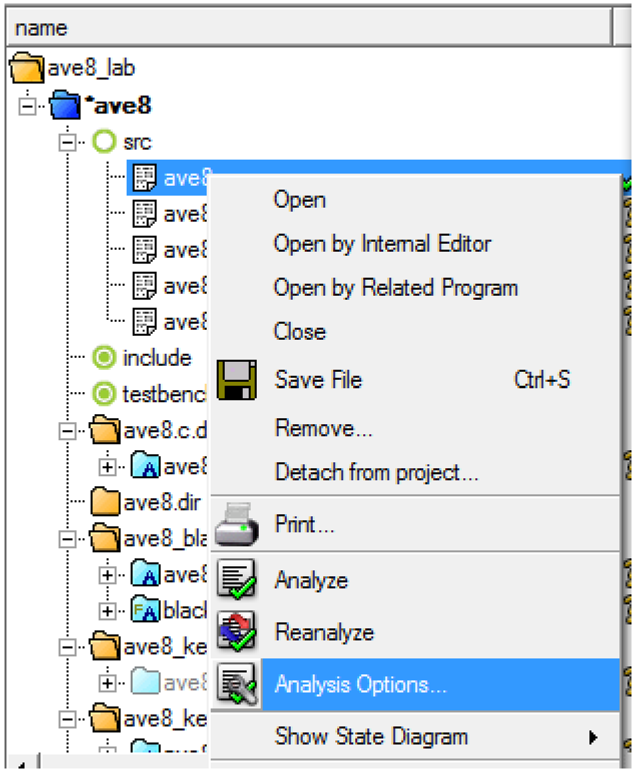
- When calling parser from command line

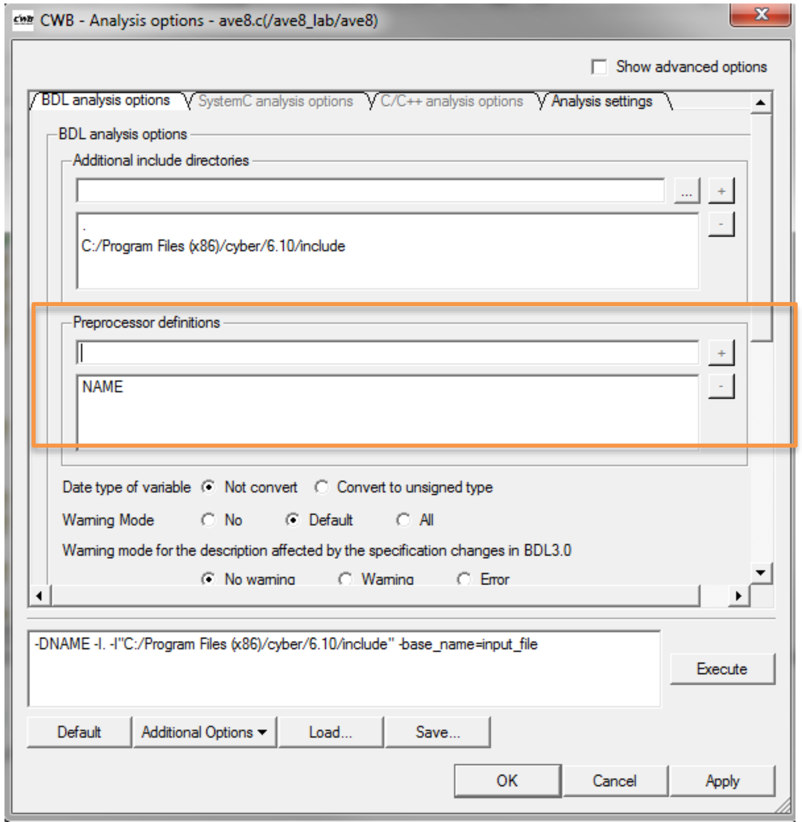
  %bdlpars foo.bdl –**D**NAME

- At IDE

  Analysis options → preprocessor directives

```
#ifdef NAME
  x=10;
#else
   x=0
#endif
```





BDL analysis options

Additional include directories: .  C:/Program Files (x86)/cyber/6.10/include

Preprocessor definitions: NAME

Date type of variable [✓ Not convert] Convert to unsigned type

Warning Mode No [✓ Default] All

Warning mode for the description affected by the specification changes in BDL3.0 [✓ No warning] Warning Error

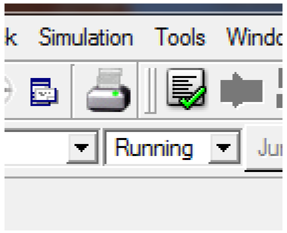
```
-DNAME -I. -I"C:/Program Files (x86)/cyber/6.10/include" -base_name=input_file
```

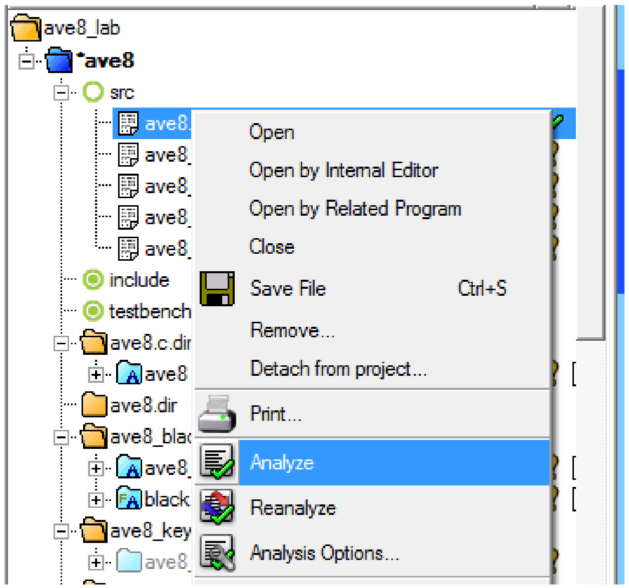
# Parsing

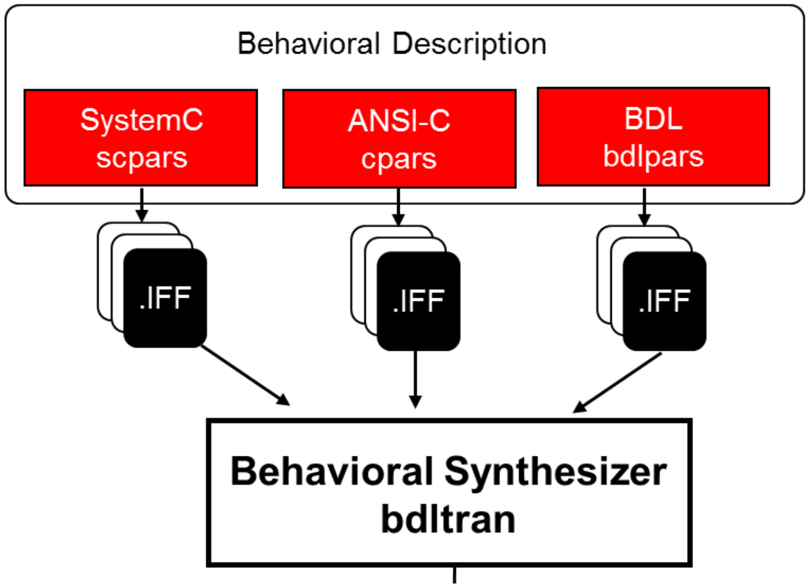
- Command line

  %cpars foo.c → foo.IFF

- IDE

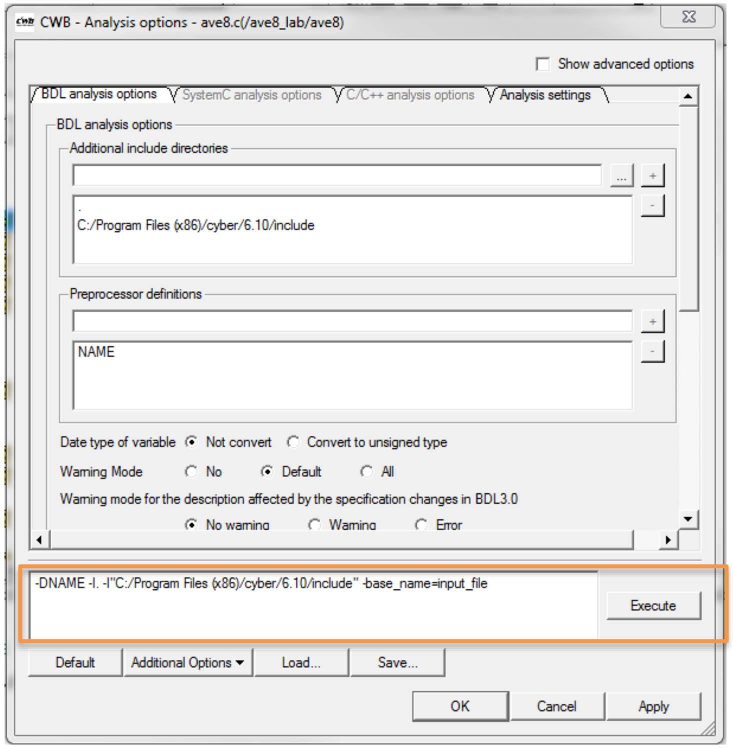






# IDE Parser

- Dialog window text box shows options selected
- Console window shows tool and options passed



CWB - Analysis options - ave8.c(/ave8_lab/ave8)

Show advanced options

BDL analysis options | SystemC analysis options | C/C++ analysis options | Analysis settings

BDL analysis options

Additional include directories

.
C:/Program Files (x86)/cyber/6.10/include

Preprocessor definitions

NAME

Date type of variable [✓ Not convert] Convert to unsigned type

Warning Mode No [✓ Default] All

Warning mode for the description affected by the specification changes in BDL3.0

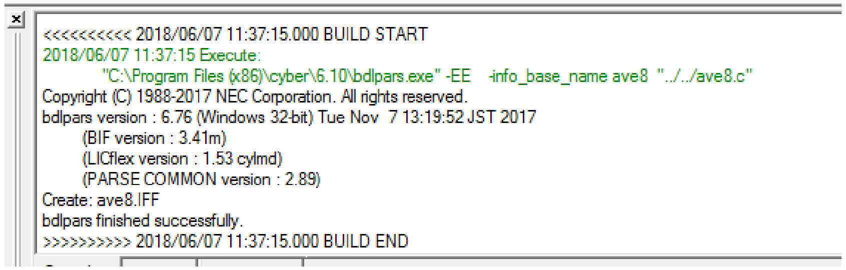
[✓ No warning] Warning Error

```
-DNAME -I. -I"C:/Program Files (x86)/cyber/6.10/include" -base_name=input_file
```

Execute

Default | Additional Options ▾ | Load... | Save...

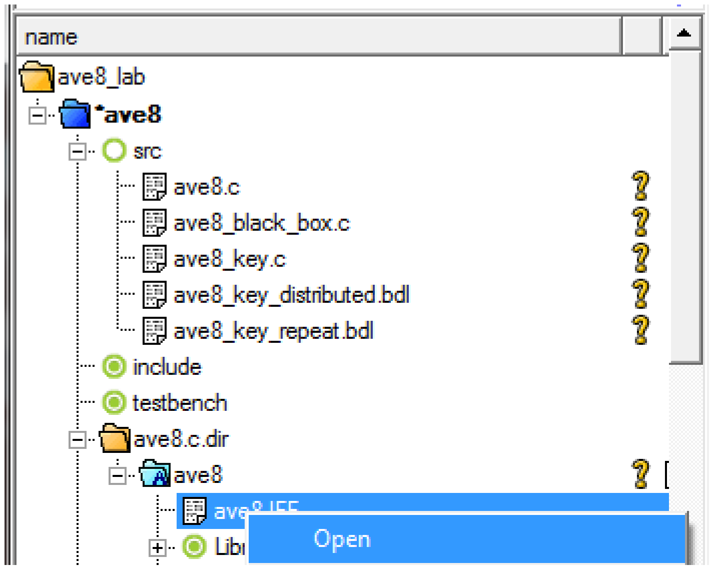
OK | Cancel | Apply

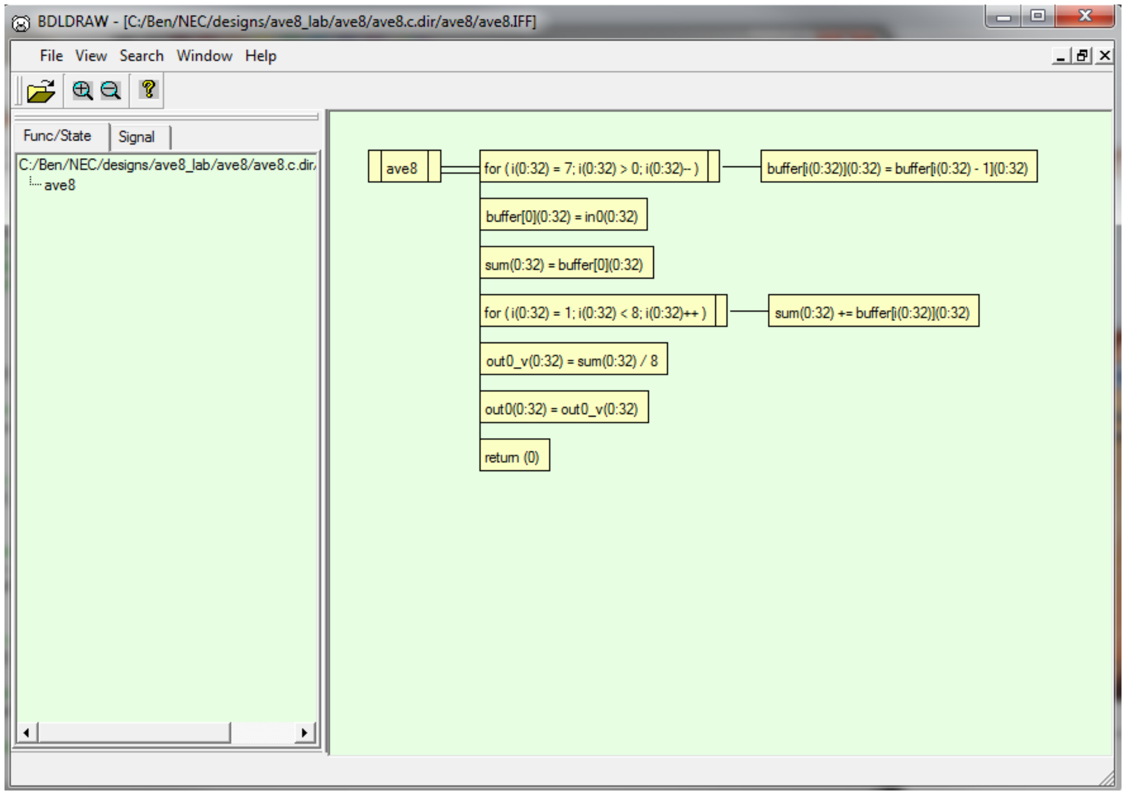


```
<<<<<<<<<< 2018/06/07 11:37:15.000 BUILD START
2018/06/07 11:37:15 Execute:
         "C:\Program Files (x86)\cyber\6.10\bdlpars.exe" -EE   -info_base_name ave8  "../../ave8.c"
Copyright (C) 1988-2017 NEC Corporation. All rights reserved.
bdlpars version : 6.76 (Windows 32-bit) Tue Nov  7 13:19:52 JST 2017
       (BIF version : 3.41m)
       (LICflex version : 1.53 cylmd)
       (PARSE COMMON version : 2.89)
Create: ave8.IFF
bdlpars finished successfully.
>>>>>>>>>> 2018/06/07 11:37:15.000 BUILD END
```

# Parser’s Output (.IFF)

- Output of parser is a .IFF file → allows to have multiple parsers and generate a common file format for all of them
- %bdldraw foo.IFF → displays the parse tree graphically

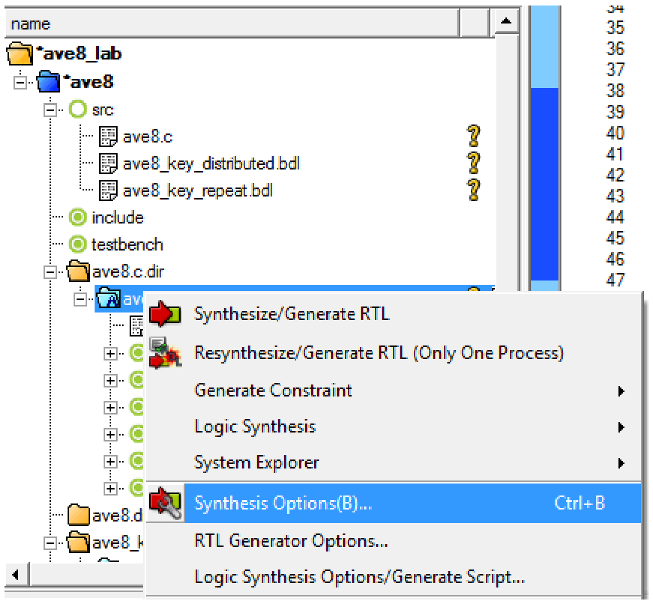


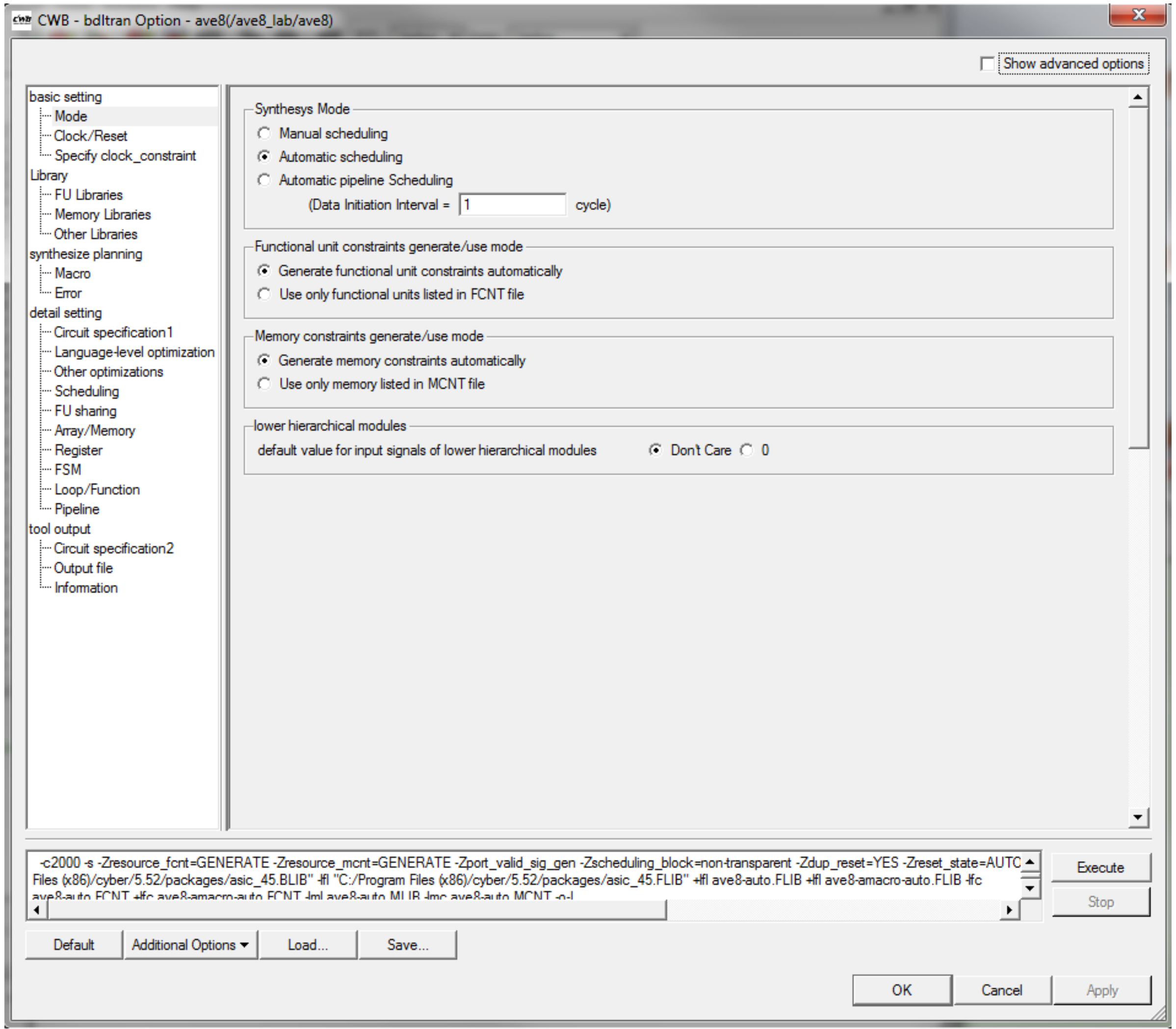

BDLDRAW - [C:/Ben/NEC/designs/ave8_lab/ave8/ave8.c.dir/ave8/ave8.IFF]
File View Search Window Help
Func/State
Signal
C:/Ben/NEC/designs/ave8_lab/ave8/ave8.c.dir/
ave8

```
ave8
for ( i(0:32) = 7; i(0:32) > 0; i(0:32)-- )
    buffer[i(0:32)](0:32) = buffer[i(0:32) - 1](0:32)
buffer[0](0:32) = in0(0:32)
sum(0:32) = buffer[0](0:32)
for ( i(0:32) = 1; i(0:32) < 8; i(0:32)++ )
    sum(0:32) += buffer[i(0:32)](0:32)
out0_v(0:32) = sum(0:32) / 8
out0(0:32) = out0_v(0:32)
return (0)
```

# CWB's Synthesizer - bdltran

- Many, many, many options




CWB - bdltran Option - ave8(/ave8_lab/ave8)

☐ Show advanced options

basic setting
- Mode
- Clock/Reset
- Specify clock_constraint

Library
- FU Libraries
- Memory Libraries
- Other Libraries

synthesize planning
- Macro
- Error

detail setting
- Circuit specification1
- Language-level optimization
- Other optimizations
- Scheduling
- FU sharing
- Array/Memory
- Register
- FSM
- Loop/Function
- Pipeline

tool output
- Circuit specification2
- Output file
- Information

Synthesys Mode
- ○ Manual scheduling
- [✓ Automatic scheduling]
- ○ Automatic pipeline Scheduling
  (Data Initiation Interval = 1 cycle)

Functional unit constraints generate/use mode
- [✓ Generate functional unit constraints automatically]
- ○ Use only functional units listed in FCNT file

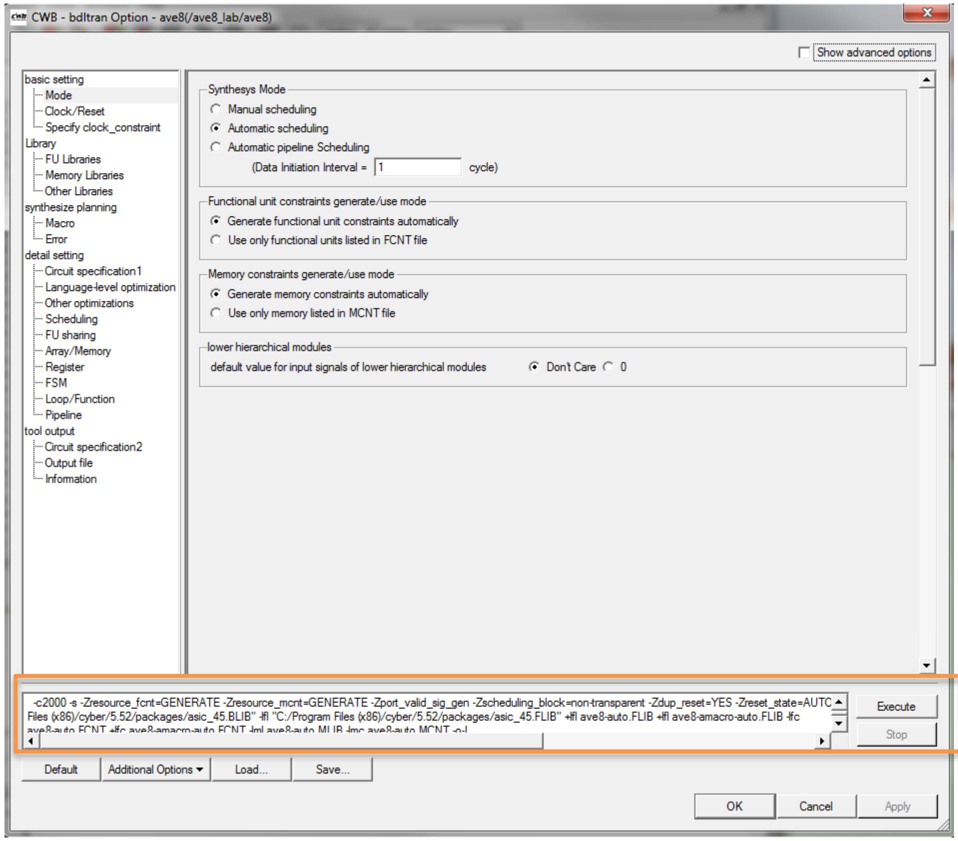
Memory constraints generate/use mode
- [✓ Generate memory constraints automatically]
- ○ Use only memory listed in MCNT file

lower hierarchical modules
default value for input signals of lower hierarchical modules [✓ Don't Care] ○ 0

```
 -c2000 -s -Zresource_fcnt=GENERATE -Zresource_mcnt=GENERATE -Zport_valid_sig_gen -Zscheduling_block=non-transparent -Zdup_reset=YES -Zreset_state=AUTO
Files (x86)/cyber/5.52/packages/asic_45.BLIB" -lfl "C:/Program Files (x86)/cyber/5.52/packages/asic_45.FLIB" +lfl ave8-auto.FLIB +lfl ave8-amacro-auto.FLIB -lfc
ave8-auto.FCNT +lfc ave8-amacro-auto.FCNT -lml ave8-auto.MLIB -lmc ave8-auto.MCNT -o-l
```

# Command line bdltran

- bdltran has many options (%bdltran –H)
- Text box in GUI shows which options are selected
- %bdltran  foo.IFF –c2000 -s -Zresource_fcnt=USE -Zresource_mcnt=GENERATE -EE -lb ${CYBER_PATH}/packages/asic_45.BLIB -lfl ${CYBER_PATH}/packages/asic_45.FLIB +lfl ave8-auto.FLIB +lfl ave8-amacro-auto.FLIB -lfc ave8-auto.FCNT


CWB - bdltran Option - ave8(/ave8_lab/ave8)

☐ Show advanced options

Synthesys Mode
- ○ Manual scheduling
- ● Automatic scheduling
- ○ Automatic pipeline Scheduling
  (Data Initiation Interval = 1 cycle)

Functional unit constraints generate/use mode
- ● Generate functional unit constraints automatically
- ○ Use only functional units listed in FCNT file

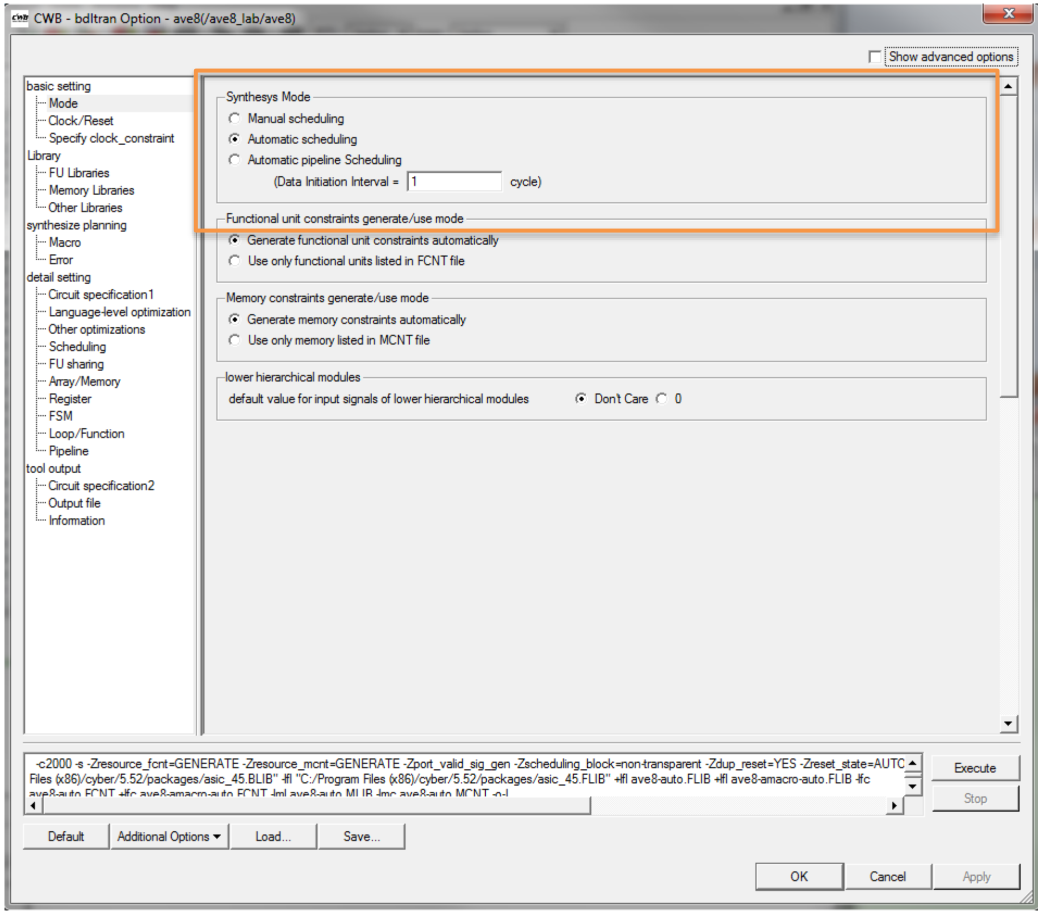
Memory constraints generate/use mode
- ● Generate memory constraints automatically
- ○ Use only memory listed in MCNT file

lower hierarchical modules
- default value for input signals of lower hierarchical modules ● Don't Care ○ 0

```
-c2000 -s -Zresource_fcnt=GENERATE -Zresource_mcnt=GENERATE -Zport_valid_sig_gen -Zscheduling_block=non-transparent -Zdup_reset=YES -Zreset_state=AUTC
Files (x86)/cyber/5.52/packages/asic_45.BLIB" -lfl "C:/Program Files (x86)/cyber/5.52/packages/asic_45.FLIB" +lfl ave8-auto.FLIB +lfl ave8-amacro-auto.FLIB -lfc
ave8-auto.FCNT +lfc ave8-amacro-auto.FCNT -lml ave8-auto.MLIB -lmc ave8-auto.MCNT -o-l
```

# Must Have Options : 1. Synthesis Mode

- Synthesis Mode
  - Automatic  (-s) (default)
    - Bdltran takes care of the scheduling/timing automatically
  - Manual (-sN)
    - Timing is specified manually by user by using $ or wait()
  - Pipelinning (-s –Zzpipeline)
    - Circuit is fully pipelined → Data Initiation Interval (DII) needs to be specified



Synthesys Mode
- Manual scheduling
- [✓ Automatic scheduling]
- Automatic pipeline Scheduling
  (Data Initiation Interval = 1 cycle)

Functional unit constraints generate/use mode
- [✓ Generate functional unit constraints automatically]
- Use only functional units listed in FCNT file

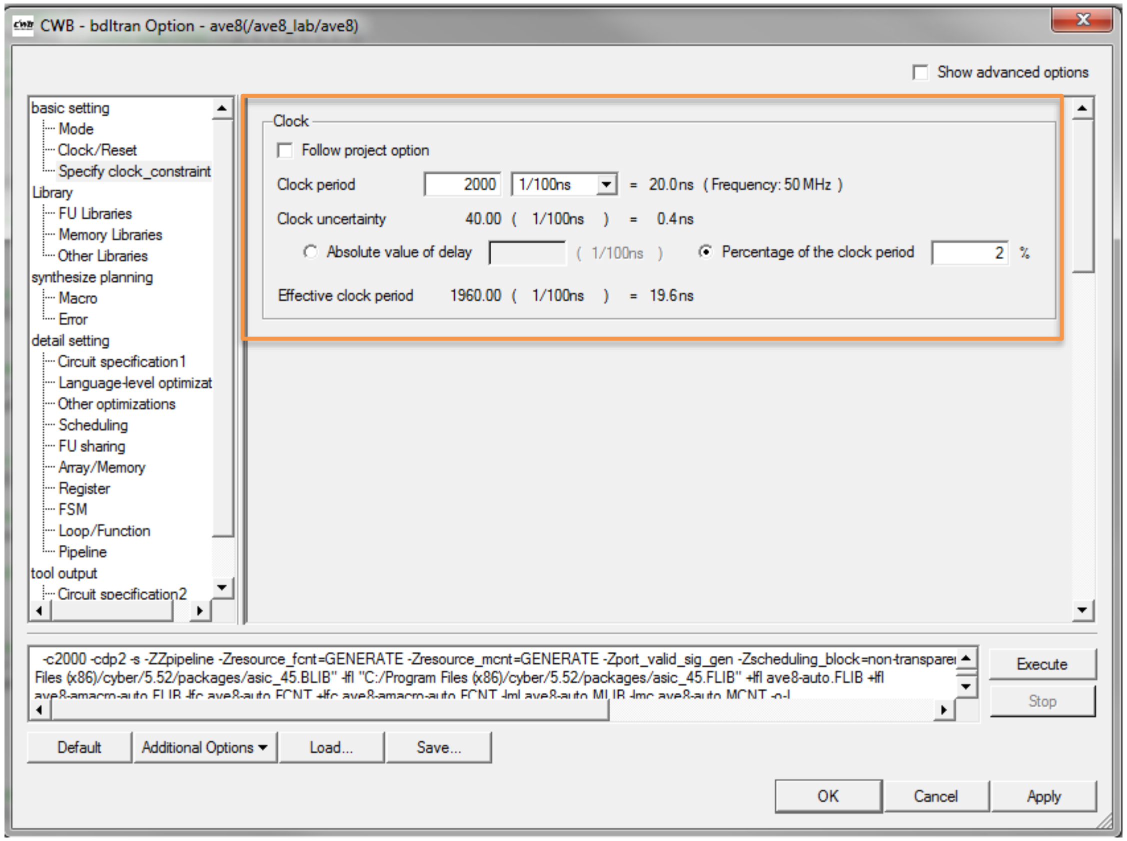
Memory constraints generate/use mode
- [✓ Generate memory constraints automatically]
- Use only memory listed in MCNT file

lower hierarchical modules
default value for input signals of lower hierarchical modules [✓ Don't Care] 0

```
-c2000 -s -Zresource_fcnt=GENERATE -Zresource_mcnt=GENERATE -Zport_valid_sig_gen -Zscheduling_block=non-transparent -Zdup_reset=YES -Zreset_state=AUTC
Files (x86)/cyber/5.52/packages/asic_45.BLIB" -lfl "C:/Program Files (x86)/cyber/5.52/packages/asic_45.FLIB" +lfl ave8-auto.FLIB +lfl ave8-amacro-auto.FLIB -lfc
ave8-auto.FCNT +lfc ave8-amacro-auto.FCNT -lml ave8-auto.MLIB -lmc ave8-auto.MCNT -o-l
```

# Must Have Options : 2. Clock Period

- Originally set when project is created (by default follows it)
- Clock uncertainty
  - Reduces the clock period by fixed value or %



Clock

- [ ] Follow project option

Clock period 2000 1/100ns = 20.0ns ( Frequency: 50 MHz )

Clock uncertainty 40.00 ( 1/100ns ) = 0.4ns

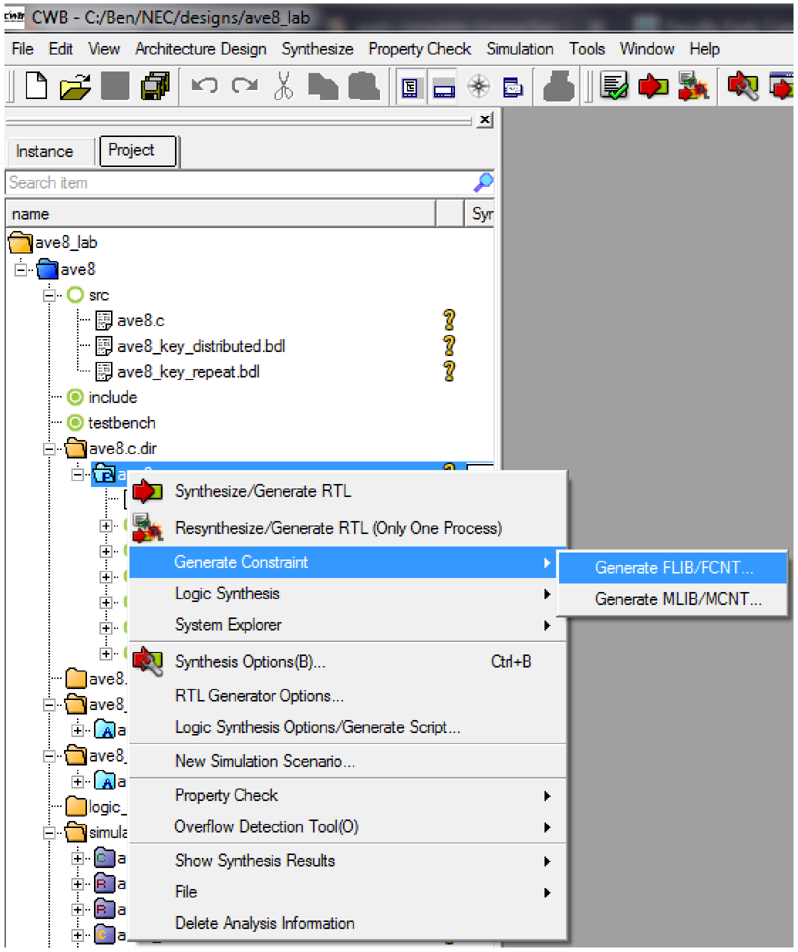
Absolute value of delay ( 1/100ns ) [✓ Percentage of the clock period] 2 %

Effective clock period 1960.00 ( 1/100ns ) = 19.6ns

```
-c2000 -cdp2 -s -ZZpipeline -Zresource_fcnt=GENERATE -Zresource_mcnt=GENERATE -Zport_valid_sig_gen -Zscheduling_block=non-transpare
Files (x86)/cyber/5.52/packages/asic_45.BLIB" -lfl "C:/Program Files (x86)/cyber/5.52/packages/asic_45.FLIB" +lfl ave8-auto.FLIB +lfl
ave8-amacro-auto.FLIB -lfc ave8-auto.FCNT +lfc ave8-amacro-auto.FCNT -lml ave8-auto.MLIB -lmc ave8-auto.MCNT -o-l
```

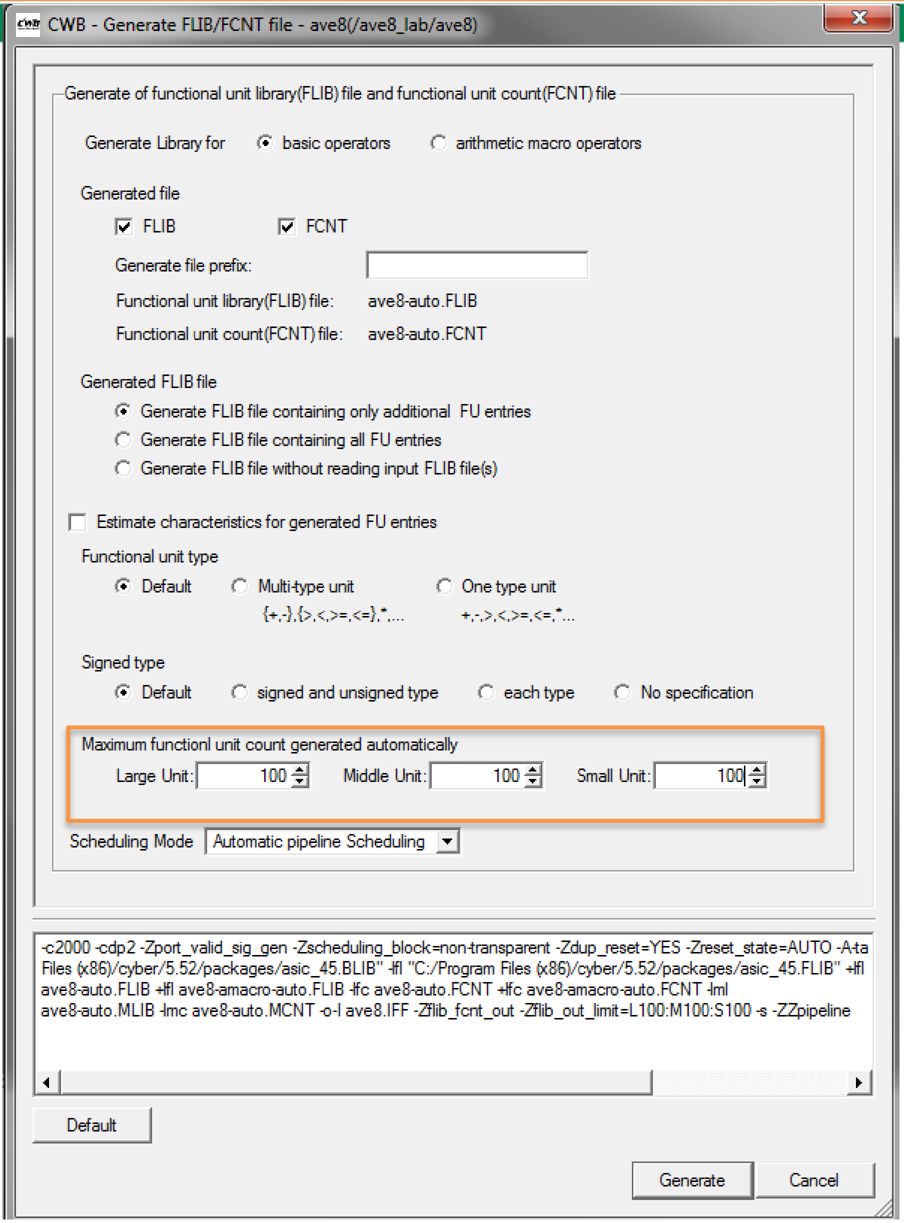
# Resource Allocation

- In the past needed to executed bdltran 2 times
  - First pass : Generate FU constraint file (FCNT)
  - Second pass : Synthesis circuit with FCNT file
- Since version 6.0 both passes done at the same time
- Previous method allowed to modify the FCNT file before doing the full synthesis
- FU constraint (FCNT)
- Memory constraint (MLIB and MCNT)
- Additional FU library file (-auto.FLIB)


CWB - C:/Ben/NEC/designs/ave8_lab
File Edit View Architecture Design Synthesize Property Check Simulation Tools Window Help
Instance
Project
Search item
name
ave8_lab
ave8
src
ave8.c
ave8_key_distributed.bdl
ave8_key_repeat.bdl
include
testbench
ave8.c.dir
Synthesize/Generate RTL
Resynthesize/Generate RTL (Only One Process)
Generate Constraint
Generate FLIB/FCNT...
Generate MLIB/MCNT...
Logic Synthesis
System Explorer
Synthesis Options(B)... Ctrl+B
RTL Generator Options...
Logic Synthesis Options/Generate Script...
New Simulation Scenario...
Property Check
Overflow Detection Tool(O)
Show Synthesis Results
File
Delete Analysis Information

# FCNT file

- Contains number and type of FUs from given FLIB file that is required to map operations from C code to RTL
- Allows to control FUs by bitwidth: large, medium and small
  - Large > 16 bits
  - Medium <16bits and > 8 bits
  - Small < 8 bits
- Option: Zflib_out_limit=L100:M100:S100



CWB - Generate FLIB/FCNT file - ave8(/ave8_lab/ave8)

Generate of functional unit library(FLIB) file and functional unit count(FCNT) file

Generate Library for [✓ basic operators] arithmetic macro operators

Generated file

- [✓ FLIB] [✓ FCNT]
- Generate file prefix:
- Functional unit library(FLIB) file: ave8-auto.FLIB
- Functional unit count(FCNT) file: ave8-auto.FCNT

Generated FLIB file

- [✓ Generate FLIB file containing only additional FU entries]
- Generate FLIB file containing all FU entries
- Generate FLIB file without reading input FLIB file(s)

☐ Estimate characteristics for generated FU entries

Functional unit type

[✓ Default] Multi-type unit {+,-},{>,<,>=,<=},*,... One type unit +,-,>,<,>=,<=,*...

Signed type

[✓ Default] signed and unsigned type each type No specification

Maximum functionl unit count generated automatically

Large Unit: 100 Middle Unit: 100 Small Unit: 100

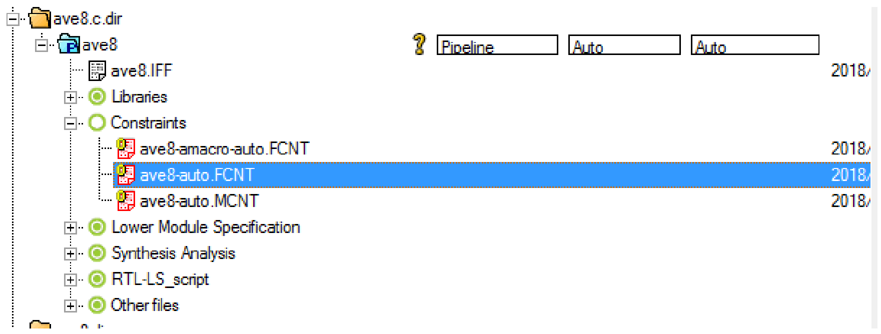
Scheduling Mode: Automatic pipeline Scheduling

```
-c2000 -cdp2 -Zport_valid_sig_gen -Zscheduling_block=non-transparent -Zdup_reset=YES -Zreset_state=AUTO -A-ta
Files (x86)/cyber/5.52/packages/asic_45.BLIB" -lfl "C:/Program Files (x86)/cyber/5.52/packages/asic_45.FLIB" +lfl
ave8-auto.FLIB +lfl ave8-amacro-auto.FLIB -lfc ave8-auto.FCNT +lfc ave8-amacro-auto.FCNT -lml
ave8-auto.MLIB -lmc ave8-auto.MCNT -o-l ave8.IFF -Zflib_fcnt_out -Zflib_out_limit=L100:M100:S100 -s -ZZpipeline
```

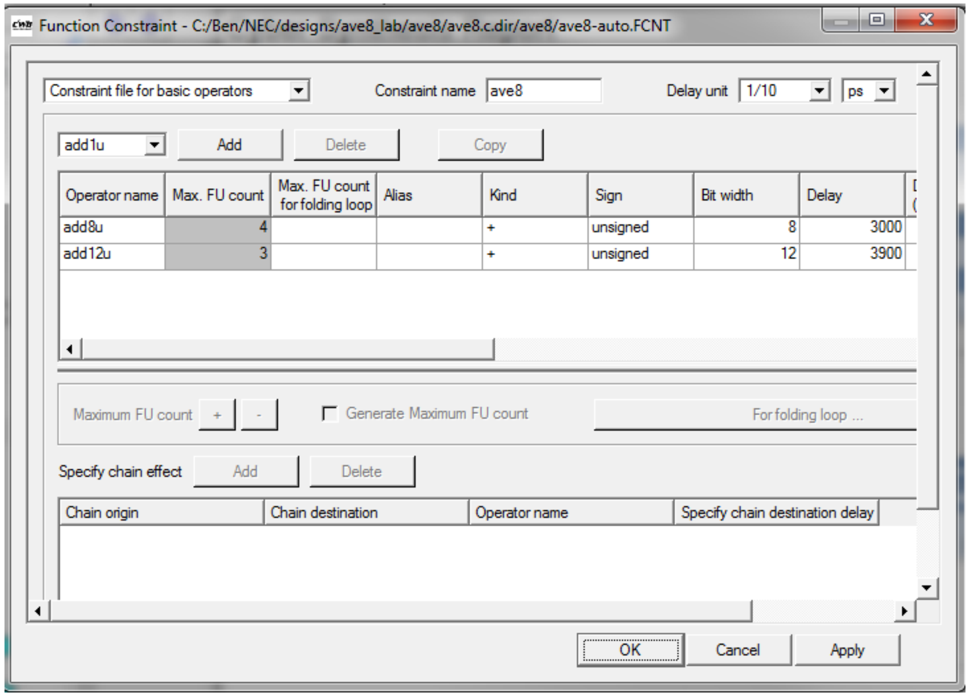
# FCNT Manual Settings

- FCNT file can be manually edited to set number of FUs of each class
  - Double click or edit with any text editor

```
#@VERSION{3.00}
#@CNT{ave8}
#@KIND{BASIC_OPERATOR}
#@CLK 200000
#@UNIT 1/10ps
@FCNT{
        NAME  add8u
        LIMIT   4  AUTO
#       COMMENT
}
@FCNT{
        NAME  add12u
        LIMIT   3  AUTO
#       COMMENT
}
#@HASH{b6836427a6977baf58df8e5ddf1f14d7}
#@END{ave8}
```



```
ave8.c.dir
  ave8                                  Pipeline  Auto  Auto
    ave8.IFF                                                    2018
    Libraries
    Constraints
      ave8-amacro-auto.FCNT                                     2018
      ave8-auto.FCNT                                            2018
      ave8-auto.MCNT                                            2018
    Lower Module Specification
    Synthesis Analysis
    RTL-LS_script
    Other files
```



Function Constraint - C:/Ben/NEC/designs/ave8_lab/ave8/ave8.c.dir/ave8/ave8-auto.FCNT

Constraint file for basic operators    Constraint name: ave8    Delay unit: 1/10 ps

add1u

| Operator name | Max. FU count | Max. FU count for folding loop | Alias | Kind | Sign | Bit width | Delay |
|---|---|---|---|---|---|---|---|
| add8u | 4 | | | + | unsigned | 8 | 3000 |
| add12u | 3 | | | + | unsigned | 12 | 3900 |

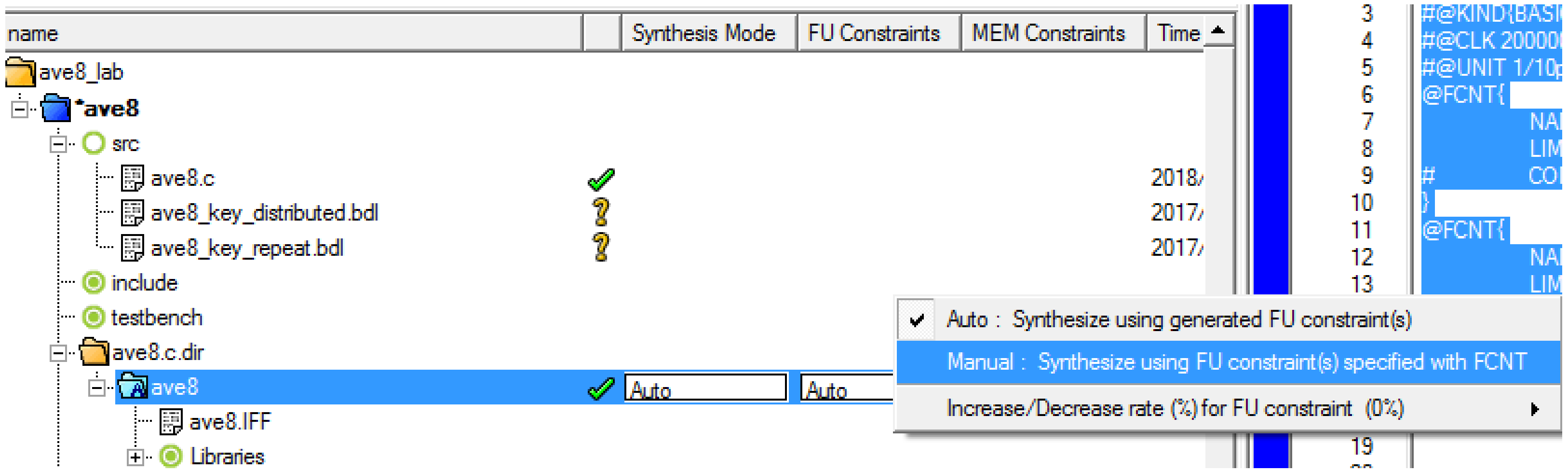
Maximum FU count    Generate Maximum FU count    For folding loop ...

Specify chain effect

| Chain origin | Chain destination | Operator name | Specify chain destination delay |
|---|---|---|---|

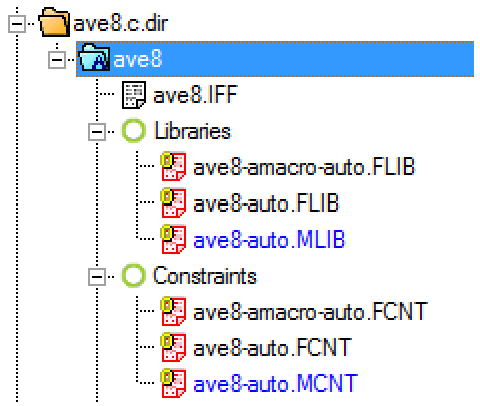
# FCNT

- By default, a new FCNT file is generated in each new synthesis
  - -Zresource_fcnt=GENERATE
- Need to specify to use manually modified FCNT file manually
  - -Zresource_fcnt=USE



# Memory Constraint Files

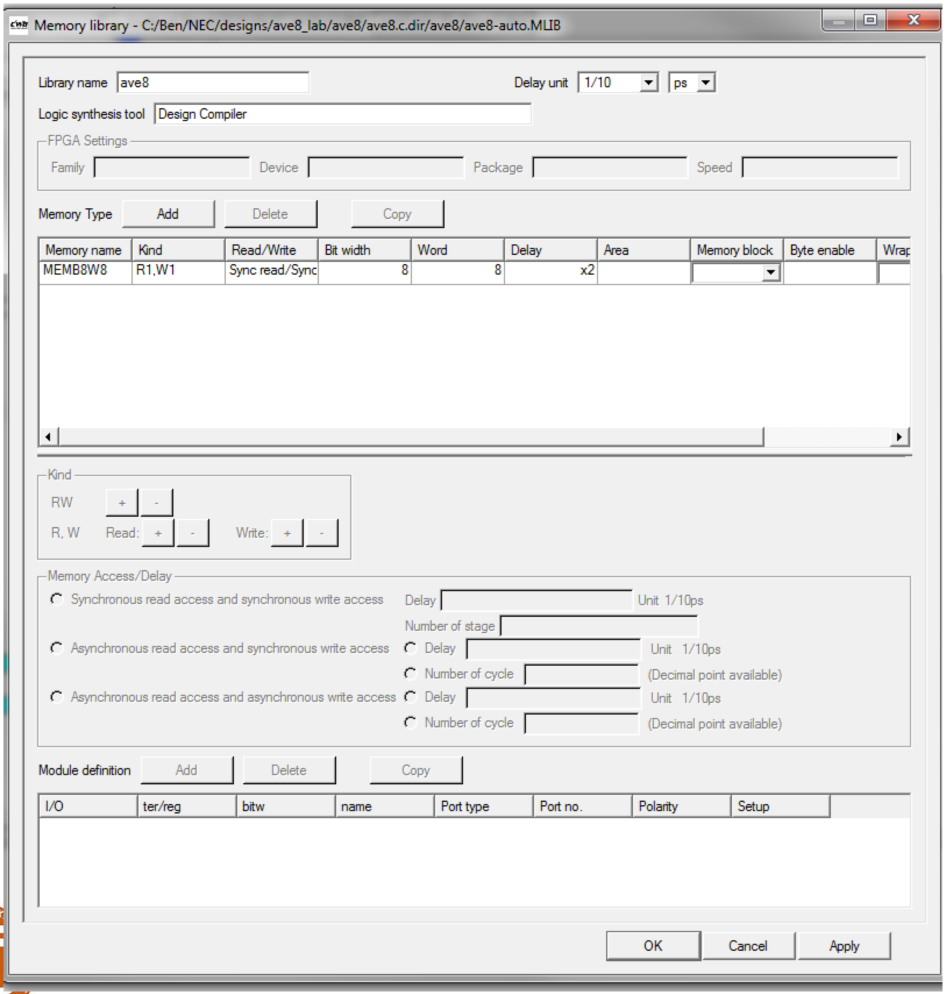
- MLIB and MCNT files
- Only generated when arrays are synthesized as memories (RAM or ROM)
- MLIB
  - Contains IOS and delay and area information that user needs to include
- MCNT: Memory Constraint File
  - Specifies how many memory modules to be instantiated



```
ave8.c.dir
  ave8
    ave8.IFF
    Libraries
      ave8-amacro-auto.FLIB
      ave8-auto.FLIB
      ave8-auto.MLIB
    Constraints
      ave8-amacro-auto.FCNT
      ave8-auto.FCNT
      ave8-auto.MCNT
```

# MLIB File

- IDE opens in dialog window
- Text editor in text format



Memory library - C:/Ben/NEC/designs/ave8_lab/ave8/ave8.c.dir/ave8/ave8-auto.MLIB

Library name: ave8
Delay unit: 1/10 ps
Logic synthesis tool: Design Compiler

FPGA Settings: Family / Device / Package / Speed

Memory Type: Add / Delete / Copy

| Memory name | Kind | Read/Write | Bit width | Word | Delay | Area | Memory block | Byte enable | Wrap |
|---|---|---|---|---|---|---|---|---|---|
| MEMB8W8 | R1,W1 | Sync read/Sync | 8 | 8 | x2 | | | | |

Kind: RW + - ; R, W Read: + - Write: + -

Memory Access/Delay
- Synchronous read access and synchronous write access — Delay ___ Unit 1/10ps; Number of stage ___
- Asynchronous read access and synchronous write access — Delay ___ Unit 1/10ps; Number of cycle ___ (Decimal point available)
- Asynchronous read access and asynchronous write access — Delay ___ Unit 1/10ps; Number of cycle ___ (Decimal point available)

Module definition: Add / Delete / Copy

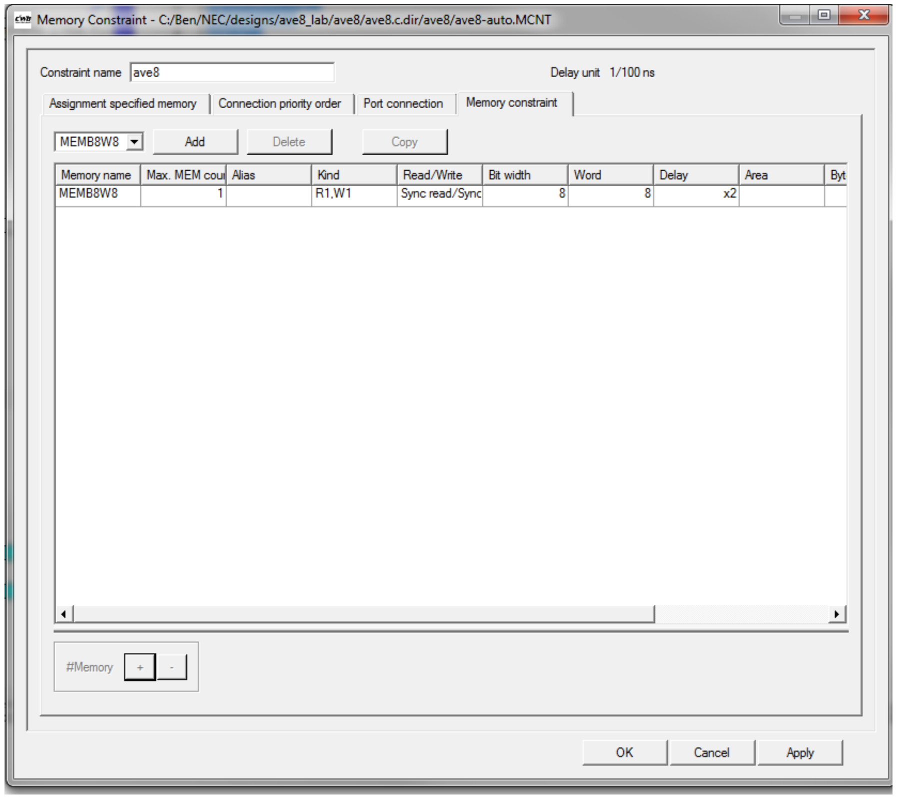
| I/O | ter/reg | bitw | name | Port type | Port no. | Polarity | Setup |
|---|---|---|---|---|---|---|---|

OK / Cancel / Apply

```
#@VERSION { 3.00 }
#@LIB { ave8 }
#@UNIT 1/10ps
#@SYN_TOOL{DC}
@MLIB {
    NAME        MEMB8W8
    KIND        R1,W1
    BITWIDTH    8
    DELAY       x2
    WORD        8
    WRITE_SYNC        YES
    DEFMOD {
        in      ter (0:3)    RA1       /* ra:1 */;
        out     ter (0:8)    RD1       /* rd:1 */;
        in      ter (0:1)    RE1       /* re:1 */;
        in      ter (0:1)    RCLK1     /* rclk:1 */;
        in      ter (0:3)    WA2       /* wa:2 */;
        in      ter (0:8)    WD2       /* wd:2 */;
        in      ter (0:1)    WE2       /* we:2 */;
        in      ter (0:1)    WCLK2     /* wclk:2 */;
    }
}
#@END { ave8 }
```

# MCNT file

- Memory Constraint File



Memory Constraint - C:/Ben/NEC/designs/ave8_lab/ave8/ave8.c.dir/ave8/ave8-auto.MCNT

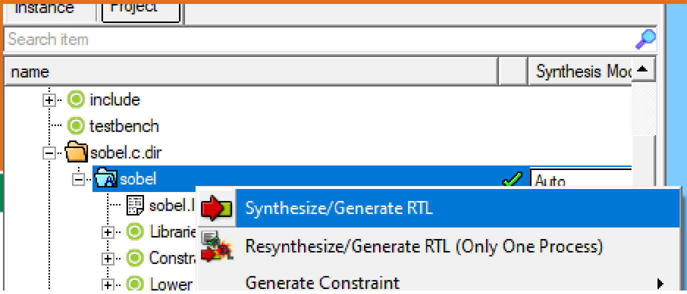
Constraint name ave8

Delay unit 1/100 ns

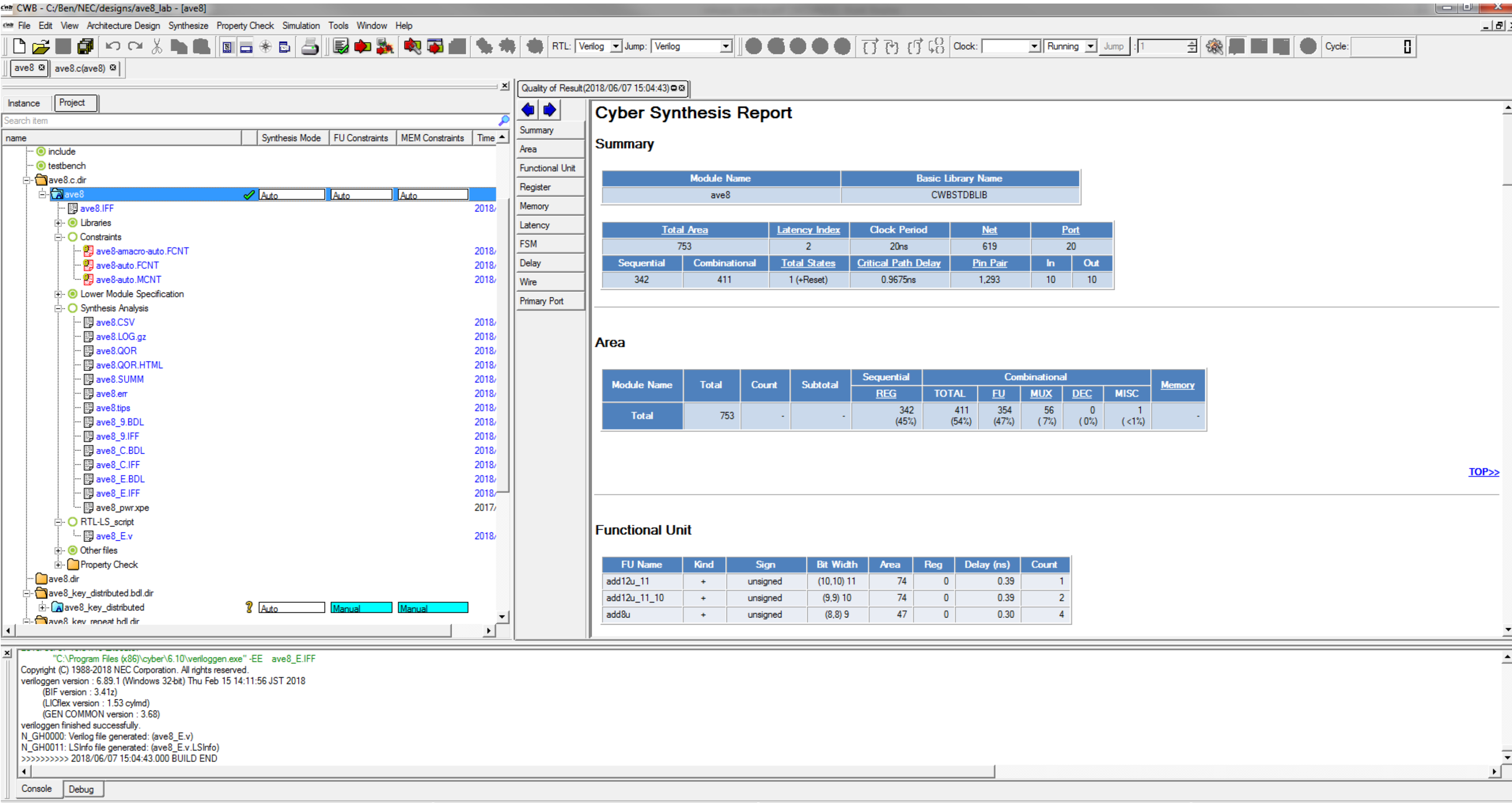
| Memory name | Max. MEM cou | Alias | Kind | Read/Write | Bit width | Word | Delay | Area | Byt |
|---|---|---|---|---|---|---|---|---|---|
| MEMB8W8 | 1 | | R1,W1 | Sync read/Sync | 8 | 8 | x2 | | |

```
#@VERSION { 3.00 }
#@CNT { ave8 }
@MCNT {
    NAME MEMB8W8
    LIMIT  1
}
#@END { ave8 }
```

# Synthesis



- Foo_E.v
- Multiple report files :.CSV, .QOR, .QOR.HTLM, .SUMM, .err, .tips



**Cyber Synthesis Report**

**Summary**

| Module Name | Basic Library Name |
|---|---|
| ave8 | CWBSTDBLIB |

| Total Area | | Latency Index | Clock Period | Net | Port | |
|---|---|---|---|---|---|---|
| 753 | | 2 | 20ns | 619 | 20 | |
| Sequential | Combinational | Total States | Critical Path Delay | Pin Pair | In | Out |
| 342 | 411 | 1 (+Reset) | 0.9675ns | 1,293 | 10 | 10 |

**Area**

| Module Name | Total | Count | Subtotal | Sequential | Combinational | | | | | Memory |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | REG | TOTAL | FU | MUX | DEC | MISC | |
| Total | 753 | - | - | 342 (45%) | 411 (54%) | 354 (47%) | 56 ( 7%) | 0 ( 0%) | 1 ( <1%) | - |

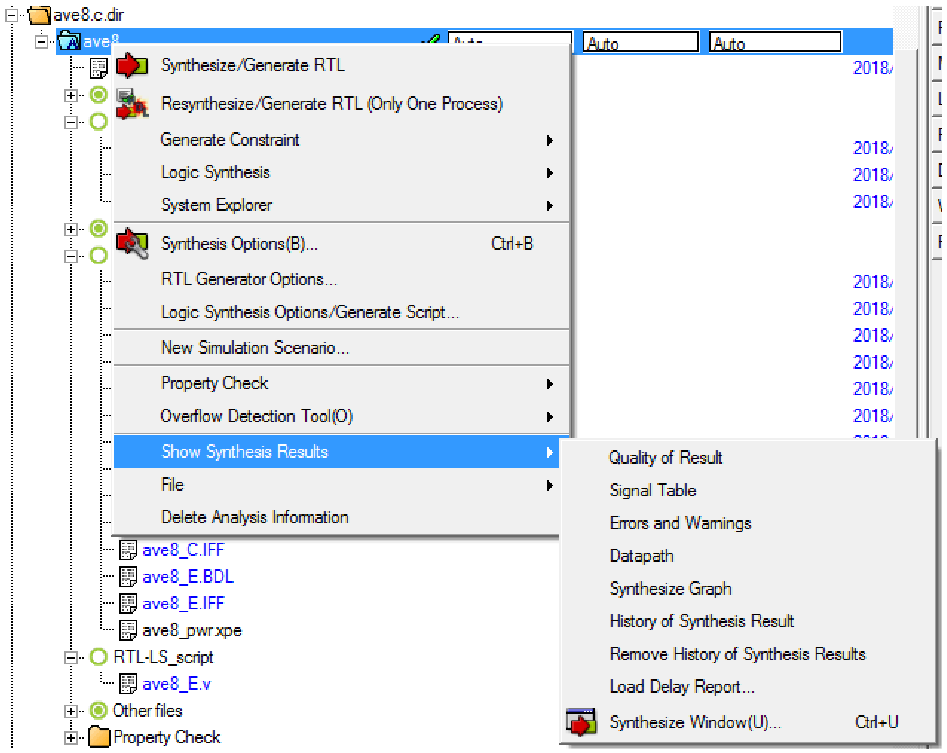
**Functional Unit**

| FU Name | Kind | Sign | Bit Width | Area | Reg | Delay (ns) | Count |
|---|---|---|---|---|---|---|---|
| add12u_11 | + | unsigned | (10,10) 11 | 74 | 0 | 0.39 | 1 |
| add12u_11_10 | + | unsigned | (9,9) 10 | 74 | 0 | 0.39 | 2 |
| add8u | + | unsigned | (8,8) 9 | 47 | 0 | 0.30 | 4 |

```
        "C:\Program Files (x86)\cyber\6.10\veriloggen.exe" -EE   ave8_E.IFF
Copyright (C) 1988-2018 NEC Corporation. All rights reserved.
veriloggen version : 6.89.1 (Windows 32-bit) Thu Feb 15 14:11:56 JST 2018
        (BIF version : 3.41z)
        (LICflex version : 1.53 cylmd)
        (GEN COMMON version : 3.68)
veriloggen finished successfully.
N_GH0000: Verilog file generated: (ave8_E.v)
N_GH0011: LSInfo file generated: (ave8_E.v.LSInfo)
>>>>>>>>>> 2018/06/07 15:04:43.000 BUILD END
```

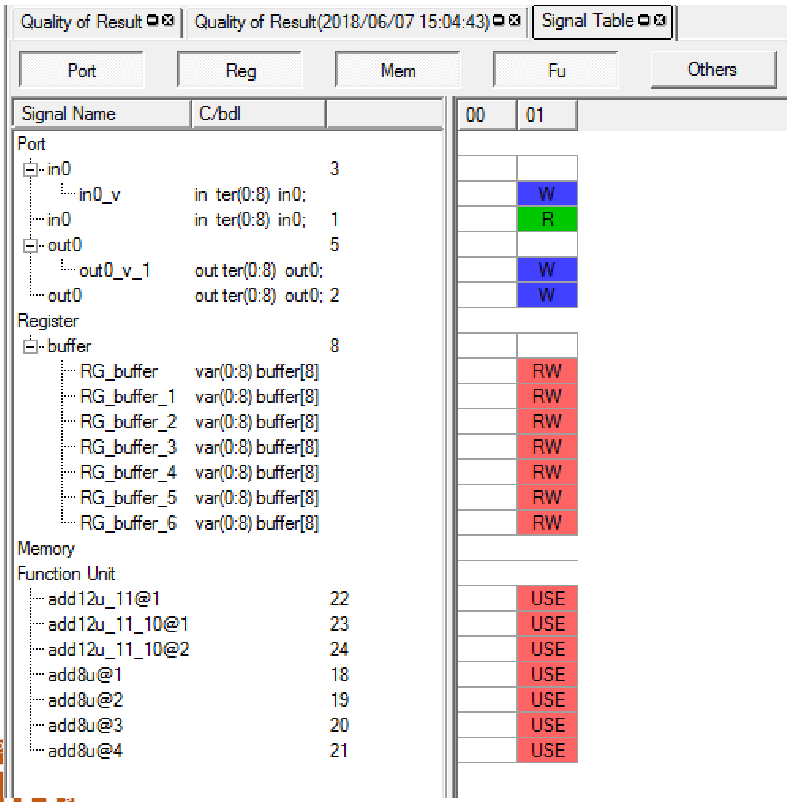
# Interpreting the Results

- QOR report
- Signal Table
- Errors and Warnings
- Datapath
- Synthesis Graph
- History of Synthesis Results

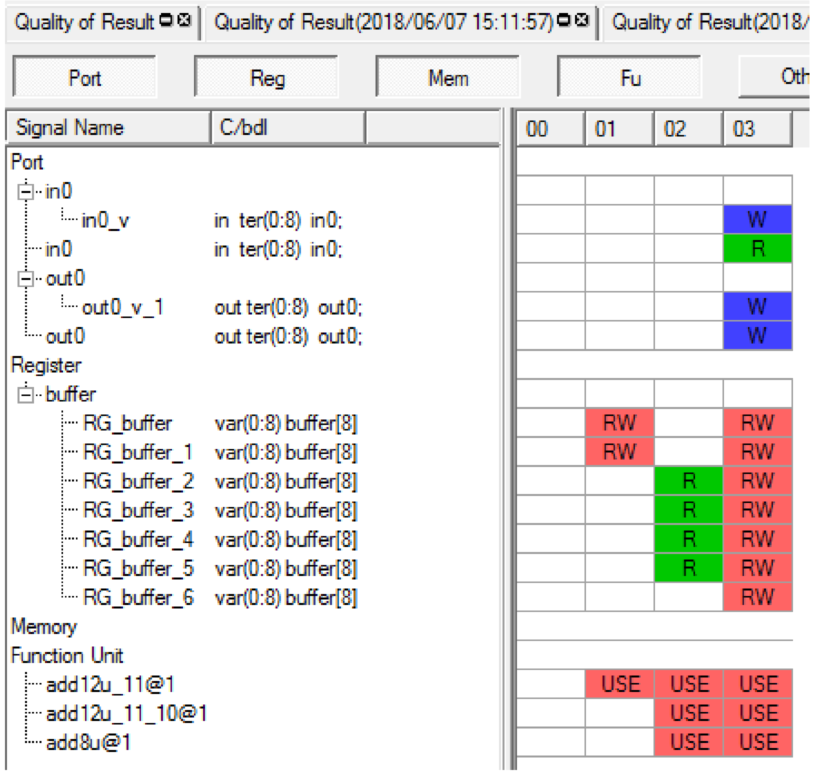


# Signal Table

- Timing information of:
  - IOs
  - Registers
  - Memories
  - FUs
- Clicking on cells jumps to C code



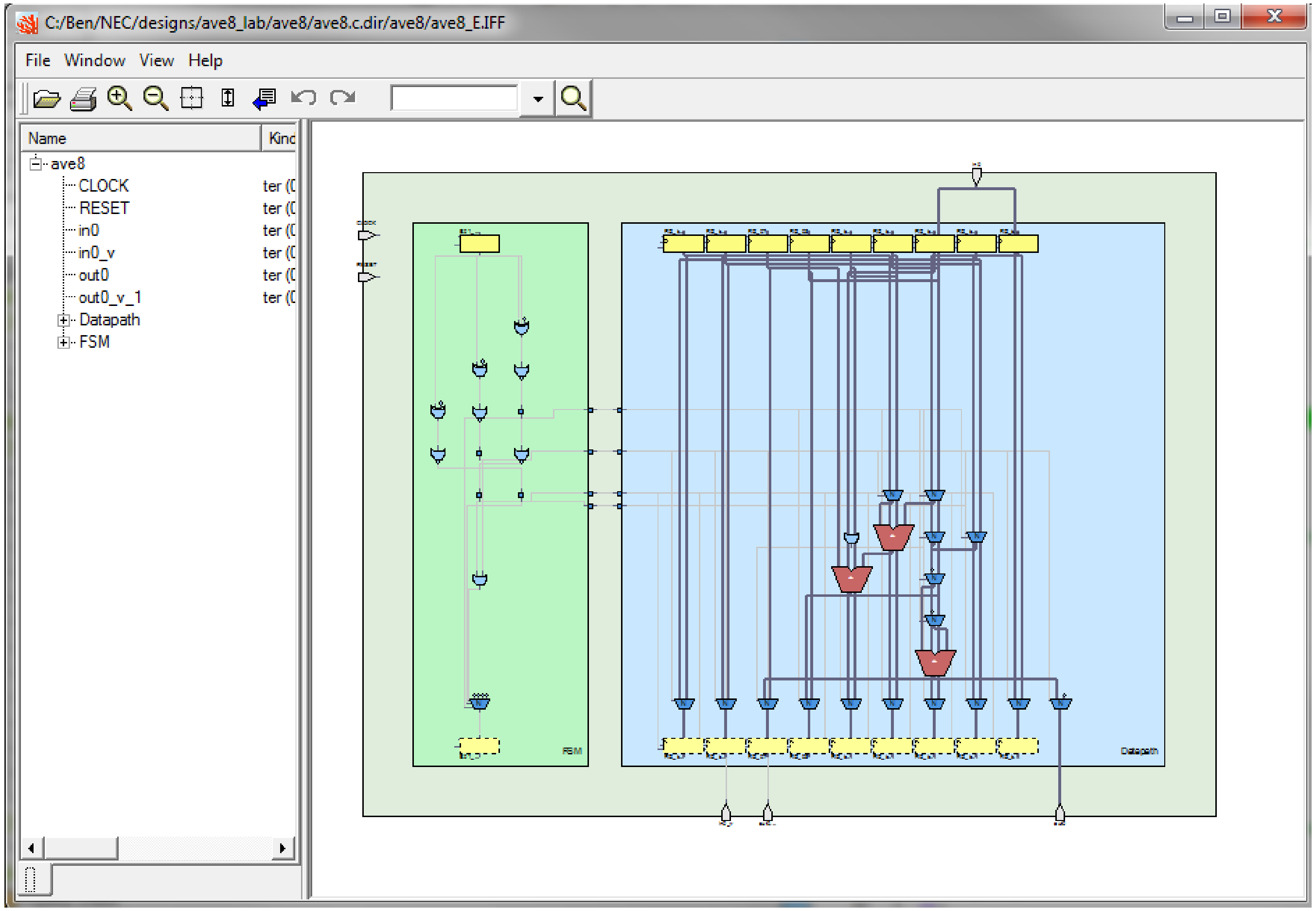
| Signal Name | C/bdl | | 00 | 01 |
|---|---|---|---|---|
| Port | | | | |
| in0 | | 3 | | |
| in0_v | in ter(0:8) in0; | | | W |
| in0 | in ter(0:8) in0; | 1 | | R |
| out0 | | 5 | | |
| out0_v_1 | out ter(0:8) out0; | | | W |
| out0 | out ter(0:8) out0; | 2 | | W |
| Register | | | | |
| buffer | | 8 | | |
| RG_buffer | var(0:8) buffer[8] | | | RW |
| RG_buffer_1 | var(0:8) buffer[8] | | | RW |
| RG_buffer_2 | var(0:8) buffer[8] | | | RW |
| RG_buffer_3 | var(0:8) buffer[8] | | | RW |
| RG_buffer_4 | var(0:8) buffer[8] | | | RW |
| RG_buffer_5 | var(0:8) buffer[8] | | | RW |
| RG_buffer_6 | var(0:8) buffer[8] | | | RW |
| Memory | | | | |
| Function Unit | | | | |
| add12u_11@1 | | 22 | | USE |
| add12u_11_10@1 | | 23 | | USE |
| add12u_11_10@2 | | 24 | | USE |
| add8u@1 | | 18 | | USE |
| add8u@2 | | 19 | | USE |
| add8u@3 | | 20 | | USE |
| add8u@4 | | 21 | | USE |



| Signal Name | C/bdl | | 00 | 01 | 02 | 03 |
|---|---|---|---|---|---|---|
| Port | | | | | | |
| in0 | | | | | | |
| in0_v | in ter(0:8) in0; | | | | | W |
| in0 | in ter(0:8) in0; | | | | | R |
| out0 | | | | | | |
| out0_v_1 | out ter(0:8) out0; | | | | | W |
| out0 | out ter(0:8) out0; | | | | | W |
| Register | | | | | | |
| buffer | | | | | | |
| RG_buffer | var(0:8) buffer[8] | | | RW | | RW |
| RG_buffer_1 | var(0:8) buffer[8] | | | RW | | RW |
| RG_buffer_2 | var(0:8) buffer[8] | | | | R | RW |
| RG_buffer_3 | var(0:8) buffer[8] | | | | R | RW |
| RG_buffer_4 | var(0:8) buffer[8] | | | | R | RW |
| RG_buffer_5 | var(0:8) buffer[8] | | | | R | RW |
| RG_buffer_6 | var(0:8) buffer[8] | | | | | RW |
| Memory | | | | | | |
| Function Unit | | | | | | |
| add12u_11@1 | | | | USE | USE | USE |
| add12u_11_10@1 | | | | | USE | USE |
| add8u@1 | | | | | USE | USE |

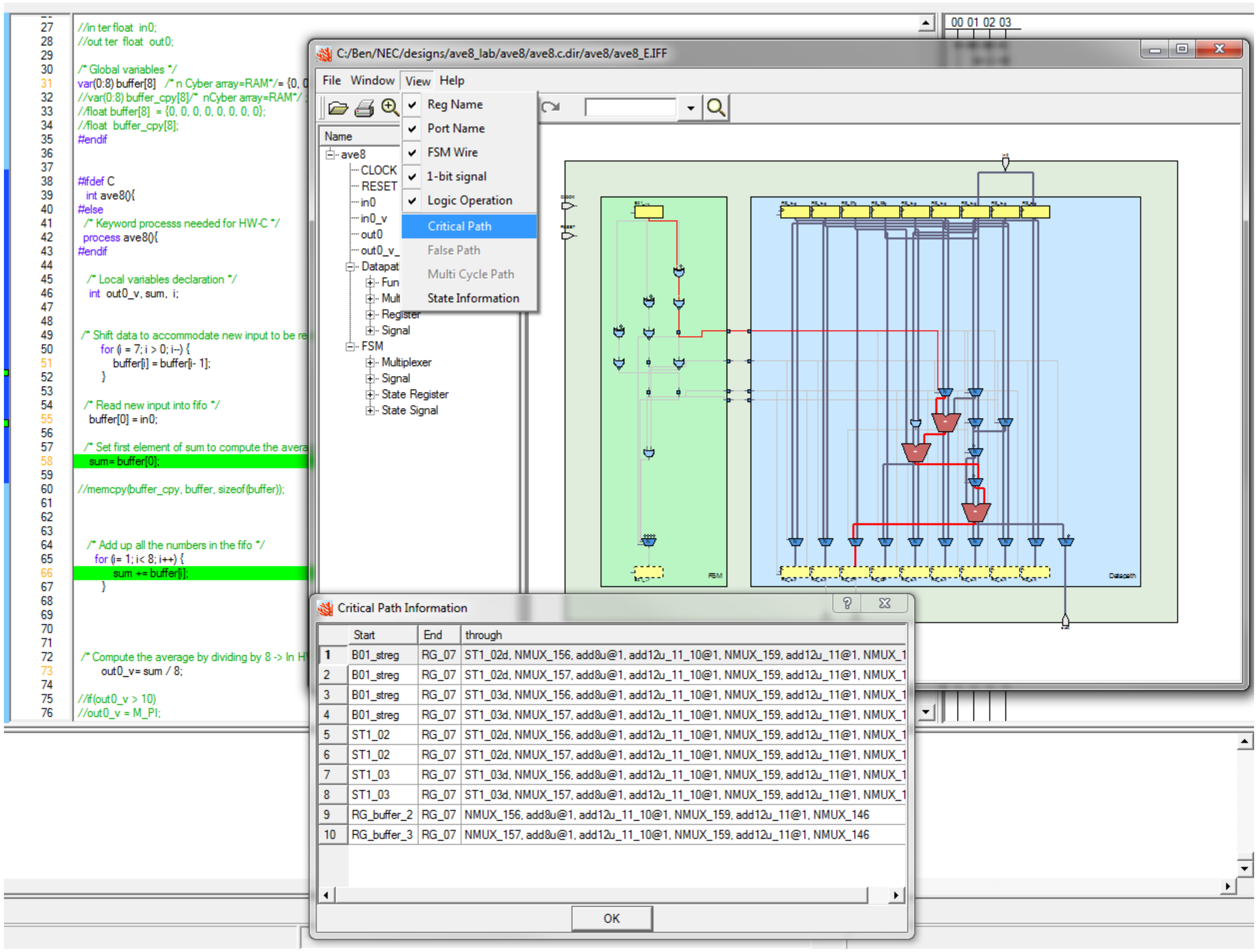
# Datapath

- Graphical representation of resultant circuit
- Pathdraw (%pathdraw foo_E.IFF)



# Datapath – Critical Paths

- Shows critical paths on diagram and C source code

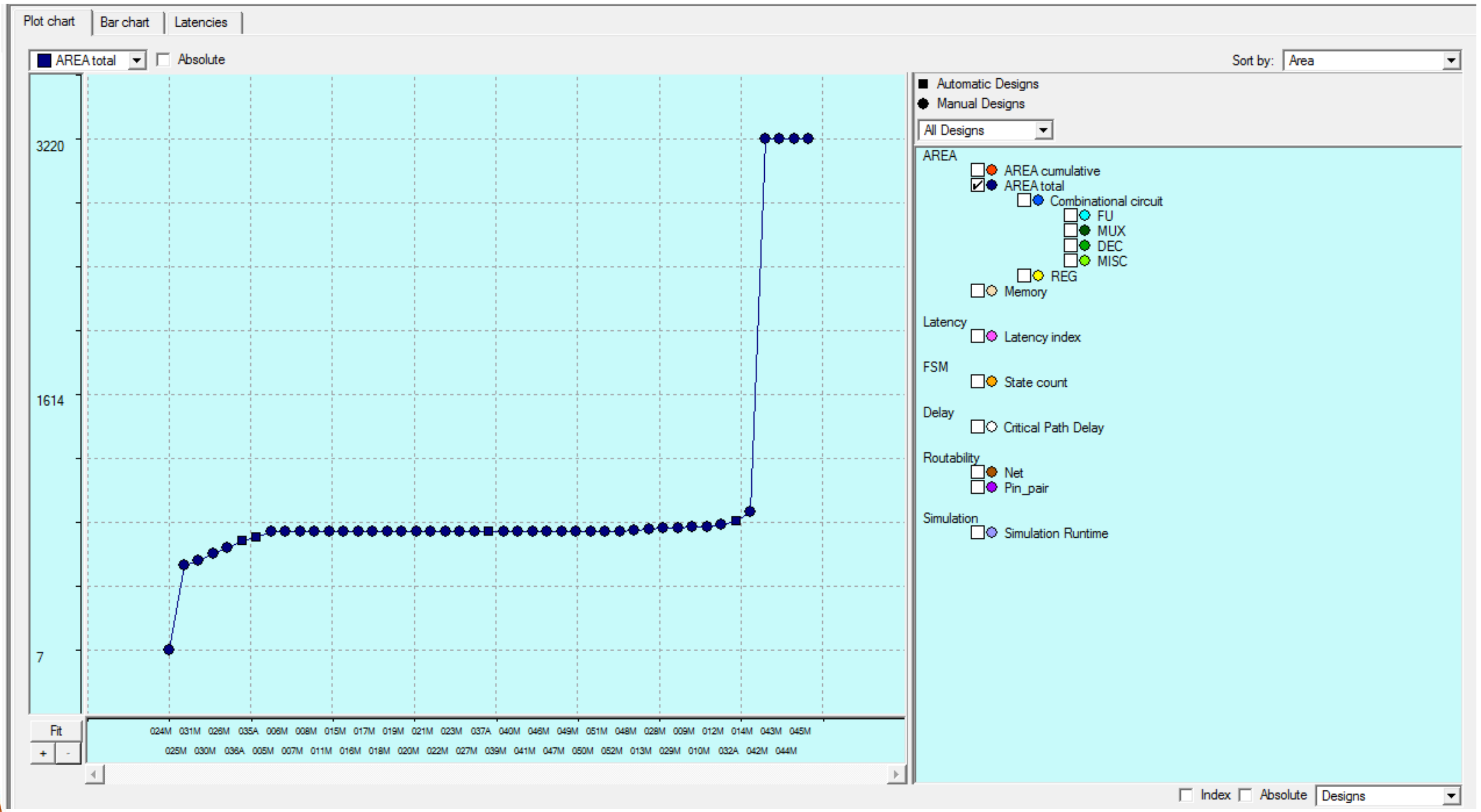


```
//in ter float  in0;
//out ter  float  out0;

/* Global variables */
var(0:8) buffer[8]   /* n Cyber array=RAM*/= {0, 0
//var(0:8) buffer_cpy[8]/*  nCyber array=RAM*/ ;
//float buffer[8]  = {0, 0, 0, 0, 0, 0, 0, 0};
//float  buffer_cpy[8];
#endif


#ifdef C
  int ave8(){
#else
  /* Keyword processs needed for HW-C */
  process ave8(){
#endif

   /* Local variables declaration */
   int  out0_v, sum,  i;


 /* Shift data to accommodate new input to be re
       for (i = 7; i > 0; i--) {
          buffer[i] = buffer[i- 1];
       }

  /* Read new input into fifo */
   buffer[0] = in0;

  /* Set first element of sum to compute the avera
   sum= buffer[0];

//memcpy(buffer_cpy, buffer, sizeof(buffer));



   /* Add up all the numbers in the fifo */
     for (i= 1; i< 8; i++) {
          sum += buffer[i];
       }




 /* Compute the average by dividing by 8 -> In H
       out0_v= sum / 8;

//if(out0_v > 10)
//out0_v = M_PI;
```

| | Start | End | through |
|---|---|---|---|
| 1 | B01_streg | RG_07 | ST1_02d, NMUX_156, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 2 | B01_streg | RG_07 | ST1_02d, NMUX_157, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 3 | B01_streg | RG_07 | ST1_03d, NMUX_156, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 4 | B01_streg | RG_07 | ST1_03d, NMUX_157, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 5 | ST1_02 | RG_07 | ST1_02d, NMUX_156, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 6 | ST1_02 | RG_07 | ST1_02d, NMUX_157, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 7 | ST1_03 | RG_07 | ST1_03d, NMUX_156, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 8 | ST1_03 | RG_07 | ST1_03d, NMUX_157, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_1 |
| 9 | RG_buffer_2 | RG_07 | NMUX_156, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_146 |
| 10 | RG_buffer_3 | RG_07 | NMUX_157, add8u@1, add12u_11_10@1, NMUX_159, add12u_11@1, NMUX_146 |

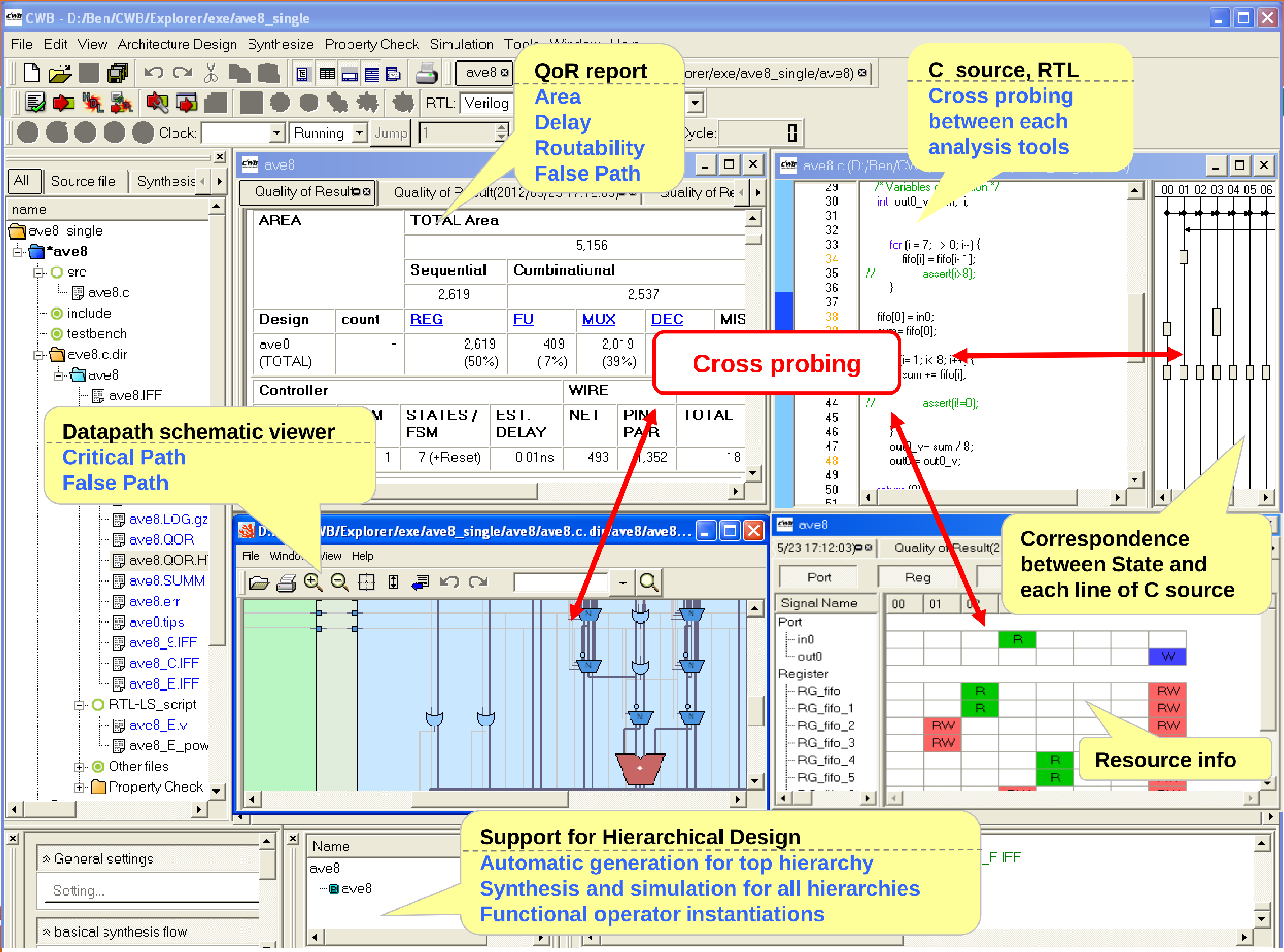
# Synthesis Graph

- Keeps record of all design combinations generated and displays in Plot chart



Sort by: Area

☐ AREA cumulative
✓ AREA total
☐ Combinational circuit
☐ FU
☐ MUX
☐ DEC
☐ MISC
☐ REG
☐ Memory
☐ Latency index
☐ State count
☐ Critical Path Delay
☐ Net
☐ Pin_pair
☐ Simulation Runtime



QoR report
**Area**
**Delay**
**Routability**
**False Path**

C source, RTL
**Cross probing between each analysis tools**

Datapath schematic viewer
**Critical Path**
**False Path**

Correspondence between State and each line of C source

Resource info

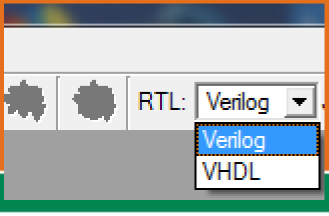
Support for Hierarchical Design
**Automatic generation for top hierarchy**
**Synthesis and simulation for all hierarchies**
**Functional operator instantiations**

| AREA | TOTAL Area | | | | |
|---|---|---|---|---|---|
| | 5,156 | | | | |
| | Sequential | Combinational | | | |
| | 2,619 | 2,537 | | | |
| Design | count | REG | FU | MUX | DEC |
| ave8 (TOTAL) | - | 2,619 (50%) | 409 (7%) | 2,019 (39%) | |

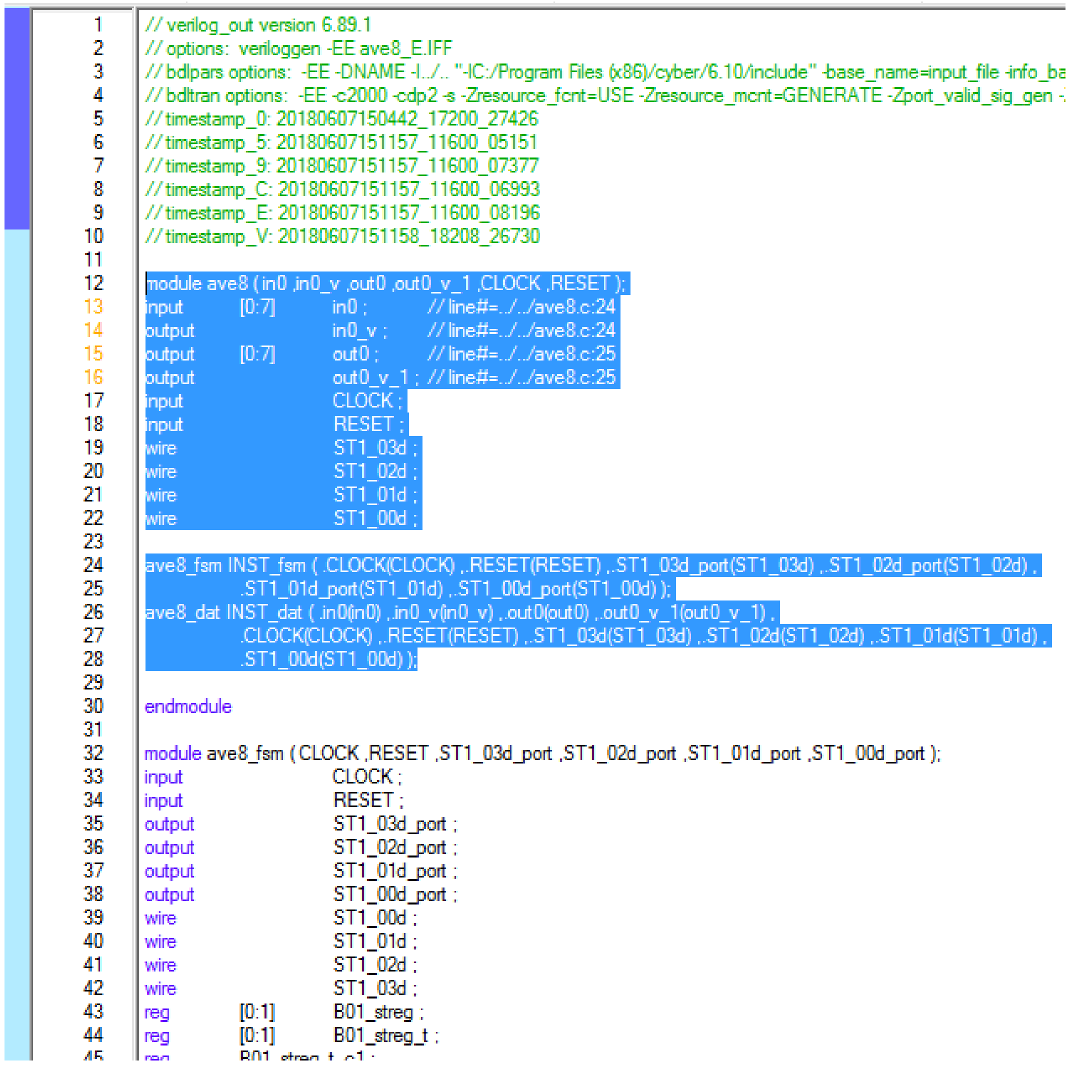
| Controller | | | WIRE | | |
|---|---|---|---|---|---|
| M | STATES / FSM | EST. DELAY | NET | PIN PAIR | TOTAL |
| 1 | 7 (+Reset) | 0.01ns | 493 | 1,352 | 18 |

```
/* Variables ... */
int out0_v ... i;

for (i = 7; i > 0; i--) {
    fifo[i] = fifo[i- 1];
//      assert(i>8);
}

fifo[0] = in0;
... = fifo[0];
... i= 1; i< 8; i++) {
    sum += fifo[i];
//      assert(i!=0);
}
out0_v= sum / 8;
out0 = out0_v;
```

# RTL Code Generation



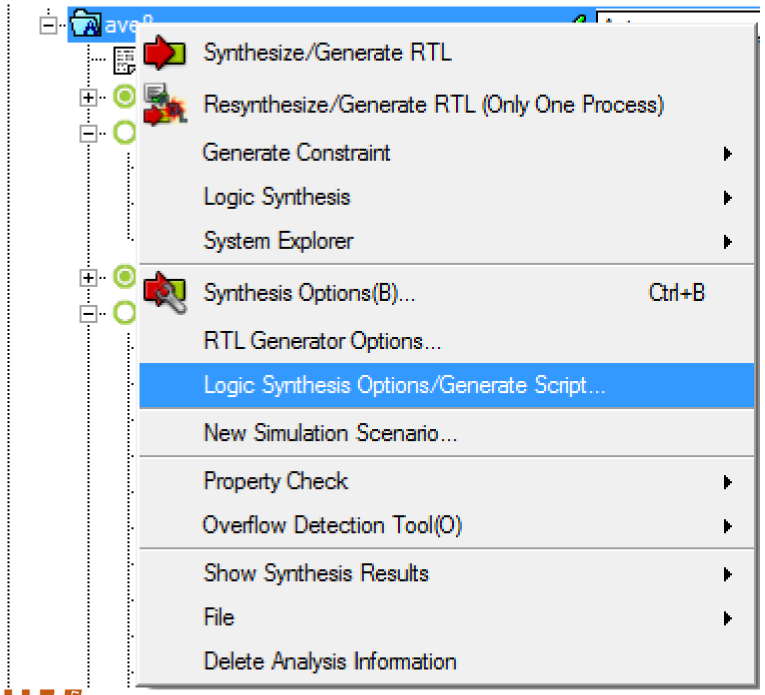
- RTL
  - Verilog (veriloggen)
  - VHDL (vhdlgen)
- Structural RTL
  - Module
    - Foo_fsm (state machine)
    - Foo_dat (data path)
- Allows cross probing
  - Click on RTL and jumps to original C code (only orange lines numbers)

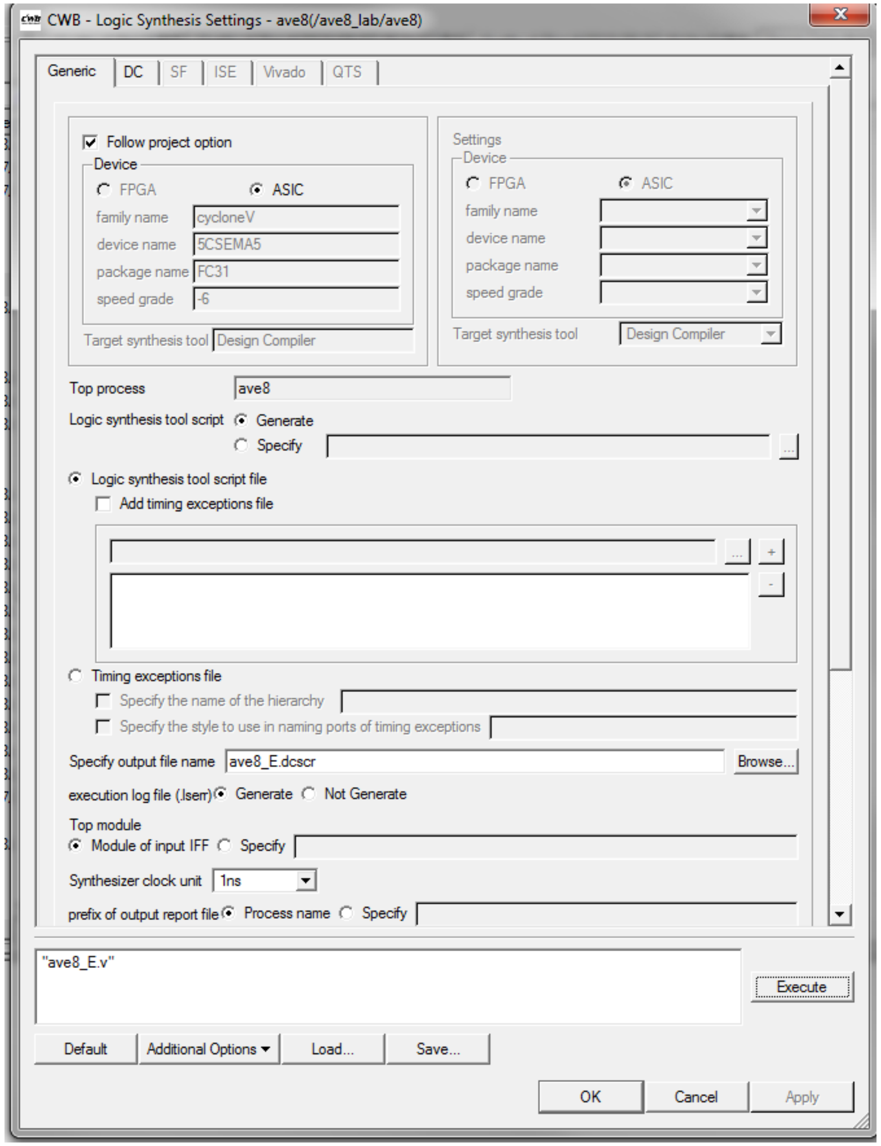


```
 1  // verilog_out version 6.89.1
 2  // options:  veriloggen -EE ave8_E.IFF
 3  // bdlpars options:  -EE -DNAME -I../.. "-IC:/Program Files (x86)/cyber/6.10/include" -base_name=input_file -info_ba
 4  // bdltran options:  -EE -c2000 -cdp2 -s -Zresource_fcnt=USE -Zresource_mcnt=GENERATE -Zport_valid_sig_gen -
 5  // timestamp_0: 20180607150442_17200_27426
 6  // timestamp_5: 20180607151157_11600_05151
 7  // timestamp_9: 20180607151157_11600_07377
 8  // timestamp_C: 20180607151157_11600_06993
 9  // timestamp_E: 20180607151157_11600_08196
10  // timestamp_V: 20180607151158_18208_26730
11
12  module ave8 ( in0 ,in0_v ,out0 ,out0_v_1 ,CLOCK ,RESET );
13  input       [0:7]       in0 ;       // line#=../../ave8.c:24
14  output                  in0_v ;     // line#=../../ave8.c:24
15  output      [0:7]       out0 ;      // line#=../../ave8.c:25
16  output                  out0_v_1 ;  // line#=../../ave8.c:25
17  input                   CLOCK ;
18  input                   RESET ;
19  wire                    ST1_03d ;
20  wire                    ST1_02d ;
21  wire                    ST1_01d ;
22  wire                    ST1_00d ;
23
24  ave8_fsm INST_fsm ( .CLOCK(CLOCK) ,.RESET(RESET) ,.ST1_03d_port(ST1_03d) ,.ST1_02d_port(ST1_02d) ,
25              .ST1_01d_port(ST1_01d) ,.ST1_00d_port(ST1_00d) );
26  ave8_dat INST_dat ( .in0(in0) ,.in0_v(in0_v) ,.out0(out0) ,.out0_v_1(out0_v_1) ,
27              .CLOCK(CLOCK) ,.RESET(RESET) ,.ST1_03d(ST1_03d) ,.ST1_02d(ST1_02d) ,.ST1_01d(ST1_01d) ,
28              .ST1_00d(ST1_00d) );
29
30  endmodule
31
32  module ave8_fsm ( CLOCK ,RESET ,ST1_03d_port ,ST1_02d_port ,ST1_01d_port ,ST1_00d_port );
33  input                   CLOCK ;
34  input                   RESET ;
35  output                  ST1_03d_port ;
36  output                  ST1_02d_port ;
37  output                  ST1_01d_port ;
38  output                  ST1_00d_port ;
39  wire                    ST1_00d ;
40  wire                    ST1_01d ;
41  wire                    ST1_02d ;
42  wire                    ST1_03d ;
43  reg         [0:1]       B01_streg ;
44  reg         [0:1]       B01_streg_t ;
45  reg         B01_streg_t_c1 ;
```

# Interfacing with Logic Synthesis Tools

- Logic synthesis script generator
  - Xilinx ISE/Vivado
  - Intel Quartus
  - Synopsys DC


ave8
Synthesize/Generate RTL
Resynthesize/Generate RTL (Only One Process)
Generate Constraint
Logic Synthesis
System Explorer
Synthesis Options(B)... Ctrl+B
RTL Generator Options...
Logic Synthesis Options/Generate Script...
New Simulation Scenario...
Property Check
Overflow Detection Tool(O)
Show Synthesis Results
File
Delete Analysis Information


CWB - Logic Synthesis Settings - ave8(/ave8_lab/ave8)
Generic | DC | SF | ISE | Vivado | QTS
Default
Additional Options
Load...
Save...
Execute
OK
Cancel
Apply
Browse...

[✓] Follow project option

Device: FPGA [✓ ASIC]

| | |
|---|---|
| family name | cycloneV |
| device name | 5CSEMA5 |
| package name | FC31 |
| speed grade | -6 |
| Target synthesis tool | Design Compiler |

Settings

Device: FPGA [✓ ASIC]

| | |
|---|---|
| family name | |
| device name | |
| package name | |
| speed grade | |
| Target synthesis tool | Design Compiler |

Top process: ave8

Logic synthesis tool script: [✓ Generate] Specify

[✓] Logic synthesis tool script file

[ ] Add timing exceptions file

( ) Timing exceptions file

[ ] Specify the name of the hierarchy

[ ] Specify the style to use in naming ports of timing exceptions

Specify output file name: ave8_E.dcscr

execution log file (.lsen): [✓ Generate] Not Generate

Top module: [✓ Module of input IFF] Specify

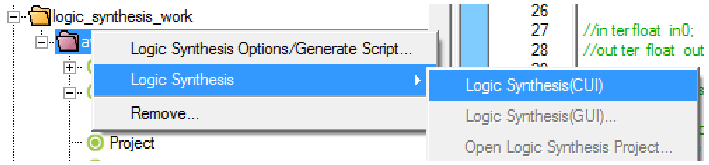
Synthesizer clock unit: 1ns

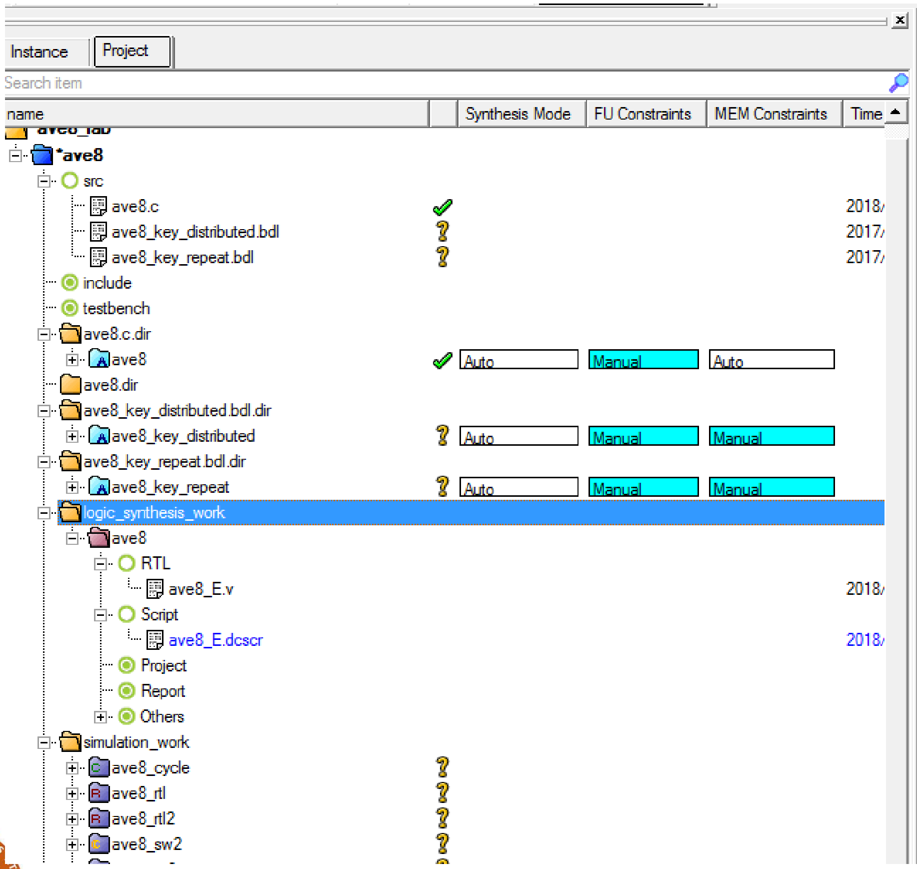
prefix of output report file: [✓ Process name] Specify

"ave8_E.v"

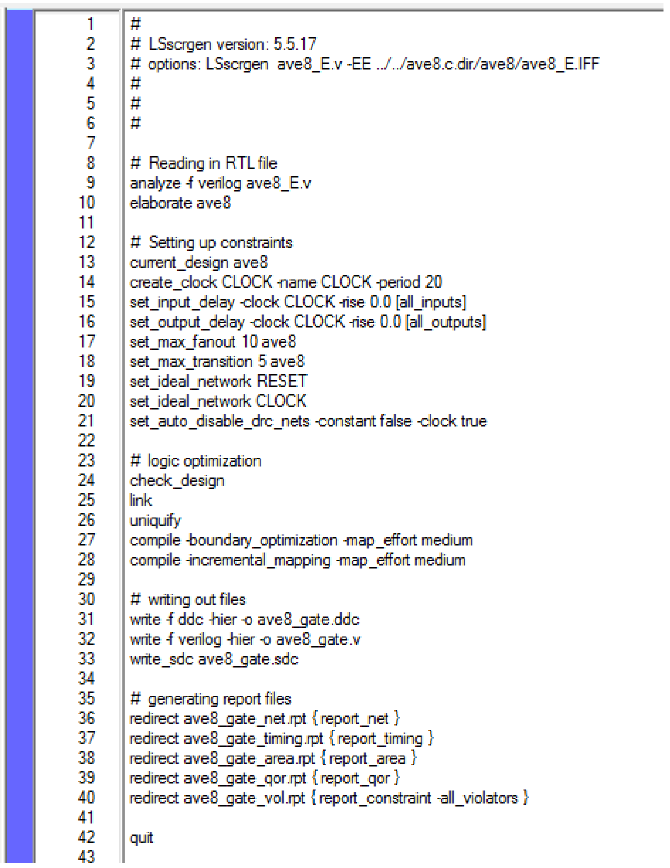
# ASIC: Design Compiler Example

- Logic Synthesis folder generated (red folder)
- Shell script for Synopsys DC generated
- Simply call:
  dc_shell –f foo_e.dcsr
  Or from IDE (right click on red folder → logic synthesis (CUI)





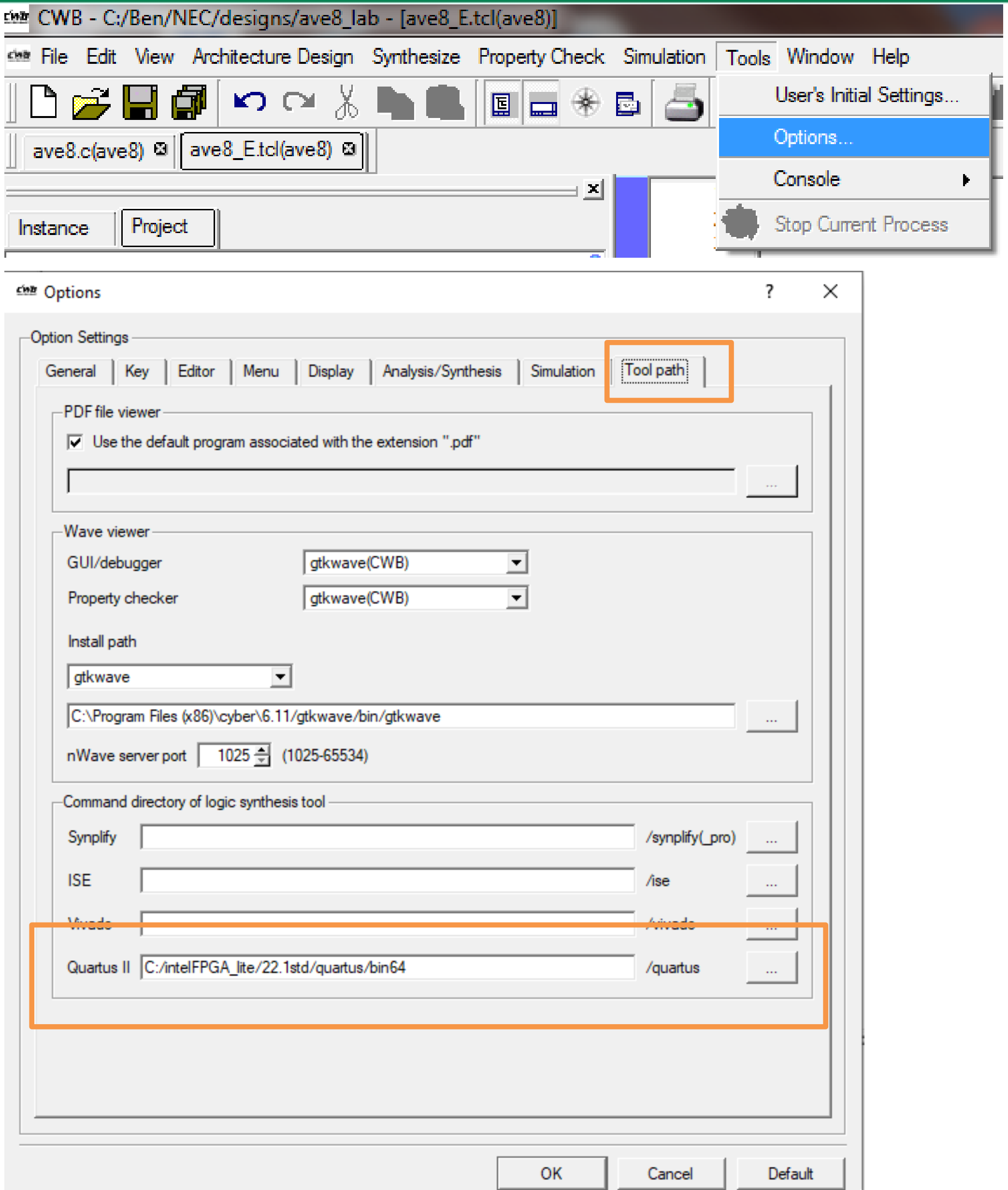
| name | Synthesis Mode | FU Constraints | MEM Constraints | Time |
|---|---|---|---|---|
| ave8_lab | | | | |
| *ave8 | | | | |
| src | | | | |
| ave8.c ✓ | | | | 2018/ |
| ave8_key_distributed.bdl ? | | | | 2017/ |
| ave8_key_repeat.bdl ? | | | | 2017/ |
| include | | | | |
| testbench | | | | |
| ave8.c.dir | | | | |
| ave8 ✓ | Auto | Manual | Auto | |
| ave8.dir | | | | |
| ave8_key_distributed.bdl.dir | | | | |
| ave8_key_distributed ? | Auto | Manual | Manual | |
| ave8_key_repeat.bdl.dir | | | | |
| ave8_key_repeat ? | Auto | Manual | Manual | |
| logic_synthesis_work | | | | |
| ave8 | | | | |
| RTL | | | | |
| ave8_E.v | | | | 2018/ |
| Script | | | | |
| ave8_E.dcscr | | | | 2018/ |
| Project | | | | |
| Report | | | | |
| Others | | | | |
| simulation_work | | | | |
| ave8_cycle ? | | | | |
| ave8_rtl ? | | | | |
| ave8_rtl2 ? | | | | |
| ave8_sw2 ? | | | | |



```
#
# LSscrgen version: 5.5.17
# options: LSscrgen ave8_E.v -EE ../../ave8.c.dir/ave8/ave8_E.IFF
#
#
#

# Reading in RTL file
analyze -f verilog ave8_E.v
elaborate ave8

# Setting up constraints
current_design ave8
create_clock CLOCK -name CLOCK -period 20
set_input_delay -clock CLOCK -rise 0.0 [all_inputs]
set_output_delay -clock CLOCK -rise 0.0 [all_outputs]
set_max_fanout 10 ave8
set_max_transition 5 ave8
set_ideal_network RESET
set_ideal_network CLOCK
set_auto_disable_drc_nets -constant false -clock true

# logic optimization
check_design
link
uniquify
compile -boundary_optimization -map_effort medium
compile -incremental_mapping -map_effort medium

# writing out files
write -f ddc -hier -o ave8_gate.ddc
write -f verilog -hier -o ave8_gate.v
write_sdc ave8_gate.sdc

# generating report files
redirect ave8_gate_net.rpt { report_net }
redirect ave8_gate_timing.rpt { report_timing }
redirect ave8_gate_area.rpt { report_area }
redirect ave8_gate_qor.rpt { report_qor }
redirect ave8_gate_vol.rpt { report_constraint -all_violators }

quit
```

# FPGA: External Tools

- Add Quartus to CWB's path: Tools→ Options →Tool Path



PDF file viewer

[✓] Use the default program associated with the extension ".pdf"

Wave viewer

GUI/debugger: gtkwave(CWB)

Property checker: gtkwave(CWB)

Install path: gtkwave

C:\Program Files (x86)\cyber\6.11/gtkwave/bin/gtkwave

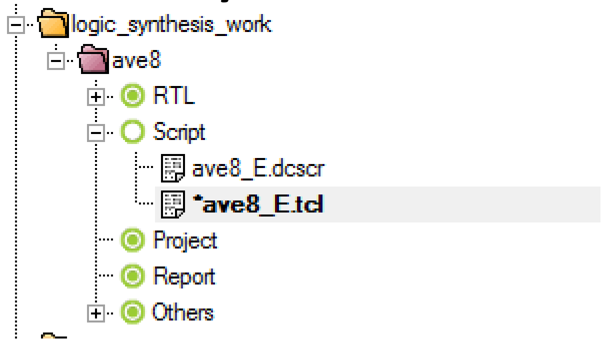
nWave server port: 1025 (1025-65534)

Command directory of logic synthesis tool

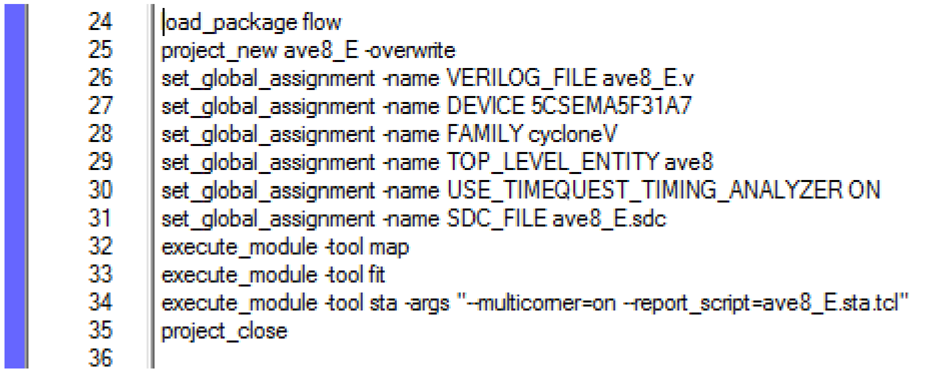
| | | |
|---|---|---|
| Synplify | | /synplify(_pro) |
| ISE | | /ise |
| Vivado | | /vivado |
| Quartus II | C:/intelFPGA_lite/22.1std/quartus/bin64 | /quartus |

# Intel Quartus II Example

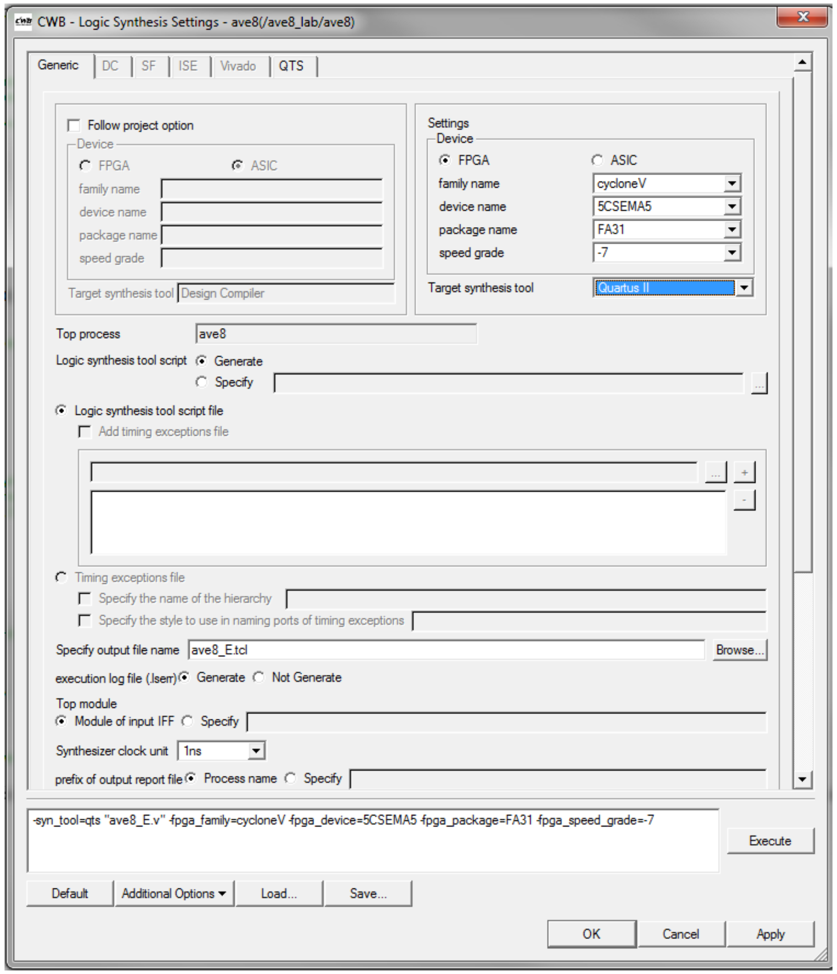
- Select Cyclone V FPGA (or follow project option)
- Set synthesis tool to Quartus



```
logic_synthesis_work
  ave8
    RTL
    Script
      ave8_E.dcscr
      *ave8_E.tcl
    Project
    Report
    Others
```



```
load_package flow
project_new ave8_E -overwrite
set_global_assignment -name VERILOG_FILE ave8_E.v
set_global_assignment -name DEVICE 5CSEMA5F31A7
set_global_assignment -name FAMILY cycloneV
set_global_assignment -name TOP_LEVEL_ENTITY ave8
set_global_assignment -name USE_TIMEQUEST_TIMING_ANALYZER ON
set_global_assignment -name SDC_FILE ave8_E.sdc
execute_module -tool map
execute_module -tool fit
execute_module -tool sta -args "--multicorner=on --report_script=ave8_E.sta.tcl"
project_close
```



CWB - Logic Synthesis Settings - ave8(/ave8_lab/ave8)

Generic | DC | SF | ISE | Vivado | QTS

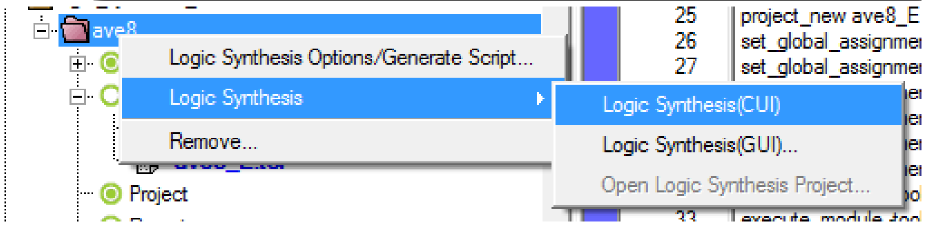
Follow project option [ ]
Device: FPGA [✓ ASIC]
family name:
device name:
package name:
speed grade:
Target synthesis tool: Design Compiler

Settings
Device: [✓ FPGA] ASIC
family name: cycloneV
device name: 5CSEMA5
package name: FA31
speed grade: -7
Target synthesis tool: Quartus II

Top process: ave8
Logic synthesis tool script: [✓ Generate] Specify
[✓ Logic synthesis tool script file]
Add timing exceptions file [ ]
Timing exceptions file
Specify the name of the hierarchy [ ]
Specify the style to use in naming ports of timing exceptions [ ]
Specify output file name: ave8_E.tcl (Browse...)
execution log file (.lserr): [✓ Generate] Not Generate
Top module: [✓ Module of input IFF] Specify
Synthesizer clock unit: 1ns
prefix of output report file: [✓ Process name] Specify

```
-syn_tool=qts "ave8_E.v" -fpga_family=cycloneV -fpga_device=5CSEMA5 -fpga_package=FA31 -fpga_speed_grade=-7
```

Execute | Default | Additional Options | Load... | Save... | OK | Cancel | Apply



ave8
- Logic Synthesis Options/Generate Script...
- Logic Synthesis
  - Logic Synthesis(CUI)
  - Logic Synthesis(GUI)...
  - Open Logic Synthesis Project...
- Remove...

# Adding your own Constraints

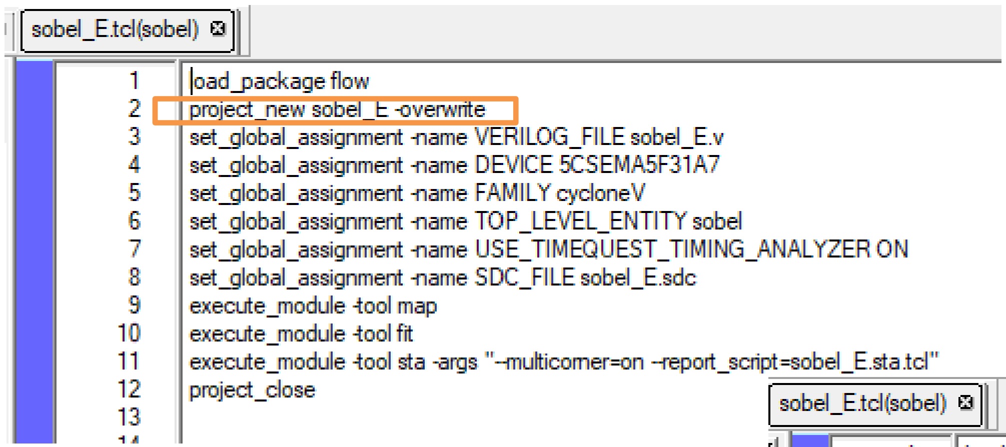
1. Edit foo_E.qsf file generated after running Quartus once. E.g., add pin constraints

   set_location_assignment PIN_AF14 -to CLOCK
   set_location_assignment PIN_AA14 -to RESET

2. Modify Logic synthesis script file:

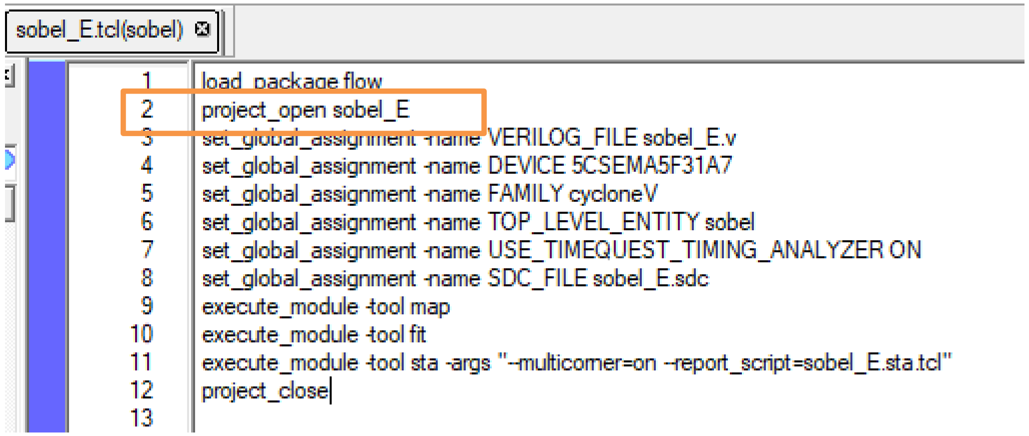
   project_new → project_open

1. Call Quartus again by clicking on foo_E.tcl file



sobel_E.tcl(sobel)

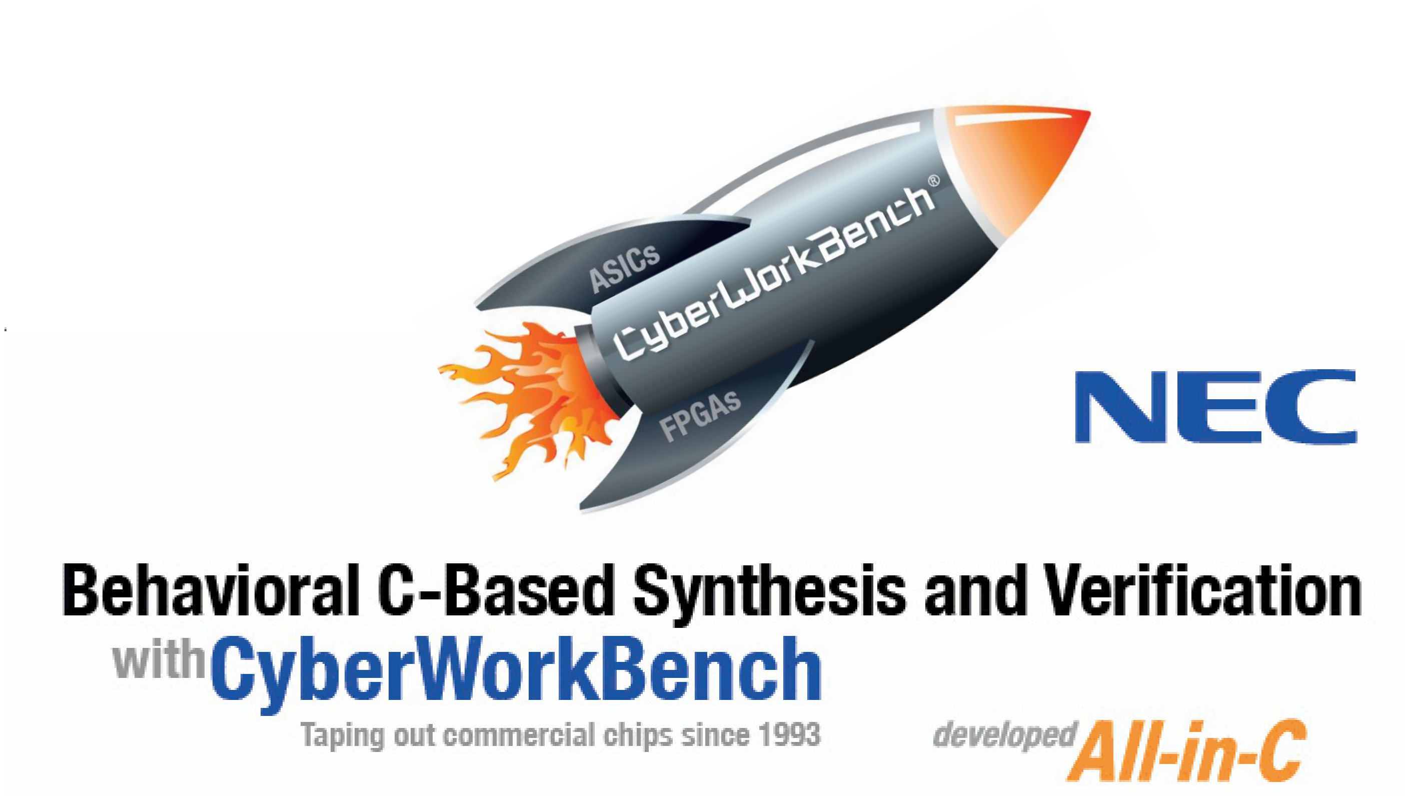
```
load_package flow
project_new sobel_E -overwrite
set_global_assignment -name VERILOG_FILE sobel_E.v
set_global_assignment -name DEVICE 5CSEMA5F31A7
set_global_assignment -name FAMILY cycloneV
set_global_assignment -name TOP_LEVEL_ENTITY sobel
set_global_assignment -name USE_TIMEQUEST_TIMING_ANALYZER ON
set_global_assignment -name SDC_FILE sobel_E.sdc
execute_module -tool map
execute_module -tool fit
execute_module -tool sta -args "--multicorner=on --report_script=sobel_E.sta.tcl"
project_close
```



sobel_E.tcl(sobel)

```
load_package flow
project_open sobel_E
set_global_assignment -name VERILOG_FILE sobel_E.v
set_global_assignment -name DEVICE 5CSEMA5F31A7
set_global_assignment -name FAMILY cycloneV
set_global_assignment -name TOP_LEVEL_ENTITY sobel
set_global_assignment -name USE_TIMEQUEST_TIMING_ANALYZER ON
set_global_assignment -name SDC_FILE sobel_E.sdc
execute_module -tool map
execute_module -tool fit
execute_module -tool sta -args "--multicorner=on --report_script=sobel_E.sta.tcl"
project_close
```

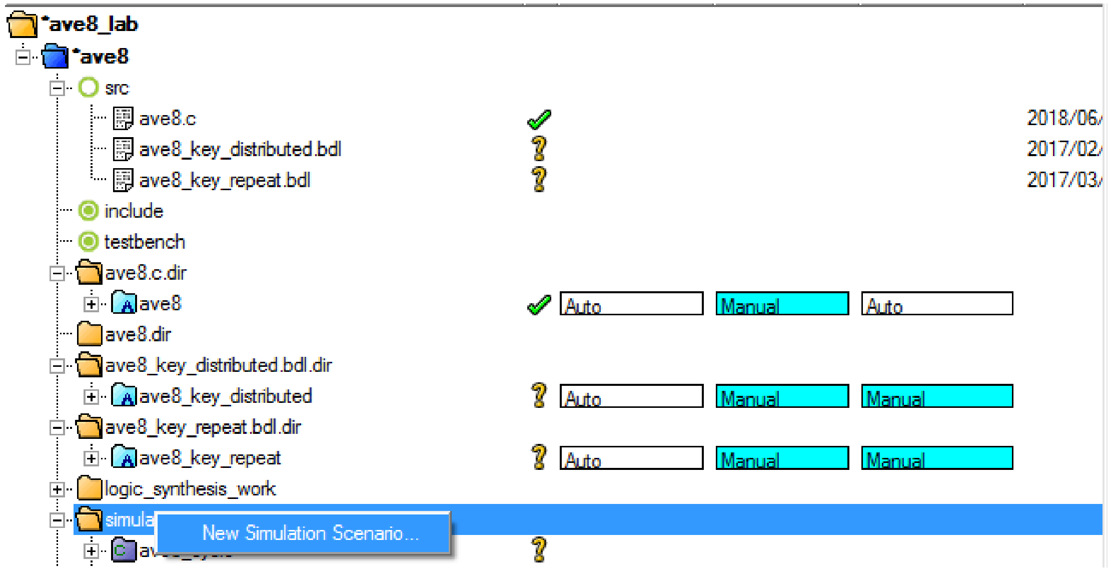
# NEC - CyberWorkBench


ASICs
CyberWorkBench®
FPGAs
NEC
Behavioral C-Based Synthesis and Verification
with CyberWorkBench
Taping out commercial chips since 1993
developed All-in-C

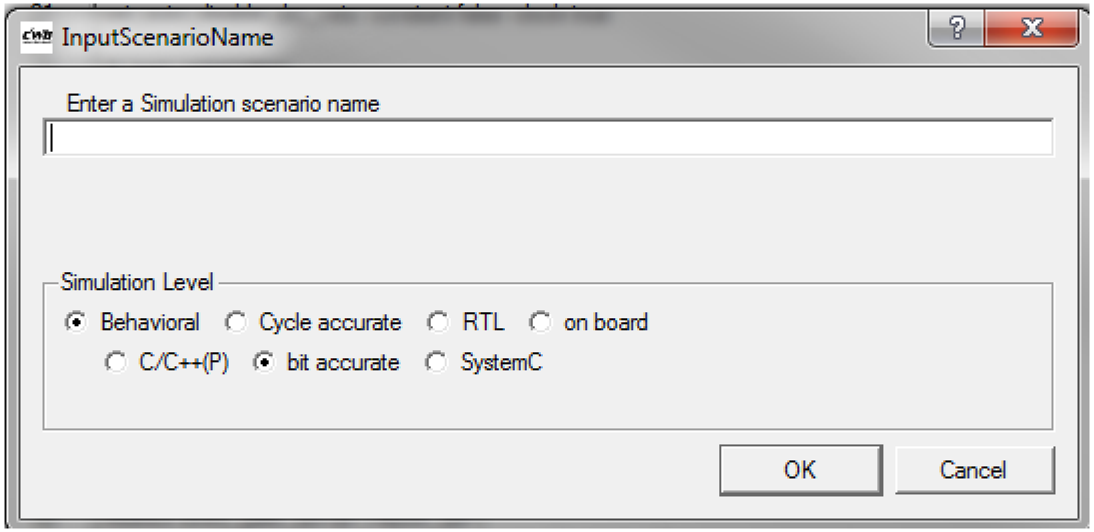
[Video]

# Verification with CWB

- Untimed:
  - Original C compilation
  - Data type simulation
- Timed
  - Cycle-accurate model generator
    - cmscgen
  - RTL testbench generator
    - tbgen



```
*ave8_lab
  *ave8
    src
      ave8.c                          ✓                                    2018/06/
      ave8_key_distributed.bdl        ?                                    2017/02/
      ave8_key_repeat.bdl             ?                                    2017/03/
    include
    testbench
    ave8.c.dir
      ave8                            ✓  Auto    Manual    Auto
    ave8.dir
    ave8_key_distributed.bdl.dir
      ave8_key_distributed            ?  Auto    Manual    Manual
    ave8_key_repeat.bdl.dir
      ave8_key_repeat                 ?  Auto    Manual    Manual
    logic_synthesis_work
    simula
      av                              ?
```



InputScenarioName

Enter a Simulation scenario name

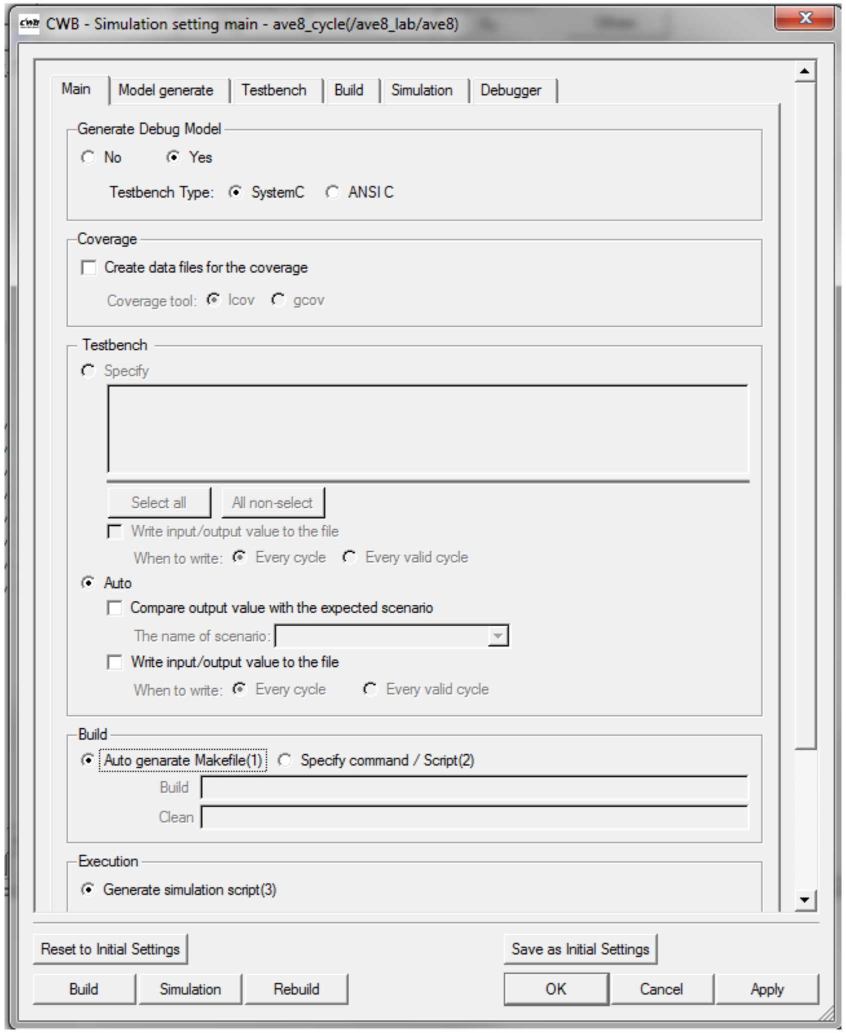
Simulation Level

[✓ Behavioral] Cycle accurate RTL on board

C/C++(P) [✓ bit accurate] SystemC

# Cycle-Accurate Simulation

- Generates SystemC model that replicates the behavior of the Verilog code cycle-accurately
- Takes as input result of HLS scheduling phase (_C.IFF)
- Benefits:
  - Is compilable using g++ (no RTL simulator is needed)
  - Faster than RTL simulation
- Test-vectors
  - Cycle-level vectors (.clv files)
  - Transaction-level vectors (.tlv files)
  - Random
  - One file required for each port
- Unselect : compare output value with expected scenario to enable all options


CWB - Simulation setting main - ave8_cycle(/ave8_lab/ave8)

Generate Debug Model
No [✓ Yes]
Testbench Type: [✓ SystemC] ANSI C

Coverage
☐ Create data files for the coverage
Coverage tool: [✓ lcov] gcov

Testbench
Specify

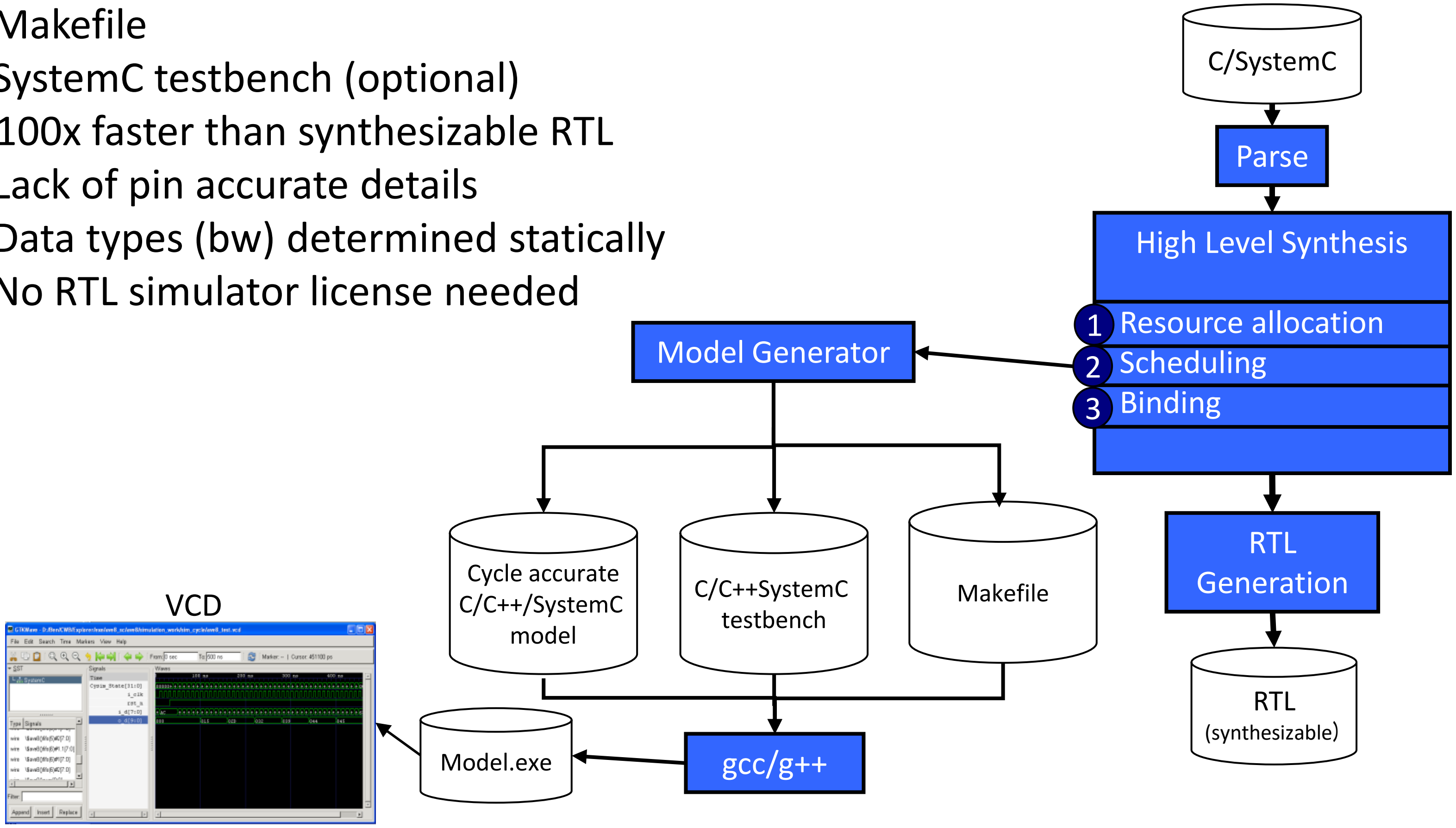
☐ Write input/output value to the file
When to write: [✓ Every cycle] Every valid cycle
[✓ Auto]
☐ Compare output value with the expected scenario
The name of scenario:
☐ Write input/output value to the file
When to write: [✓ Every cycle] Every valid cycle

Build
[✓ Auto genarate Makefile(1)] Specify command / Script(2)
Build
Clean

Execution
[✓ Generate simulation script(3)]

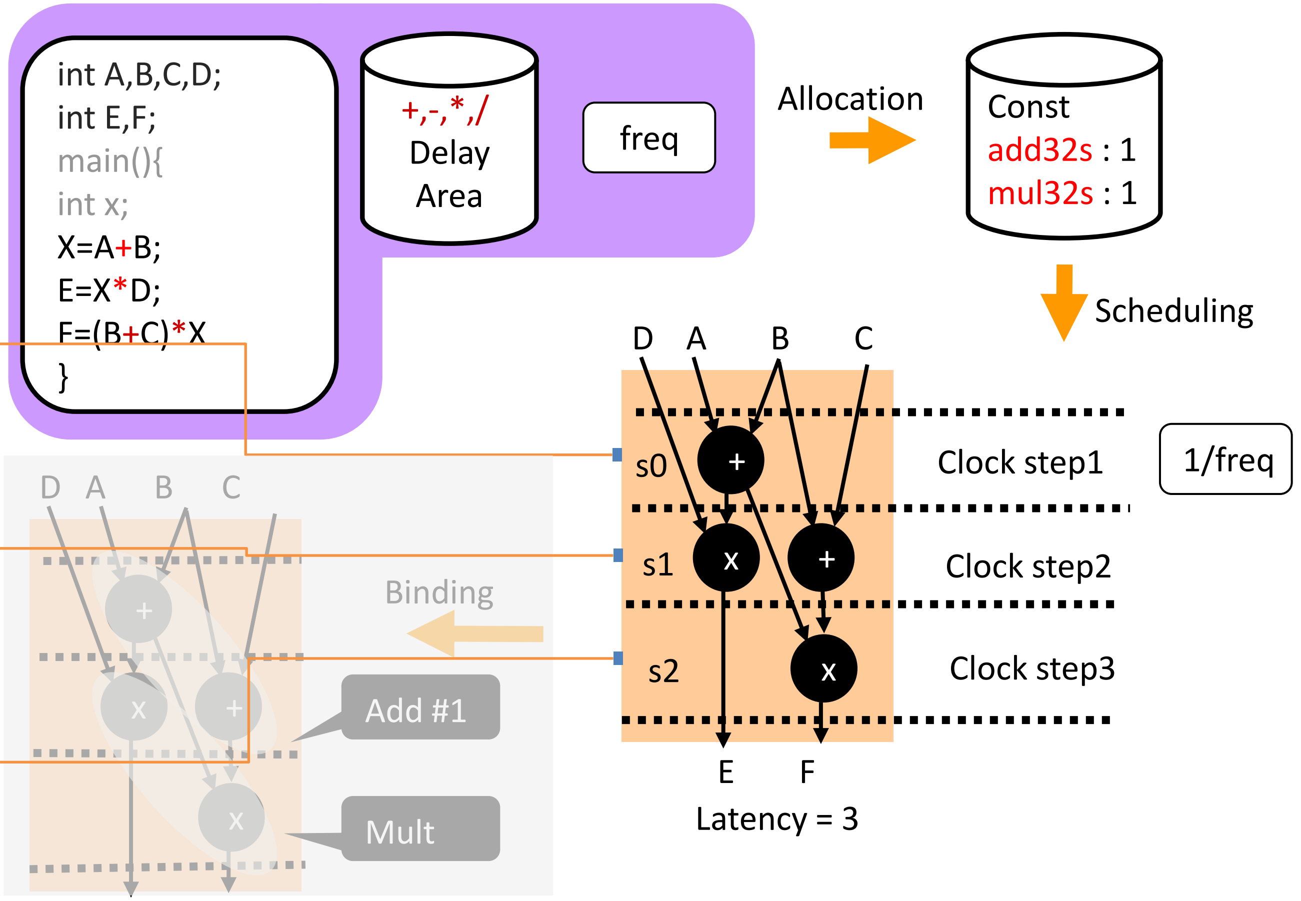
# Cycle-Accurate Model Overview

- Input to model generator
  - Scheduled CDFG (before binding stage) (_C.IFF)
- Outputs
  - C/C++SystemC cycle accurate model
  - Makefile
  - SystemC testbench (optional)
- ~10-100x faster than synthesizable RTL
  - Lack of pin accurate details
  - Data types (bw) determined statically
  - No RTL simulator license needed



# Cycle-Accurate Models

```
// Execute every clock cycle
switch(state){
  case reset:
      tmp1=tmp2=tmp3=0;
      next_state=s0;
      break;
   case s0:
      tmp1=A+B;
      next_state =s1;
     break;
  case s1:
     tmp2 = tmp1*d;
     tmp3 = B+C;
     E= tmp2;          // output1
      next_state=s2;
     break;
  case s2:
     F= tmp1*tmp3; // output2
     next_state=s0;
     break;
}
```



## CWB – Testbench Generator

- CWB comes with an automatic testbench generator
- The testbench reads untimed inputs/outputs from a text file with extension .clv or .tlv
- Each IO should have a file with its name and .clv or .tlv extension
- clv vs. tlv files
  – Clv: Cycle-level vectors
  – Tlv : Transaction-level vectors

## CLV/TLV files

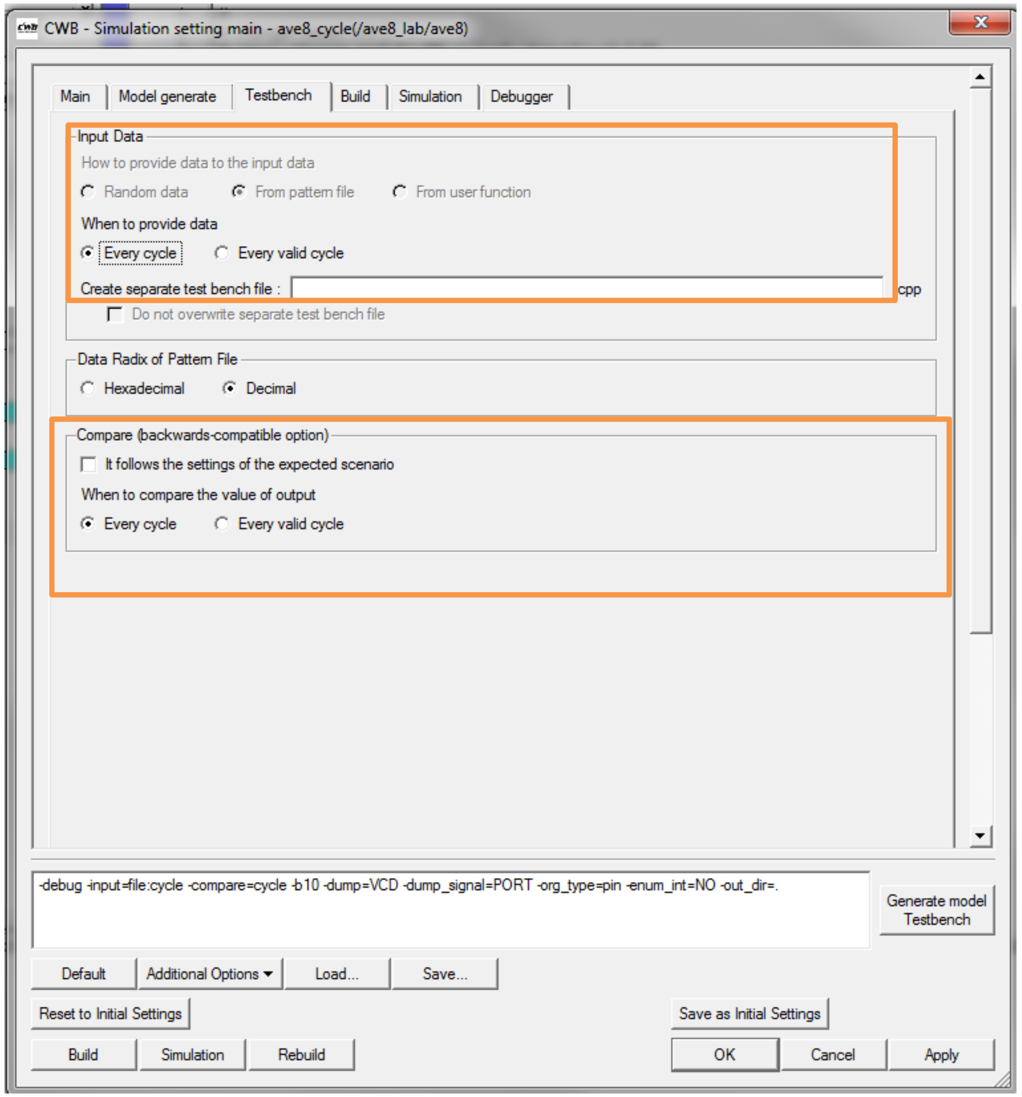
- Every IO has to have a .clv or .tlv file
  - <io_name>.clv .e.g.
  - in ter(7..0) a; → a.clv or a.tlv
- In case that IO is an array. E.g.
  - in ter(0:16) data[8];
  - data_a00.clv, data_a001.clv, etc..

a.clv./tlv

```
3
4
5
1
14
65
34
```

# Differences between CLV and TLV files

- The file is **EXACTLY the same**
- In the case of clv file → Testbench reads EVERY clock cycle and passes the value to the UUT
- In the case of tlv file → Testbench reads a value only when needed. How does the TB know when a new value is needed?
- Pay attention to the **data radix**


CWB - Simulation setting main - ave8_cycle(/ave8_lab/ave8)

Main | Model generate | Testbench | Build | Simulation | Debugger

Input Data
How to provide data to the input data
Random data [✓ From pattern file] From user function
When to provide data
[✓ Every cycle] Every valid cycle
Create separate test bench file : .cpp
Do not overwrite separate test bench file

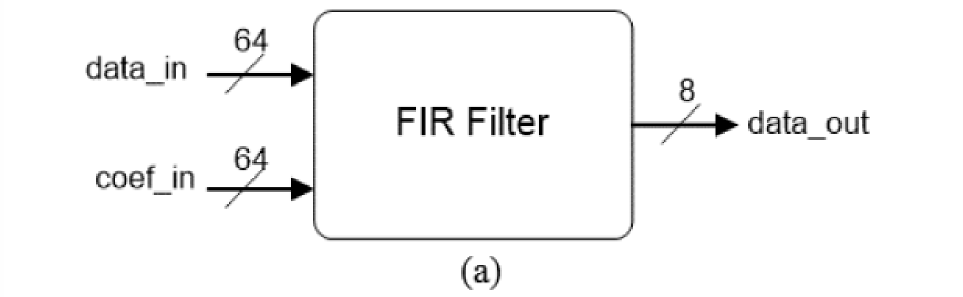
Data Radix of Pattern File
Hexadecimal [✓ Decimal]

Compare (backwards-compatible option)
It follows the settings of the expected scenario
When to compare the value of output
[✓ Every cycle] Every valid cycle

```
-debug -input=file:cycle -compare=cycle -b10 -dump=VCD -dump_signal=PORT -org_type=pin -enum_int=NO -out_dir=.
```

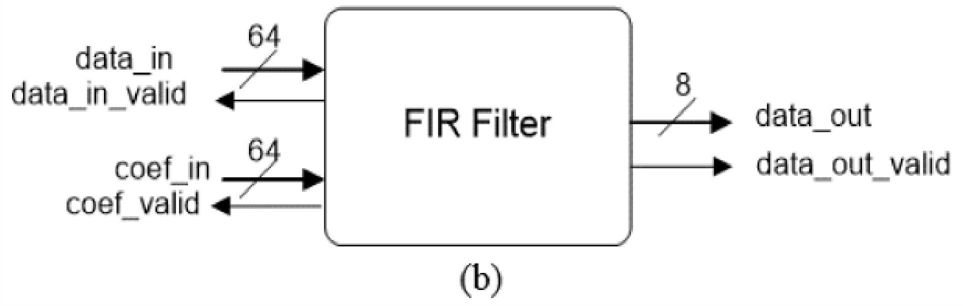
## Valid Signals

- For every IO a single bit signal is generated
- This signal only becomes ‘1’ when:
  - An Input is required
  - A valid out has been generated



(a)

Before valid signal generation

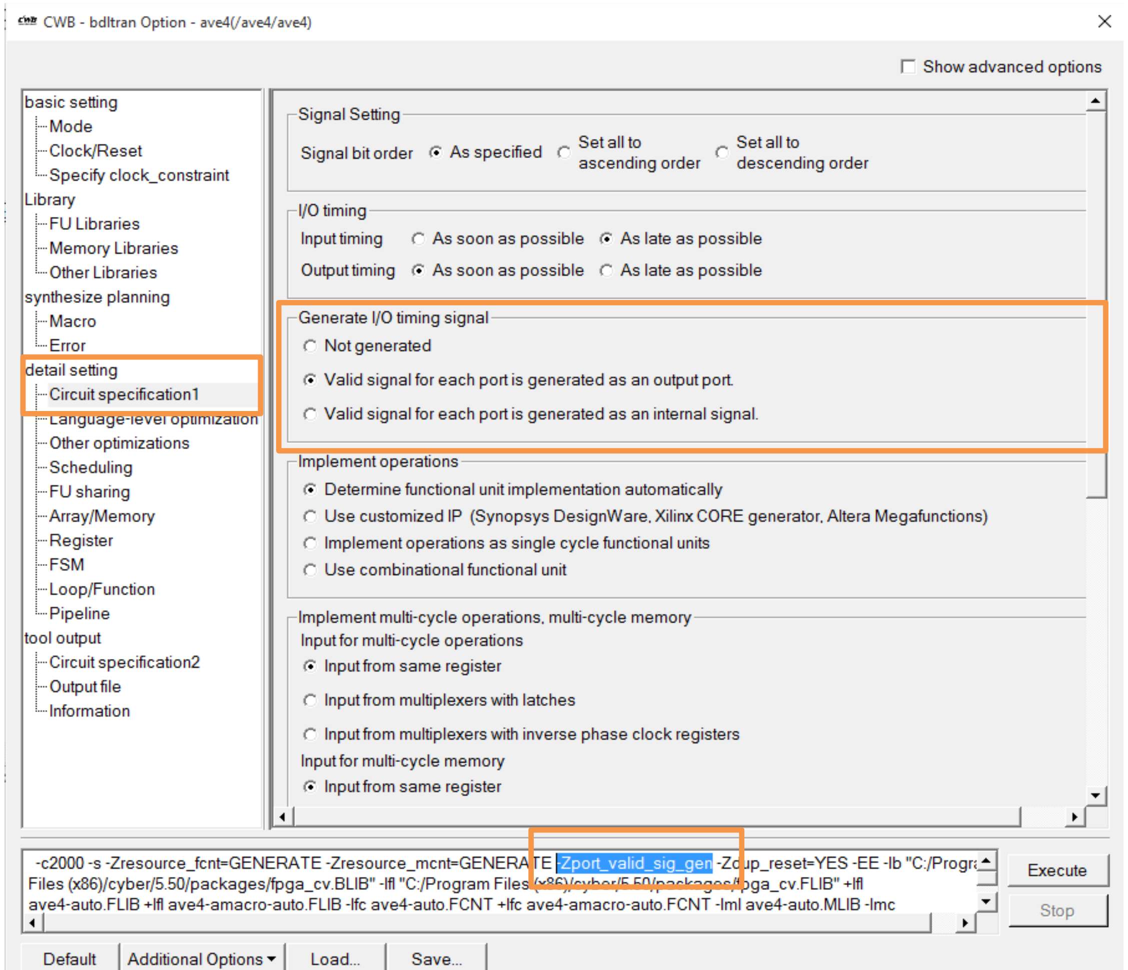


(b)

After valid signal generation

# Automatic Valid Signal Generation

- CWB generates a valid signal for every IO automatically
  → Need to specify it as a synthesis option



CWB - bdltran Option - ave4(/ave4/ave4)

☐ Show advanced options

- basic setting
  - Mode
  - Clock/Reset
  - Specify clock_constraint
- Library
  - FU Libraries
  - Memory Libraries
  - Other Libraries
- synthesize planning
  - Macro
  - Error
- detail setting
  - Circuit specification1
  - Language-level optimization
  - Other optimizations
  - Scheduling
  - FU sharing
  - Array/Memory
  - Register
  - FSM
  - Loop/Function
  - Pipeline
- tool output
  - Circuit specification2
  - Output file
  - Information

Signal Setting

Signal bit order [✓ As specified] Set all to ascending order / Set all to descending order

I/O timing

Input timing: As soon as possible [✓ As late as possible]

Output timing: [✓ As soon as possible] As late as possible

Generate I/O timing signal

- Not generated
- [✓ Valid signal for each port is generated as an output port.]
- Valid signal for each port is generated as an internal signal.

Implement operations

- [✓ Determine functional unit implementation automatically]
- Use customized IP (Synopsys DesignWare, Xilinx CORE generator, Altera Megafunctions)
- Implement operations as single cycle functional units
- Use combinational functional unit

Implement multi-cycle operations, multi-cycle memory

Input for multi-cycle operations

- [✓ Input from same register]
- Input from multiplexers with latches
- Input from multiplexers with inverse phase clock registers

Input for multi-cycle memory

- [✓ Input from same register]

```
-c2000 -s -Zresource_fcnt=GENERATE -Zresource_mcnt=GENERATE -Zport_valid_sig_gen -Zdup_reset=YES -EE -lb "C:/Progra
Files (x86)/cyber/5.50/packages/fpga_cv.BLIB" -lfl "C:/Program Files (x86)/cyber/5.50/packages/fpga_cv.FLIB" +lfl
ave4-auto.FLIB +lfl ave4-amacro-auto.FLIB -lfc ave4-auto.FCNT +lfc ave4-amacro-auto.FCNT -lml ave4-auto.MLIB -lmc
```

## Importance of tlv files

- Allows the functional verification of the module
- Not intended to be part of the final design
- CLV files require the appending of 0's every time the latency of the circuit changes. E.g.

Latency 1

```
3
4
5
1
14
65
```

Latency 2

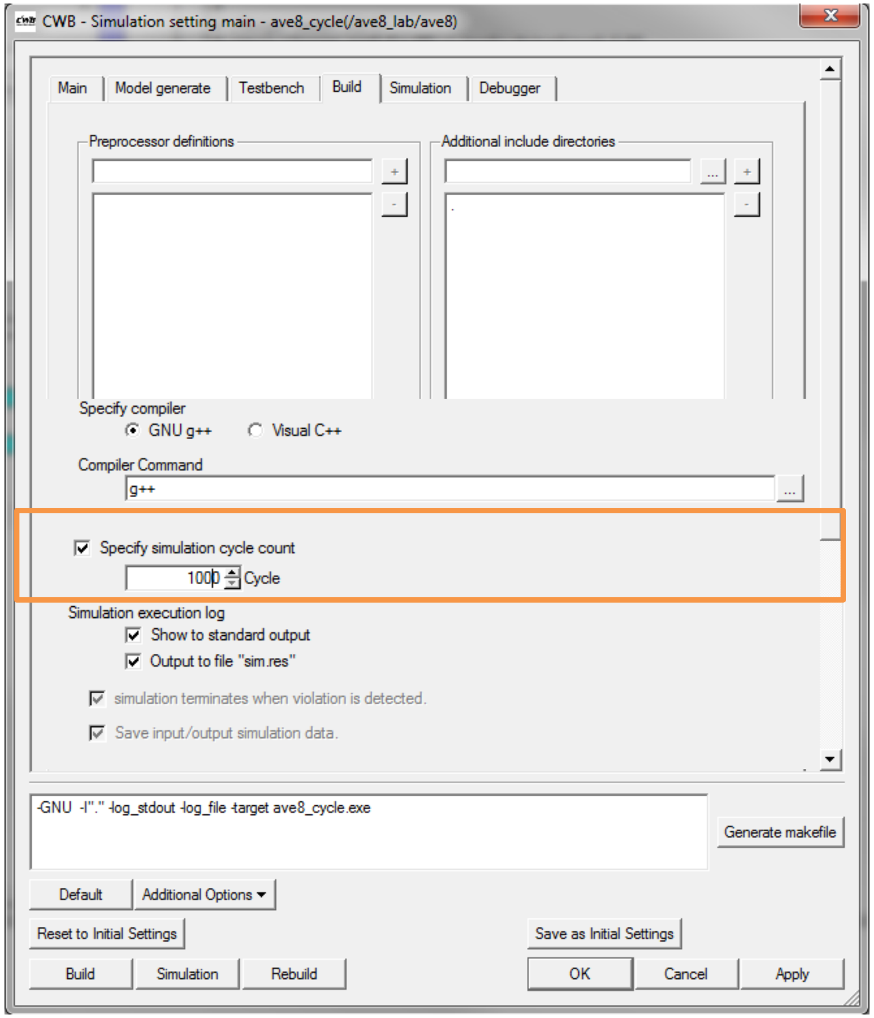
```
3
0
4
0
5
0
1
0
14
0
65
```

Latency 3

```
3
0
0
4
0
0
5
0
0
1
0
0
```

# Random Inputs

- In case of random inputs
  - Need to specify how many
  - Cannot compare against expected output
- Advantages and disadvantages of selecting
  - Show to standard output
  - Output to file “sim.res”


CWB - Simulation setting main - ave8_cycle(/ave8_lab/ave8)

Specify compiler: [✓ GNU g++] Visual C++

Compiler Command: g++

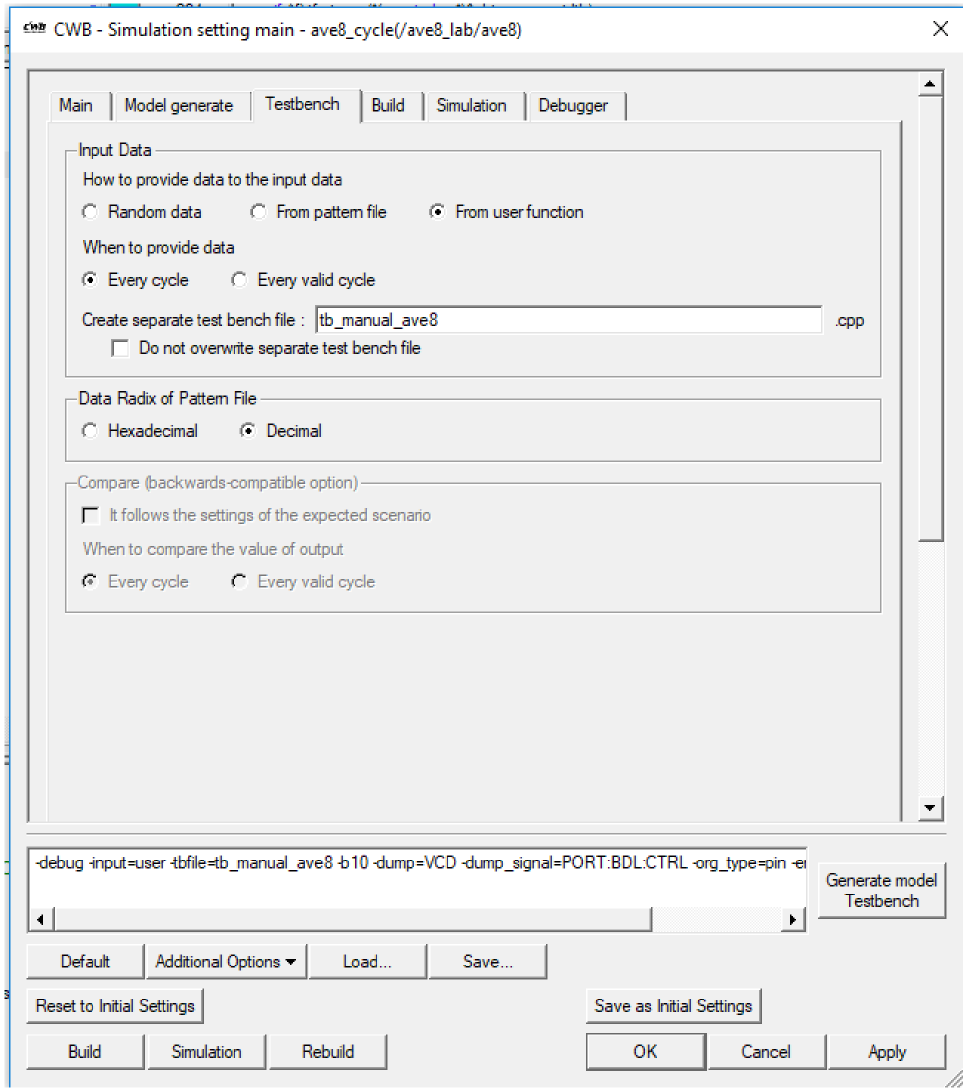
[✓] Specify simulation cycle count: 1000 Cycle

Simulation execution log
[✓] Show to standard output
[✓] Output to file "sim.res"
[✓] simulation terminates when violation is detected.
[✓] Save input/output simulation data.

```
-GNU -I"." -log_stdout -log_file -target ave8_cycle.exe
```

# Manual TB Generation

- In cycle-accurate model generation → Testbench → From user function → create separate testbench file


CWB - Simulation setting main - ave8_cycle(/ave8_lab/ave8)

Input Data

How to provide data to the input data

( ) Random data ( ) From pattern file (● From user function)

When to provide data

(● Every cycle) ( ) Every valid cycle

Create separate test bench file : tb_manual_ave8 .cpp

☐ Do not overwrite separate test bench file

Data Radix of Pattern File

( ) Hexadecimal (● Decimal)

Compare (backwards-compatible option)

☐ It follows the settings of the expected scenario

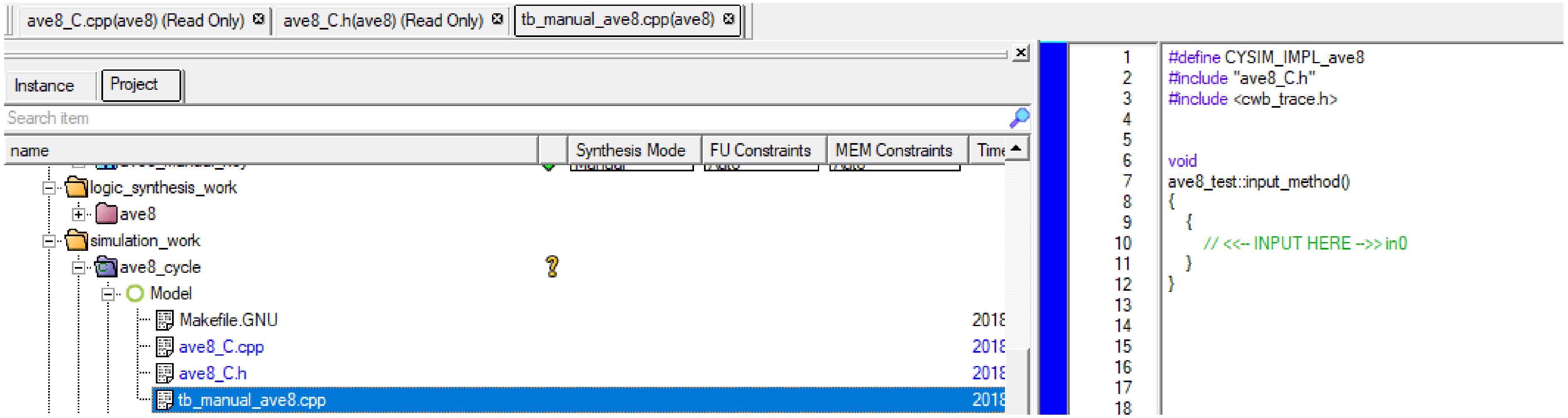
When to compare the value of output

(● Every cycle) ( ) Every valid cycle

```
-debug -input=user -tbfile=tb_manual_ave8 -b10 -dump=VCD -dump_signal=PORT:BDL:CTRL -org_type=pin -e
```

# Manual TB Generation

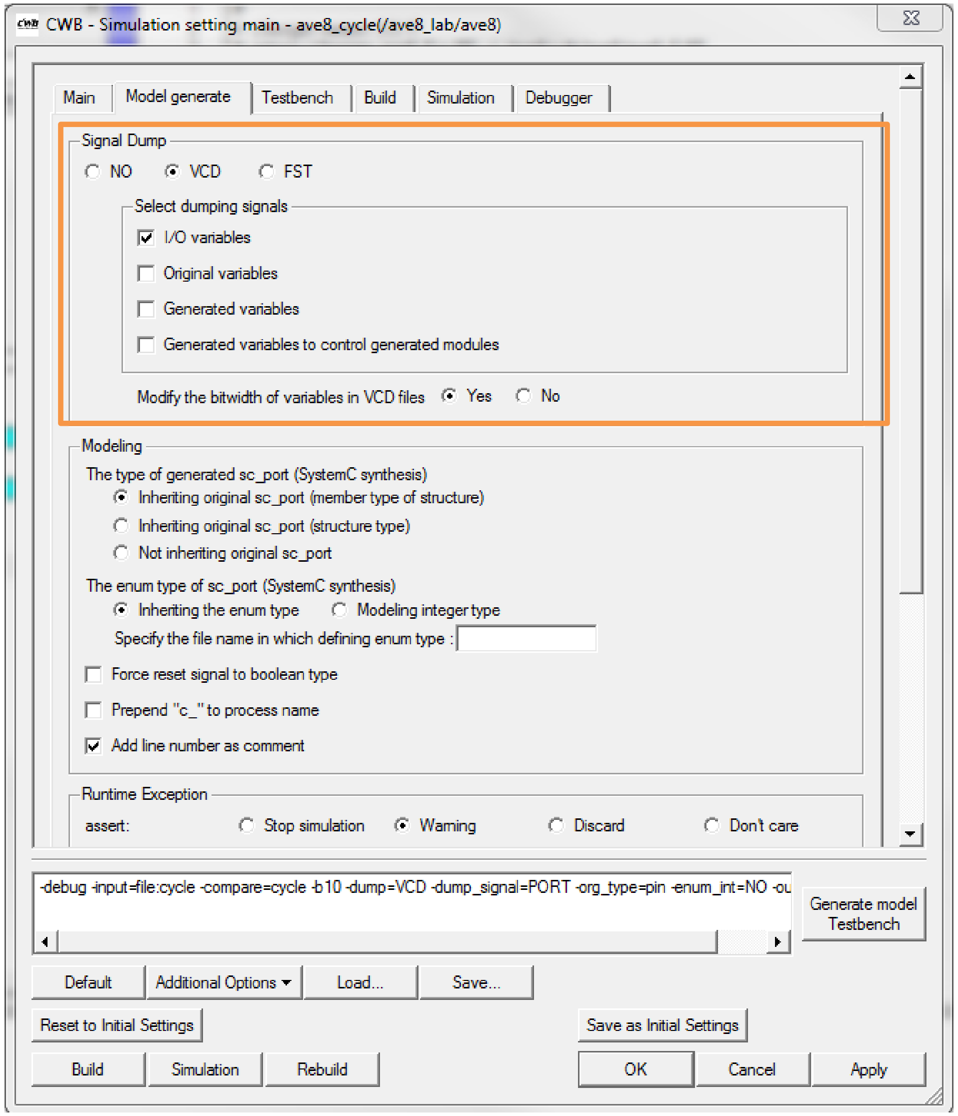
- New file with empty TB is generated



```
#define CYSIM_IMPL_ave8
#include "ave8_C.h"
#include <cwb_trace.h>


void
ave8_test::input_method()
{
   {
      // <<-- INPUT HERE -->> in0
   }
}
```

# Simulation Results

- Text files
- VCD (value change dump)



CWB - Simulation setting main - ave8_cycle(/ave8_lab/ave8)

Main | Model generate | Testbench | Build | Simulation | Debugger

Signal Dump

NO [✓ VCD] FST

Select dumping signals

- [✓] I/O variables
- [ ] Original variables
- [ ] Generated variables
- [ ] Generated variables to control generated modules

Modify the bitwidth of variables in VCD files [✓ Yes] No

Modeling

The type of generated sc_port (SystemC synthesis)

- [✓ Inheriting original sc_port (member type of structure)]
- Inheriting original sc_port (structure type)
- Not inheriting original sc_port

The enum type of sc_port (SystemC synthesis)

[✓ Inheriting the enum type] Modeling integer type

Specify the file name in which defining enum type :

- [ ] Force reset signal to boolean type
- [ ] Prepend "c_" to process name
- [✓] Add line number as comment

Runtime Exception

assert: Stop simulation [✓ Warning] Discard Don't care

```
-debug -input=file:cycle -compare=cycle -b10 -dump=VCD -dump_signal=PORT -org_type=pin -enum_int=NO -ou
```

Generate model Testbench

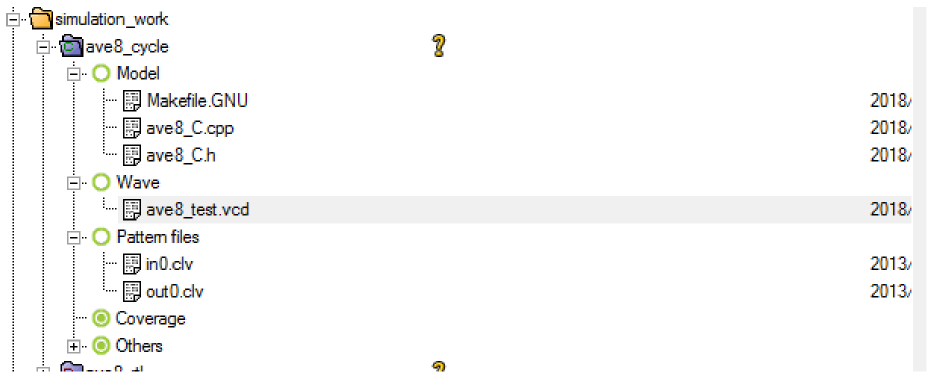
Default | Additional Options | Load... | Save...

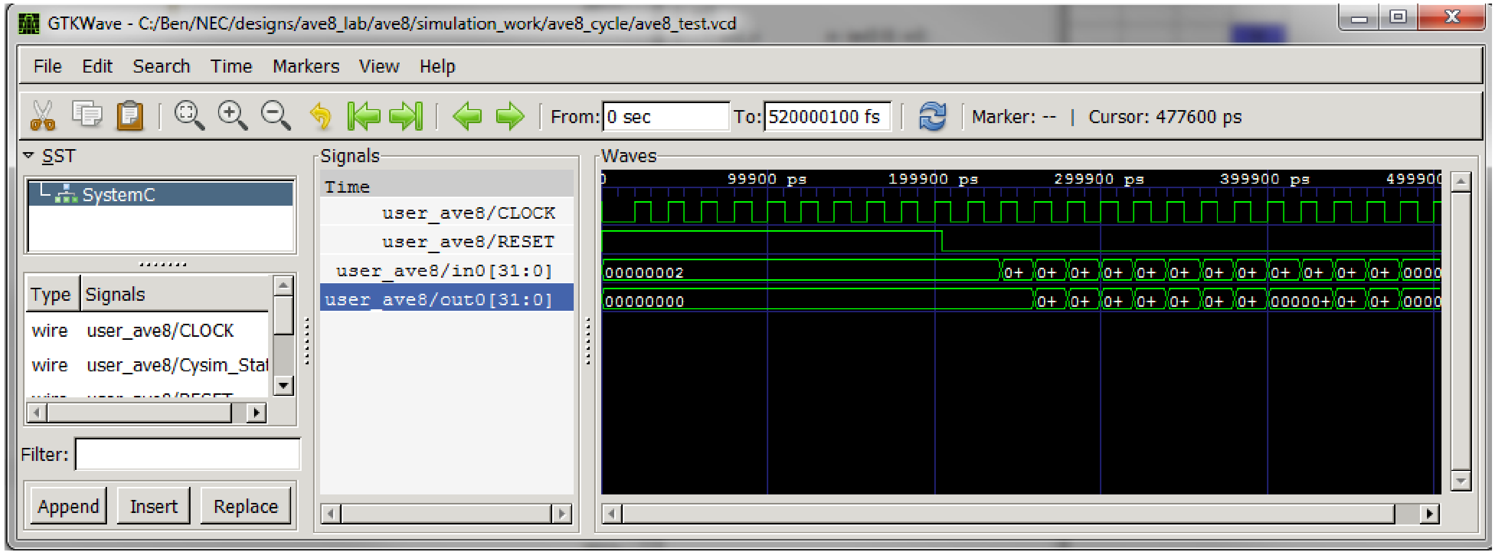
Reset to Initial Settings | Save as Initial Settings

Build | Simulation | Rebuild | OK | Cancel | Apply

# Model Generators Outputs

- Model generator:
  - SystemC model
  - Makefile
- Needed files
  - Pattern files (.clv/.tlv)
- After compiling and executing
  - VCD, .res, terminal
- VCD file can be opened with third party VCD viewer, e.g., GTKwave





# Cycle-Accurate Simulation Enhancements

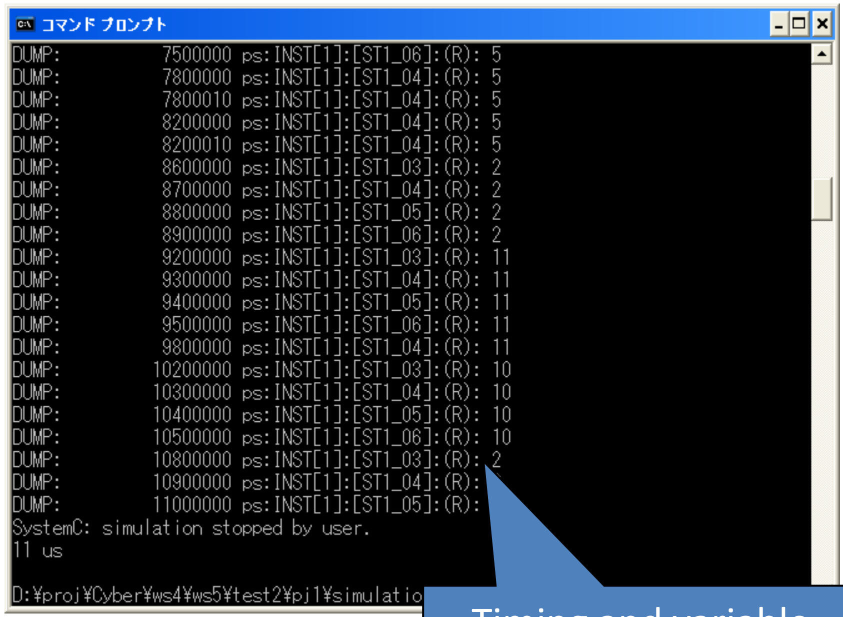
- printf' style debugging capability
- Runtime verification of 'assert' condition

Source Code

```
in ter(0:4) a1, a2;
out ter(0:8) dout;
process main()
{
  reg unsigned char i;
  reg unsigned char x, x1, x2, x3, x4, x5;
  x = a1;
  /* Cyber folding = 1 */
  for(i=0; i<x; i++) {
     x1 /* Cyber dump={"%d", #} */ = a1;
     x2 = x1/* Cyber dump={"%d", #} */ + a2;
     x3 = x1/* Cyber dump={"%d", #} */ - x2;
     x4 = x1/* Cyber dump={"%d", #} */ * x3;
     if(x4==0) continue;
     x5 = x1/* Cyber dump={"%d", #} */ / x4;
     dout = x5;
  }
  dout = x3;
  dout = x2;
  dout = x1/* Cyber dump={"%d", #} */;
}
```

Synthesis

Cycle-accurate simulation

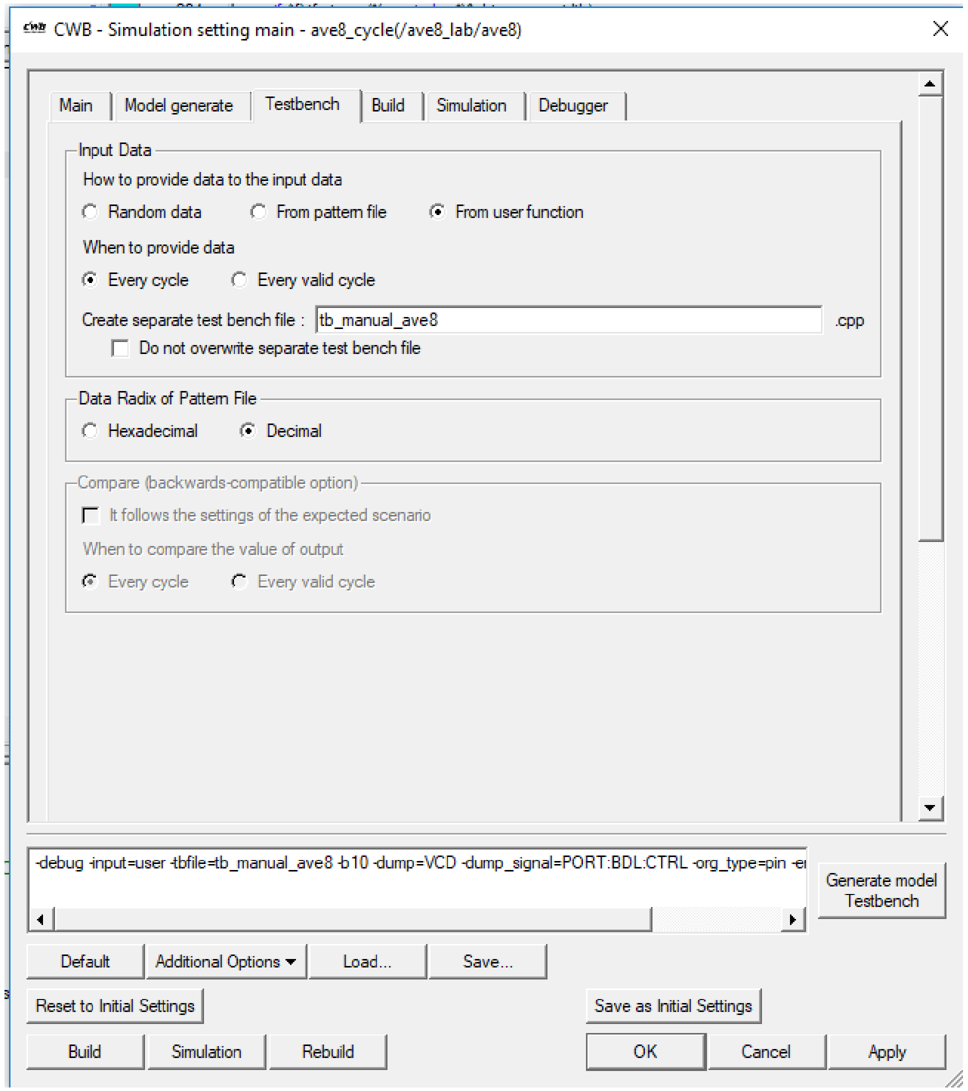


```
DUMP:          7500000 ps:INST[1]:[ST1_06]:(R): 5
DUMP:          7800000 ps:INST[1]:[ST1_04]:(R): 5
DUMP:          7800010 ps:INST[1]:[ST1_04]:(R): 5
DUMP:          8200000 ps:INST[1]:[ST1_04]:(R): 5
DUMP:          8200010 ps:INST[1]:[ST1_04]:(R): 5
DUMP:          8600000 ps:INST[1]:[ST1_03]:(R): 2
DUMP:          8700000 ps:INST[1]:[ST1_04]:(R): 2
DUMP:          8800000 ps:INST[1]:[ST1_05]:(R): 2
DUMP:          8900000 ps:INST[1]:[ST1_06]:(R): 2
DUMP:          9200000 ps:INST[1]:[ST1_03]:(R): 11
DUMP:          9300000 ps:INST[1]:[ST1_04]:(R): 11
DUMP:          9400000 ps:INST[1]:[ST1_05]:(R): 11
DUMP:          9500000 ps:INST[1]:[ST1_06]:(R): 11
DUMP:          9800000 ps:INST[1]:[ST1_04]:(R): 11
DUMP:         10200000 ps:INST[1]:[ST1_03]:(R): 10
DUMP:         10300000 ps:INST[1]:[ST1_04]:(R): 10
DUMP:         10400000 ps:INST[1]:[ST1_05]:(R): 10
DUMP:         10500000 ps:INST[1]:[ST1_06]:(R): 10
DUMP:         10800000 ps:INST[1]:[ST1_03]:(R): 2
DUMP:         10900000 ps:INST[1]:[ST1_04]:(R):
DUMP:         11000000 ps:INST[1]:[ST1_05]:(R):
SystemC: simulation stopped by user.
11 us

D:¥proj¥Cyber¥ws4¥ws5¥test2¥pj1¥simulatio
```

Timing and variable value dumped at the console

# Manual TB Generation

- In cycle-accurate model generation → Testbench → From user function → create separate testbench file



Input Data

How to provide data to the input data

Random data ( ) From pattern file ( ) From user function (✓)

When to provide data

Every cycle (✓) Every valid cycle ( )

Create separate test bench file : tb_manual_ave8 .cpp

Do not overwrite separate test bench file [ ]

Data Radix of Pattern File

Hexadecimal ( ) Decimal (✓)

Compare (backwards-compatible option)

It follows the settings of the expected scenario [ ]

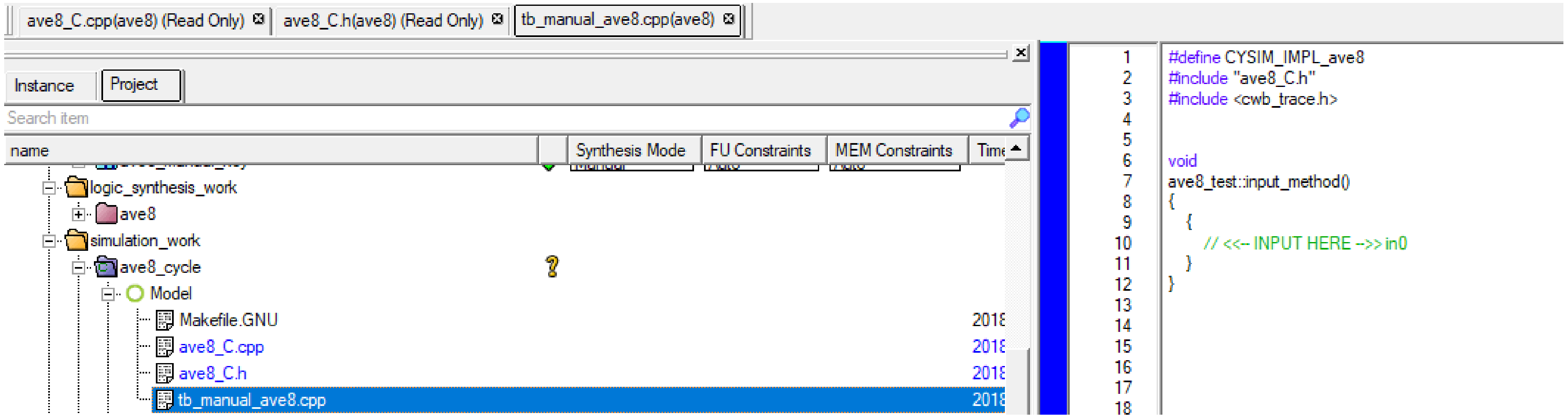
When to compare the value of output

Every cycle (✓) Every valid cycle ( )

```
-debug -input=user -tbfile=tb_manual_ave8 -b10 -dump=VCD -dump_signal=PORT:BDL:CTRL -org_type=pin -e
```

# Manual TB Generation

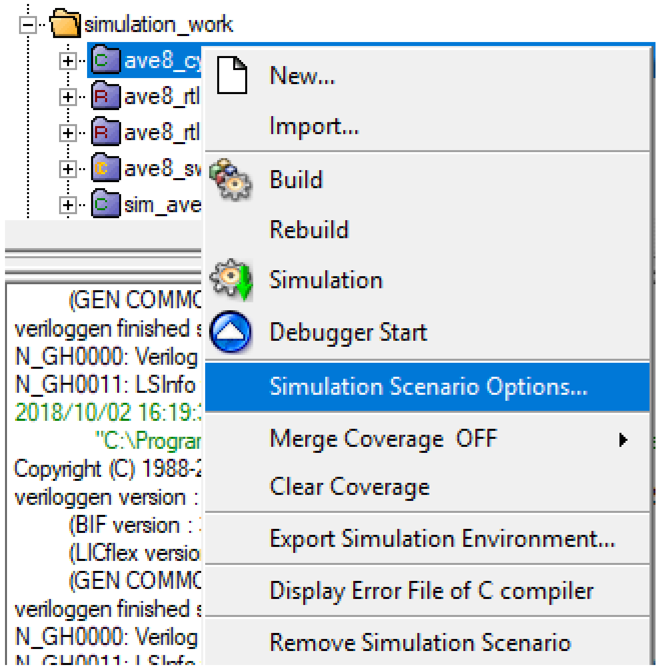
- New file with empty TB is generated



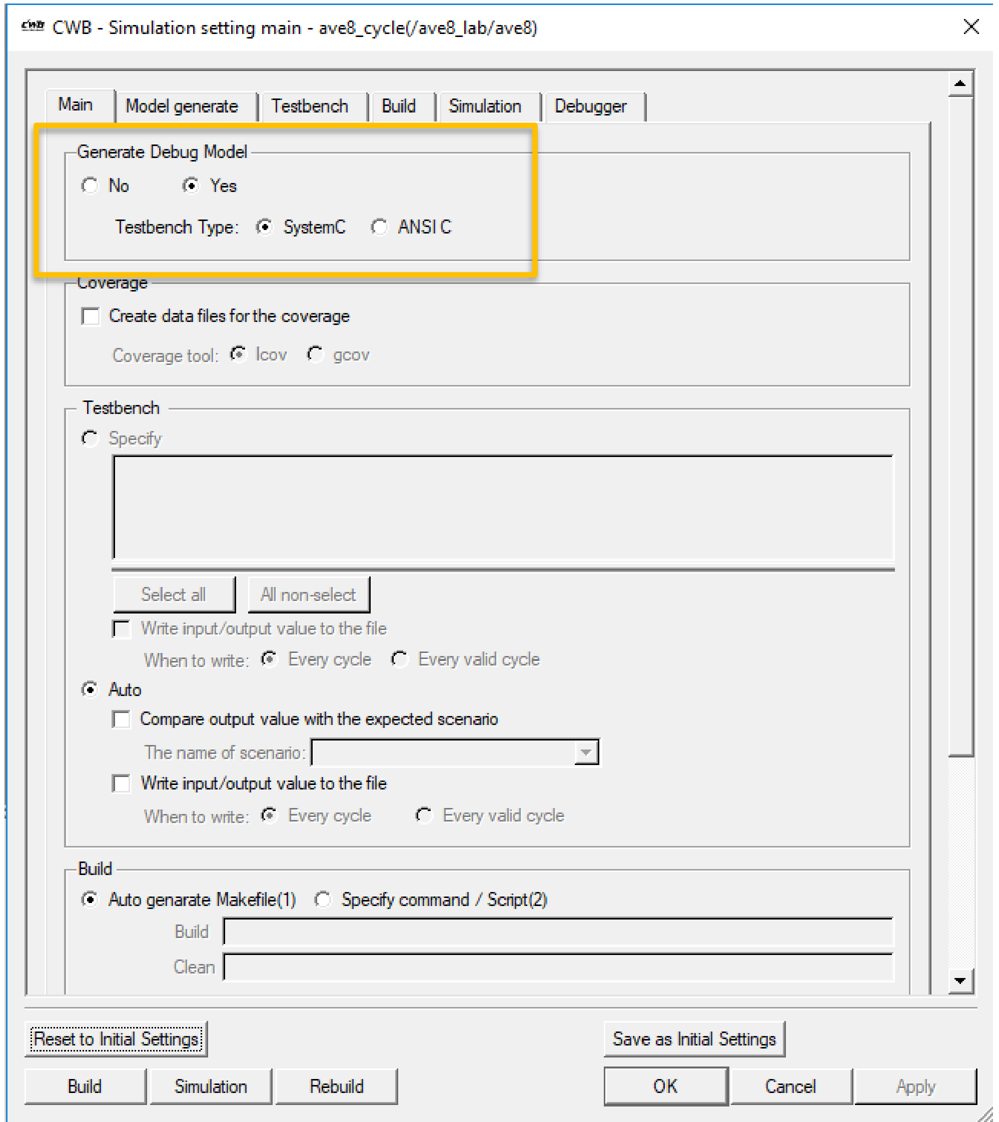
```
#define CYSIM_IMPL_ave8
#include "ave8_C.h"
#include <cwb_trace.h>


void
ave8_test::input_method()
{
   {
      // <<-- INPUT HERE -->> in0
   }
}
```

# Creating a Debuggable Model

- In Simulation scenario options → Generate Debug Model



```
        (GEN COMMC
verilоggen finished s
N_GH0000: Verilog
N_GH0011: LSInfo
2018/10/02 16:19::
        "C:\Prograr
Copyright (C) 1988-2
veriloggen version :
        (BIF version : 
        (LICflex versio
        (GEN COMMC
veriloggen finished s
N_GH0000: Verilog
N_GH0011: LSInfo
```


CWB - Simulation setting main - ave8_cycle(/ave8_lab/ave8)

Generate Debug Model
No [✓ Yes]
Testbench Type: [✓ SystemC] ANSI C

Coverage
☐ Create data files for the coverage
Coverage tool: [✓ lcov] gcov

Testbench
Specify

☐ Write input/output value to the file
When to write: [✓ Every cycle] Every valid cycle

[✓ Auto]
☐ Compare output value with the expected scenario
The name of scenario:
☐ Write input/output value to the file
When to write: [✓ Every cycle] Every valid cycle

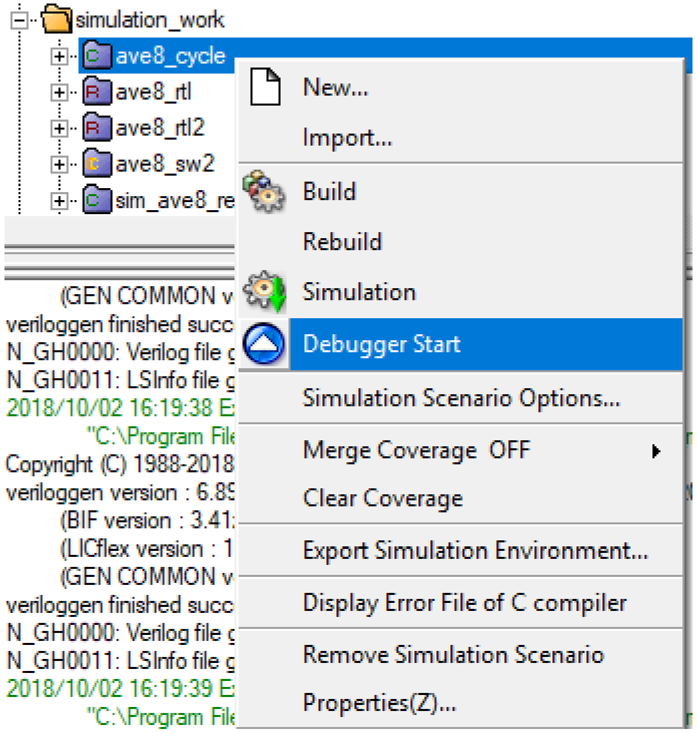
Build
[✓ Auto genarate Makefile(1)] Specify command / Script(2)
Build
Clean

# Starting Debugger

- Right-click on simulation folder → debugger start

OR

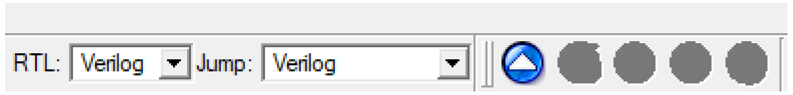
- On toolbar



```
simulation_work
  ave8_cycle
  ave8_rtl
  ave8_rtl2
  ave8_sw2
  sim_ave8_re
```

```
(GEN COMMON v
verilogen finished succ
N_GH0000: Verilog file g
N_GH0011: LSInfo file g
2018/10/02 16:19:38 E
"C:\Program File
Copyright (C) 1988-2018
verilogen version : 6.89
(BIF version : 3.41:
(LICflex version : 1
(GEN COMMON v
verilogen finished succ
N_GH0000: Verilog file g
N_GH0011: LSInfo file g
2018/10/02 16:19:39 E
"C:\Program File
```

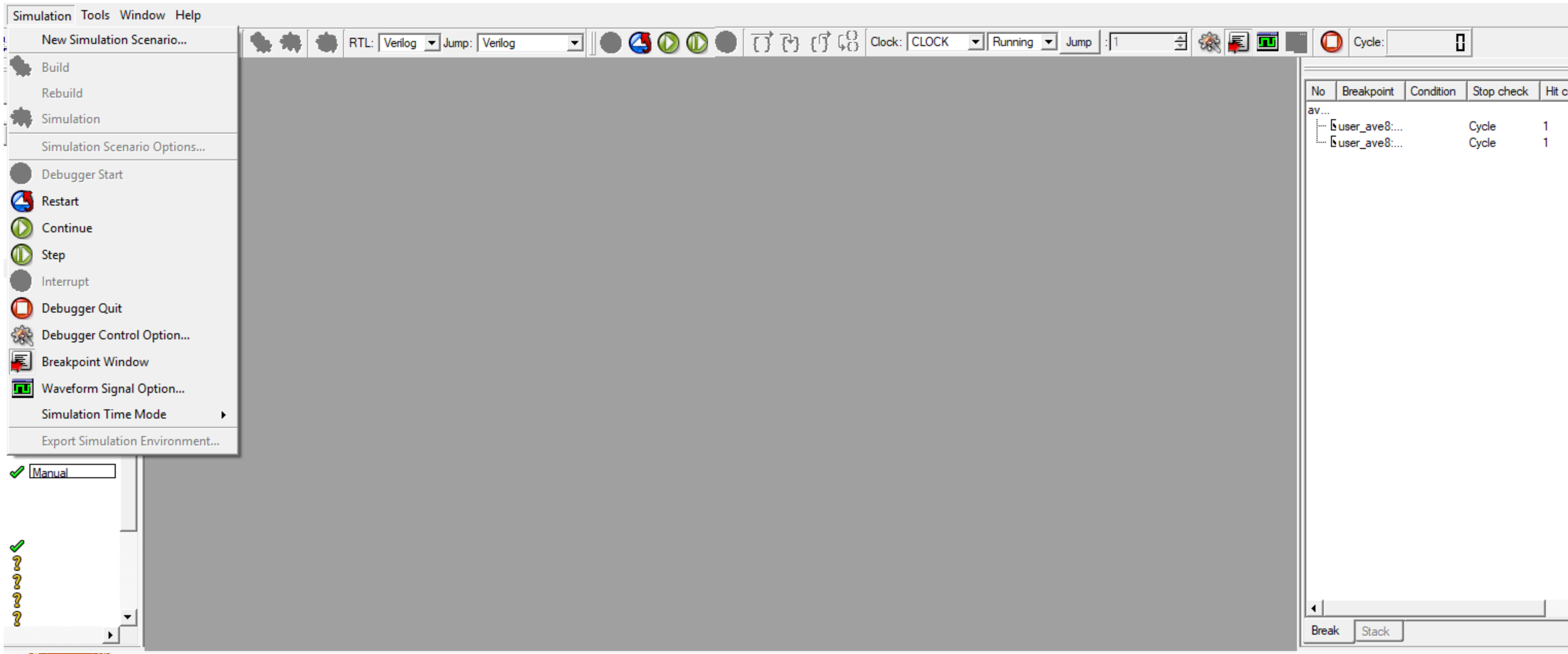
New...
Import...
Build
Rebuild
Simulation
Debugger Start
Simulation Scenario Options...
Merge Coverage OFF
Clear Coverage
Export Simulation Environment...
Display Error File of C compiler
Remove Simulation Scenario
Properties(Z)...



RTL: Verilog Jump: Verilog

# Debugging Options

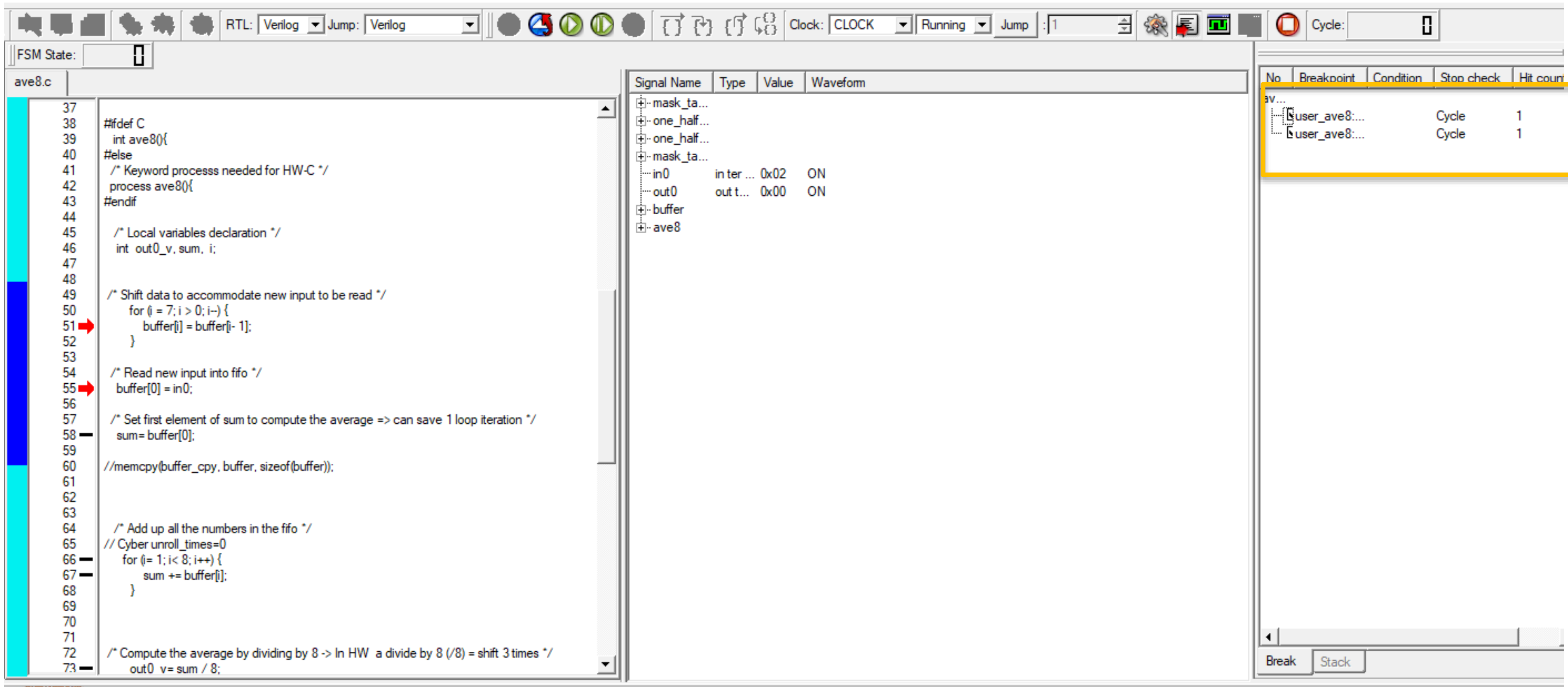
- Restart
- Step
- Run



| No | Breakpoint | Condition | Stop check | Hit c |
|---|---|---|---|---|
| av... | | | | |
| | user_ave8:... | | Cycle | 1 |
| | user_ave8:... | | Cycle | 1 |

# Instance Viewer

- Click on new pane → opens instance viewer



RTL: Verilog  Jump: Verilog  Clock: CLOCK  Running  Jump  1  Cycle:

FSM State:

ave8.c

```
#ifdef C
  int ave8(){
#else
 /* Keyword processs needed for HW-C */
 process ave8(){
#endif

  /* Local variables declaration */
  int  out0_v, sum,  i;


 /* Shift data to accommodate new input to be read */
     for (i = 7; i > 0; i--) {
        buffer[i] = buffer[i- 1];
     }

 /* Read new input into fifo */
  buffer[0] = in0;

 /* Set first element of sum to compute the average => can save 1 loop iteration */
  sum= buffer[0];

//memcpy(buffer_cpy, buffer, sizeof(buffer));



  /* Add up all the numbers in the fifo */
// Cyber unroll_times=0
    for (i= 1; i< 8; i++) {
        sum += buffer[i];
     }



 /* Compute the average by dividing by 8 -> In HW  a divide by 8 (/8) = shift 3 times */
     out0_v= sum / 8;
```

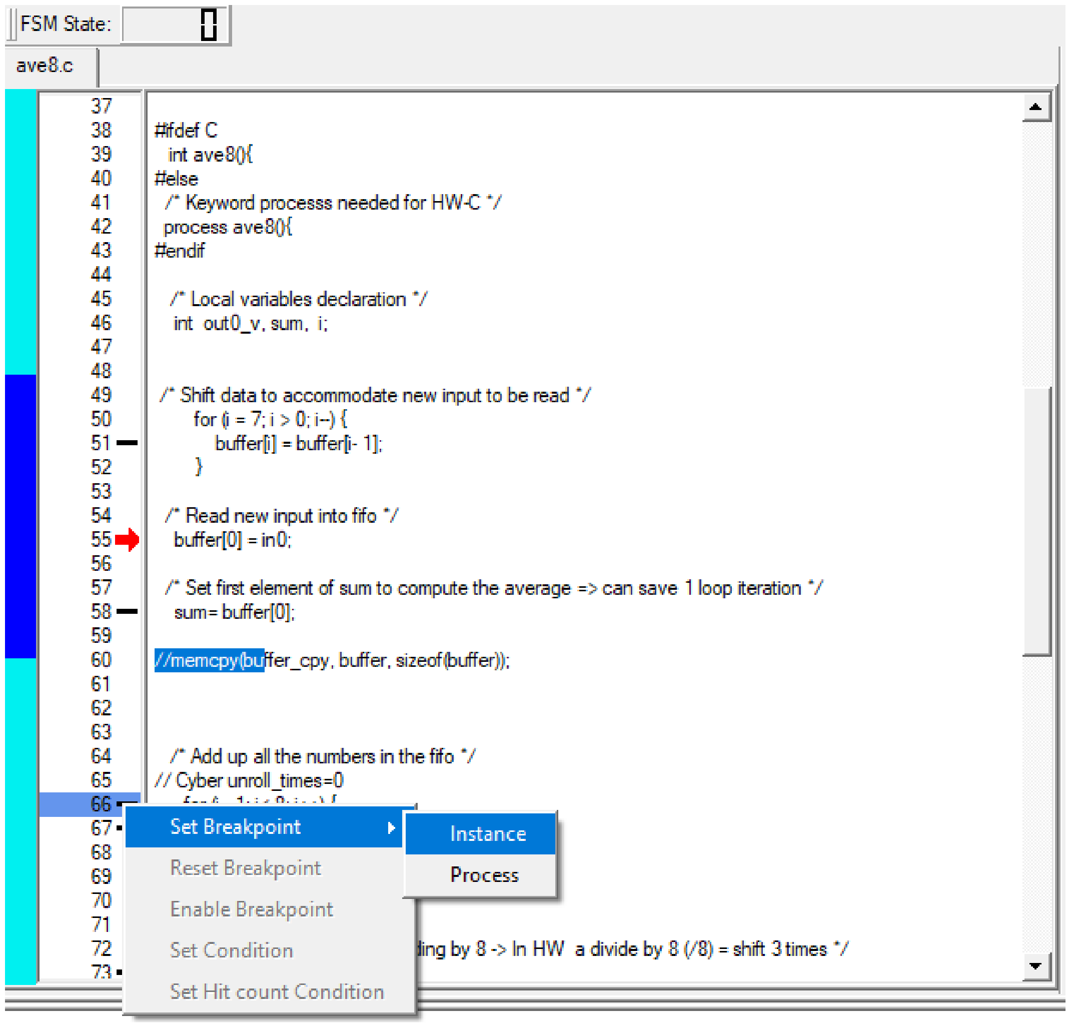
| Signal Name | Type | Value | Waveform |
|---|---|---|---|
| mask_ta... | | | |
| one_half... | | | |
| one_half... | | | |
| mask_ta... | | | |
| in0 | in ter ... | 0x02 | ON |
| out0 | out t... | 0x00 | ON |
| buffer | | | |
| ave8 | | | |

| No | Breakpoint | Condition | Stop check | Hit coun |
|---|---|---|---|---|
| av... | | | | |
| | user_ave8:... | | Cycle | 1 |
| | user_ave8:... | | Cycle | 1 |

Break  Stack

# Setting breakpoints

- Right-click on line number of instance viewer → insert breakpoint

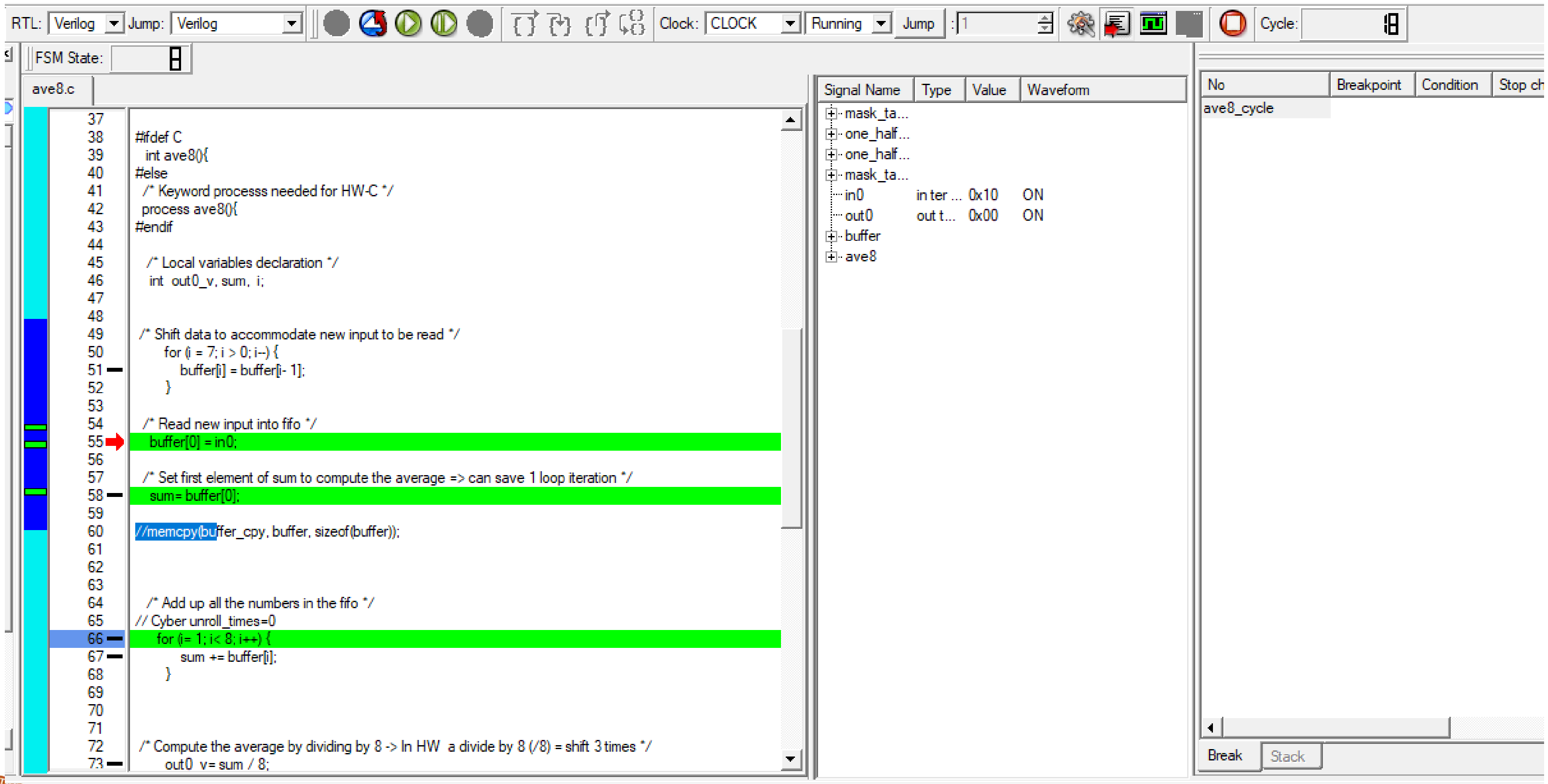


```
#ifdef C
  int ave8(){
#else
 /* Keyword processs needed for HW-C */
 process ave8(){
#endif

  /* Local variables declaration */
  int  out0_v, sum,  i;


 /* Shift data to accommodate new input to be read */
      for (i = 7; i > 0; i--) {
         buffer[i] = buffer[i- 1];
      }

 /* Read new input into fifo */
  buffer[0] = in0;

 /* Set first element of sum to compute the average => can save 1 loop iteration */
  sum= buffer[0];

//memcpy(buffer_cpy, buffer, sizeof(buffer));



  /* Add up all the numbers in the fifo */
// Cyber unroll_times=0
```

```
ding by 8 -> In HW  a divide by 8 (/8) = shift 3 times */
```

# Start Debugging

- Click on start debugging
- Stops at breakpoint
- Highlights lines of code executed concurrently



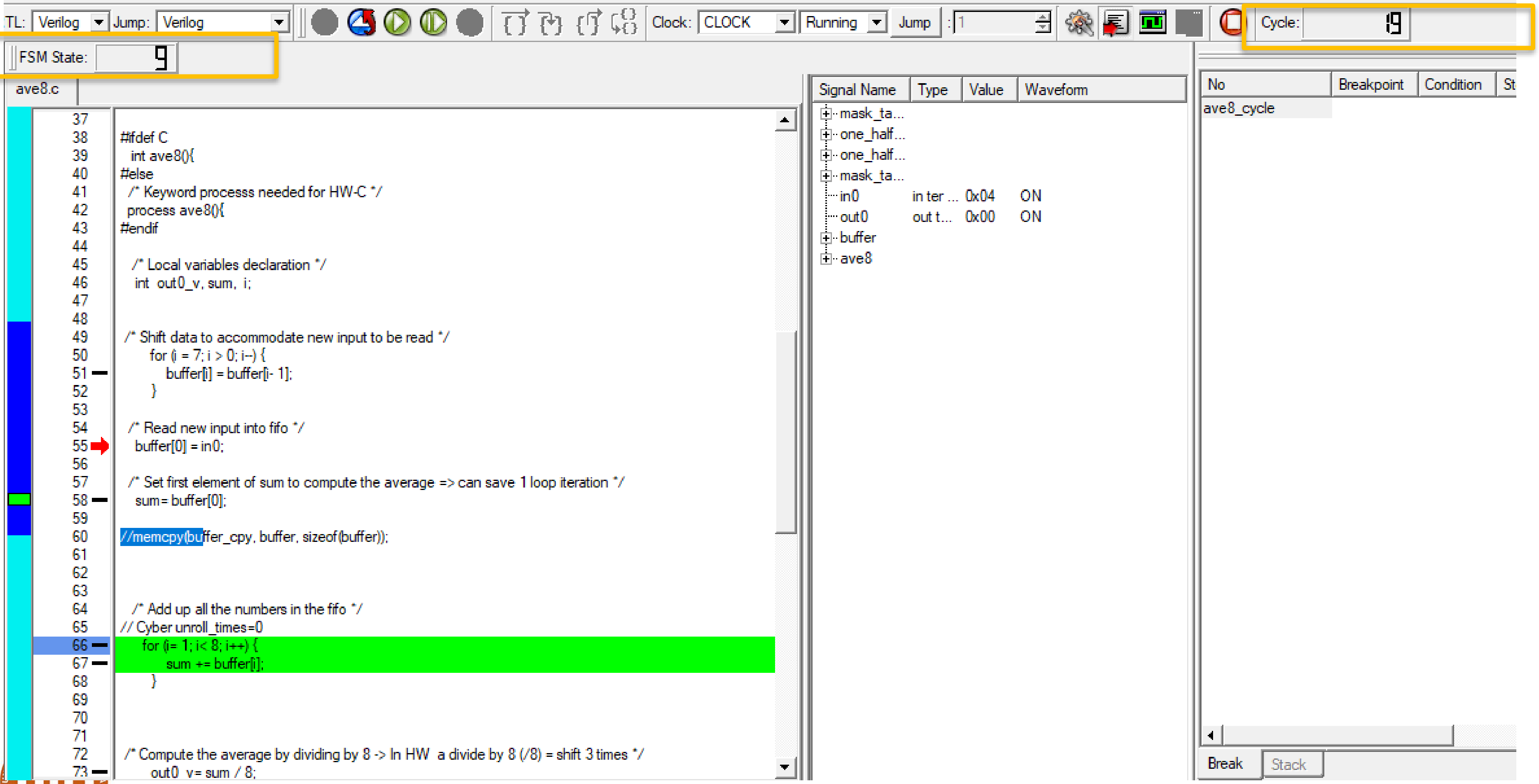
```
#ifdef C
  int ave8(){
#else
  /* Keyword processs needed for HW-C */
  process ave8(){
#endif

   /* Local variables declaration */
   int  out0_v, sum,  i;


 /* Shift data to accommodate new input to be read */
      for (i = 7; i > 0; i--) {
         buffer[i] = buffer[i- 1];
      }

  /* Read new input into fifo */
   buffer[0] = in0;

  /* Set first element of sum to compute the average => can save 1 loop iteration */
   sum= buffer[0];

//memcpy(buffer_cpy, buffer, sizeof(buffer));



   /* Add up all the numbers in the fifo */
// Cyber unroll_times=0
     for (i= 1; i< 8; i++) {
         sum += buffer[i];
      }



 /* Compute the average by dividing by 8 -> In HW  a divide by 8 (/8) = shift 3 times */
      out0_v = sum / 8;
```

| Signal Name | Type | Value | Waveform |
|---|---|---|---|
| mask_ta... | | | |
| one_half... | | | |
| one_half... | | | |
| mask_ta... | | | |
| in0 | in ter ... | 0x10 | ON |
| out0 | out t... | 0x00 | ON |
| buffer | | | |
| ave8 | | | |

| No | Breakpoint | Condition | Stop ch |
|---|---|---|---|
| ave8_cycle | | | |

# Stepping

- Click on step
- Cycle count and FSM change
- New lines of code higlighted



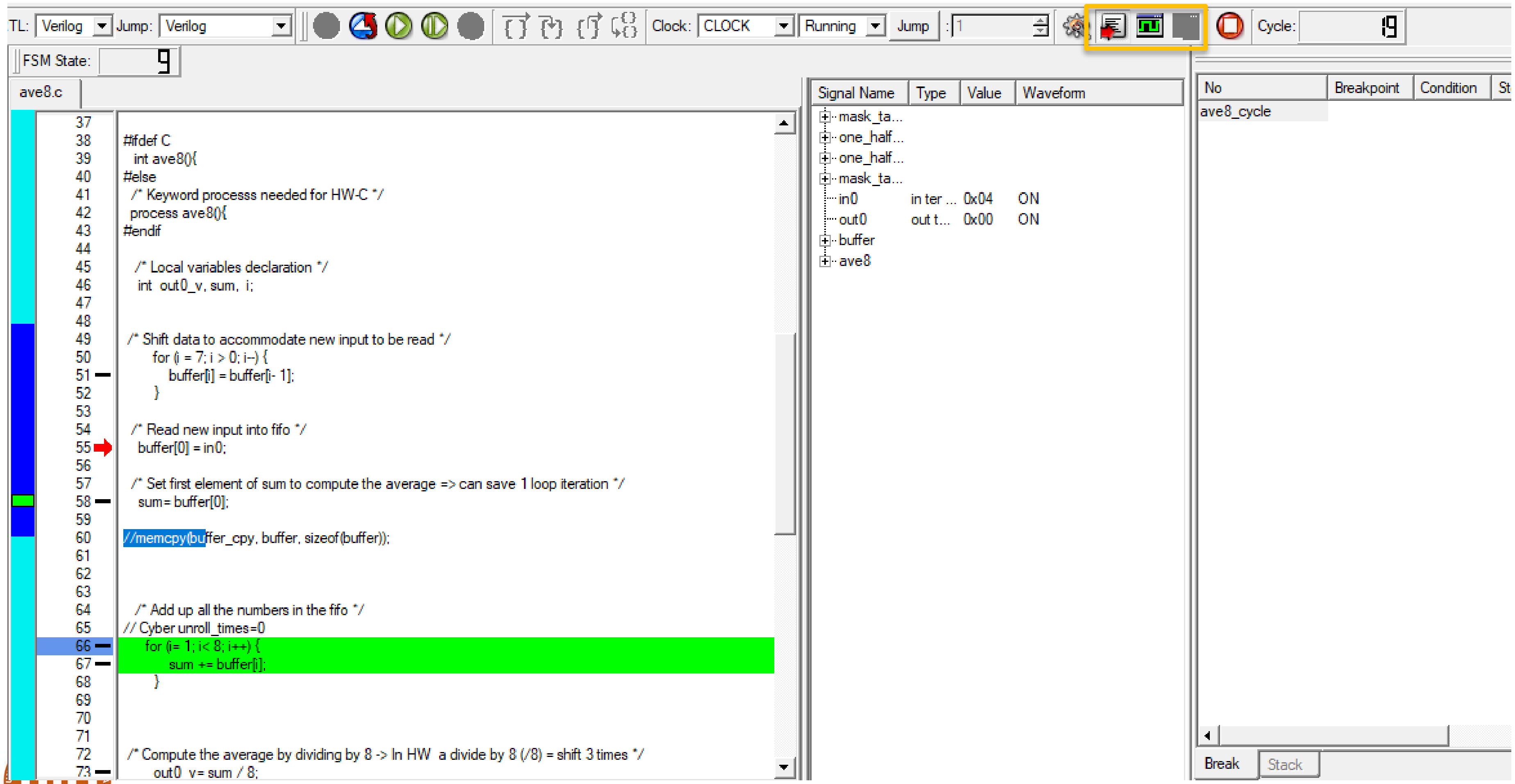
```
#ifdef C
  int ave8(){
#else
  /* Keyword processs needed for HW-C */
 process ave8(){
#endif

   /* Local variables declaration */
   int  out0_v, sum,  i;


 /* Shift data to accommodate new input to be read */
      for (i = 7; i > 0; i--) {
          buffer[i] = buffer[i- 1];
      }

  /* Read new input into fifo */
   buffer[0] = in0;

  /* Set first element of sum to compute the average => can save 1 loop iteration */
   sum= buffer[0];

//memcpy(buffer_cpy, buffer, sizeof(buffer));



   /* Add up all the numbers in the fifo */
// Cyber unroll_times=0
      for (i= 1; i< 8; i++) {
          sum += buffer[i];
      }



 /* Compute the average by dividing by 8 -> In HW  a divide by 8 (/8) = shift 3 times */
      out0_v= sum / 8;
```

| Signal Name | Type | Value | Waveform |
|---|---|---|---|
| mask_ta... | | | |
| one_half... | | | |
| one_half... | | | |
| mask_ta... | | | |
| in0 | in ter ... | 0x04 | ON |
| out0 | out t... | 0x00 | ON |
| buffer | | | |
| ave8 | | | |

| No | Breakpoint | Condition | St |
|---|---|---|---|
| ave8_cycle | | | |

# Debugger+ Waveform Viewer

- Click on waveform button on toolbar



```
#ifdef C
  int ave8(){
#else
  /* Keyword processs needed for HW-C */
  process ave8(){
#endif

   /* Local variables declaration */
   int  out0_v, sum,  i;


 /* Shift data to accommodate new input to be read */
      for (i = 7; i > 0; i--) {
          buffer[i] = buffer[i- 1];
      }

  /* Read new input into fifo */
   buffer[0] = in0;

  /* Set first element of sum to compute the average => can save 1 loop iteration */
   sum= buffer[0];

//memcpy(buffer_cpy, buffer, sizeof(buffer));



   /* Add up all the numbers in the fifo */
// Cyber unroll_times=0
    for (i= 1; i< 8; i++) {
          sum += buffer[i];
      }



 /* Compute the average by dividing by 8 -> In HW  a divide by 8 (/8) = shift 3 times */
      out0_v= sum / 8;
```

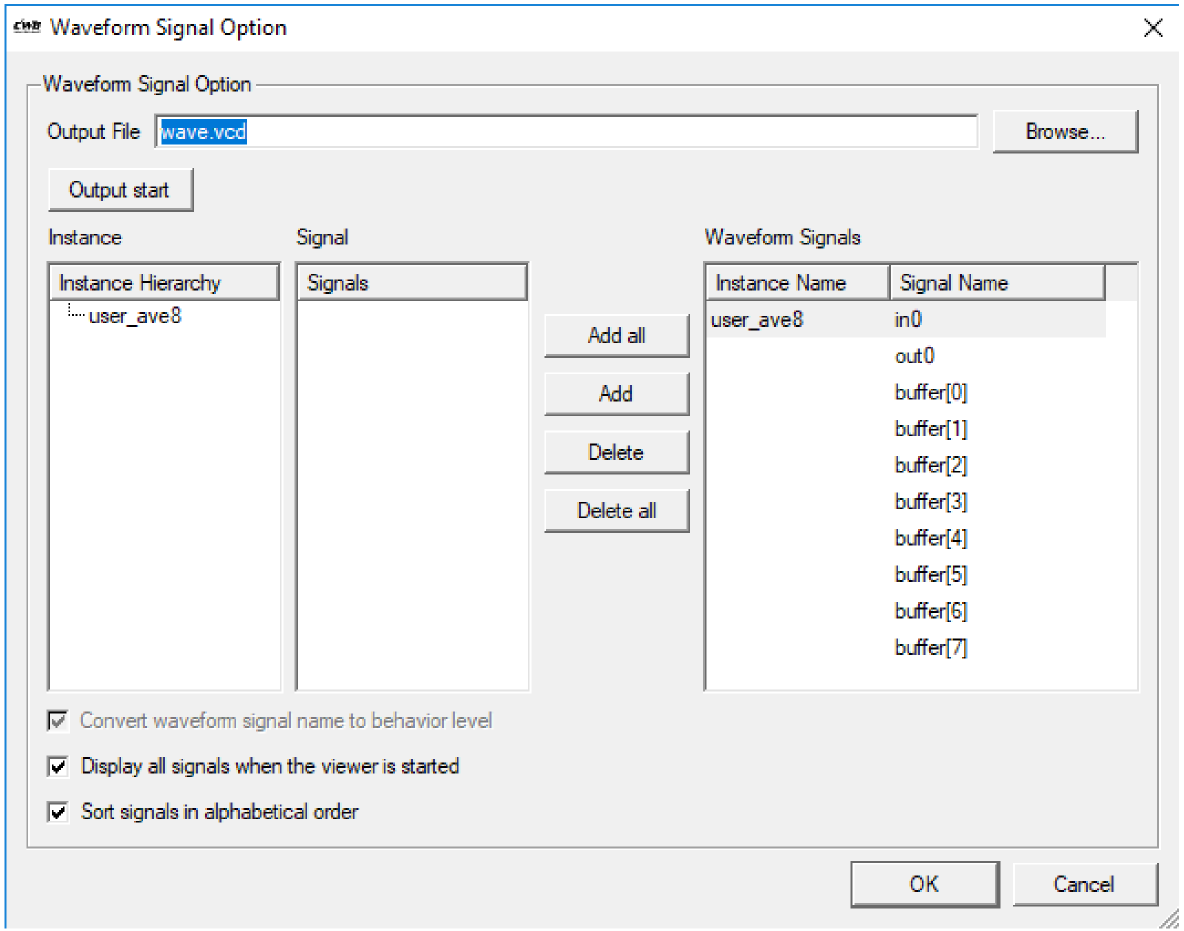
| Signal Name | Type | Value | Waveform |
|---|---|---|---|
| mask_ta... | | | |
| one_half... | | | |
| one_half... | | | |
| mask_ta... | | | |
| in0 | in ter ... | 0x04 | ON |
| out0 | out t... | 0x00 | ON |
| buffer | | | |
| ave8 | | | |

| No | Breakpoint | Condition | St |
|---|---|---|---|
| ave8_cycle | | | |

Break Stack

# Debugger+ Waveform Viewer

- Specify signals to be traced (click on instance hiearcy)
- Specify name of VCD file to dump signals



Waveform Signal Option

Waveform Signal Option

Output File wave.vcd

Browse...

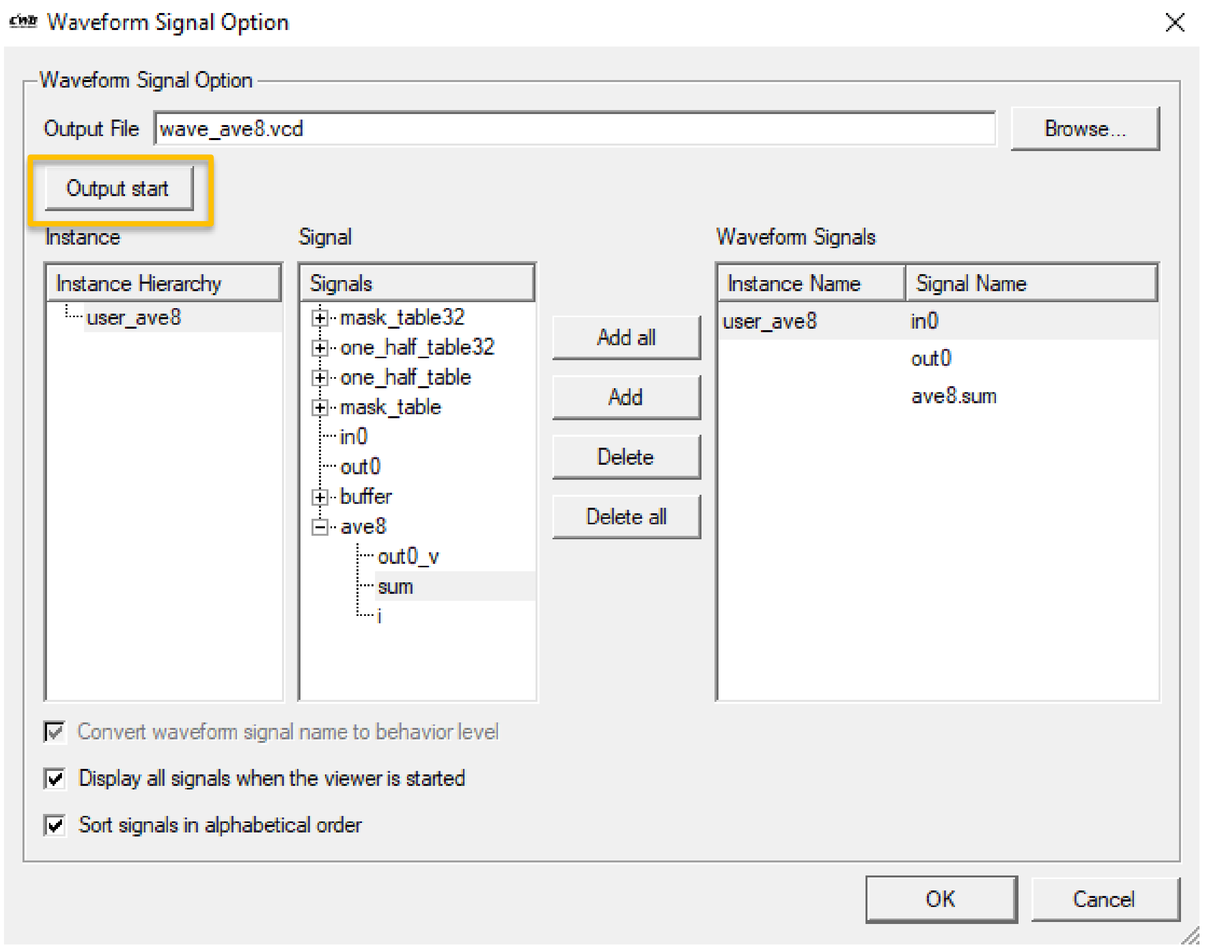
Output start

| Instance Name | Signal Name |
|---|---|
| user_ave8 | in0 |
| | out0 |
| | buffer[0] |
| | buffer[1] |
| | buffer[2] |
| | buffer[3] |
| | buffer[4] |
| | buffer[5] |
| | buffer[6] |
| | buffer[7] |

- [✓] Convert waveform signal name to behavior level
- [✓] Display all signals when the viewer is started
- [✓] Sort signals in alphabetical order

# Tracing Internal Signals

- Instance hierarchy → signals → add
- DON'T FORGET → Click "Output Start" to start dumping VCD File



Waveform Signal Option

Output File: wave_ave8.vcd

Output start

| Instance Hierarchy |
|---|
| user_ave8 |

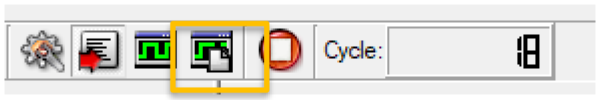
Signals:
- mask_table32
- one_half_table32
- one_half_table
- mask_table
- in0
- out0
- buffer
- ave8
  - out0_v
  - sum
  - i

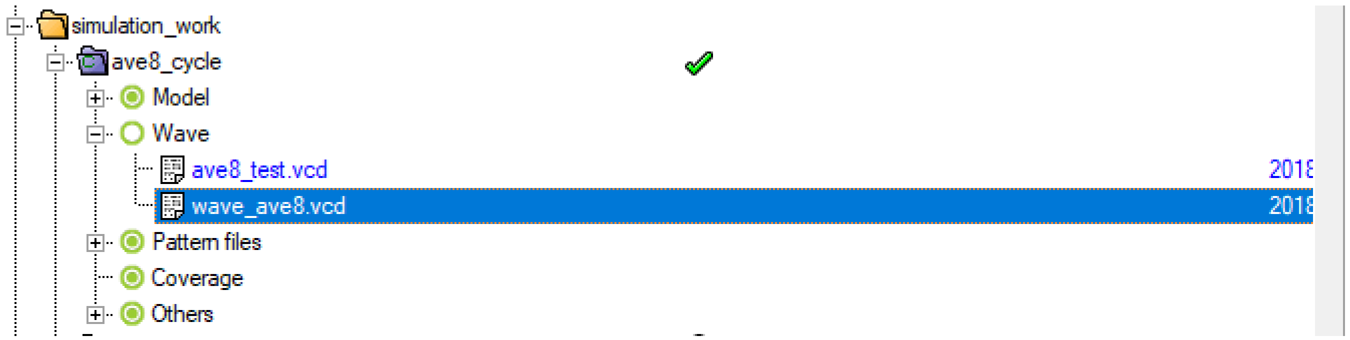
| Instance Name | Signal Name |
|---|---|
| user_ave8 | in0 |
| | out0 |
| | ave8.sum |

[✓] Convert waveform signal name to behavior level
[✓] Display all signals when the viewer is started
[✓] Sort signals in alphabetical order

# Dump new VCD

- Step through debugger
- Every time the VCD file should be updated → click → write VCD file icon on toolbar

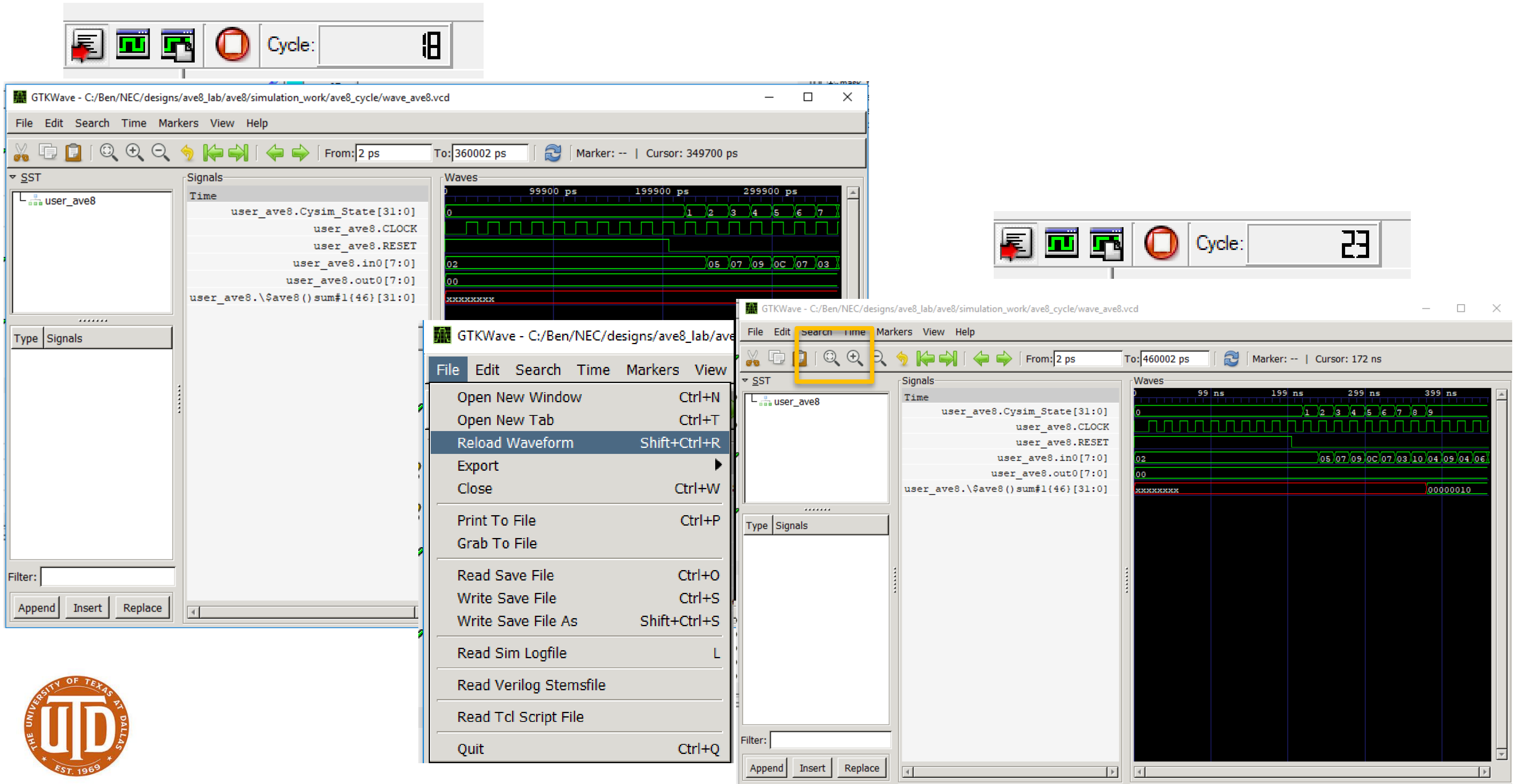




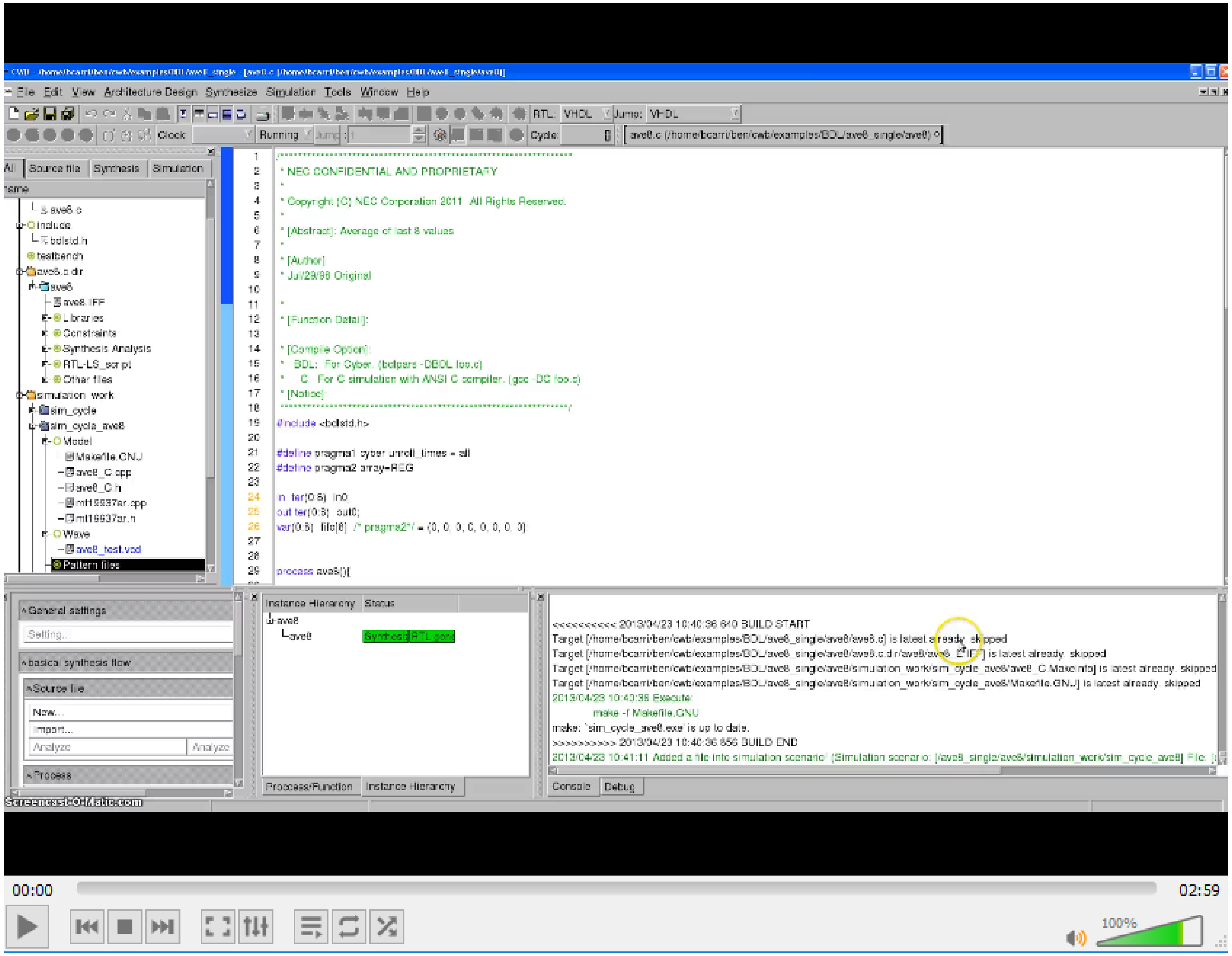
ave8_test.vcd 2018
wave_ave8.vcd 2018

# Updating Waveform

1. Advance debugger
2. Write/update VCD file
3. Reload open VCD viewer
4. Re-size VCD viewer


Cycle: 18
GTKWave - C:/Ben/NEC/designs/ave8_lab/ave8/simulation_work/ave8_cycle/wave_ave8.vcd
File Edit Search Time Markers View Help
From: 2 ps To: 360002 ps Marker: -- | Cursor: 349700 ps
SST
user_ave8
Type Signals
Filter:
Append Insert Replace
Signals
Time
user_ave8.Cysim_State[31:0]
user_ave8.CLOCK
user_ave8.RESET
user_ave8.in0[7:0]
user_ave8.out0[7:0]
user_ave8.\$ave8()sum#1{46}[31:0]
Waves
99900 ps 199900 ps 299900 ps
Open New Window Ctrl+N
Open New Tab Ctrl+T
Reload Waveform Shift+Ctrl+R
Export
Close Ctrl+W
Print To File Ctrl+P
Grab To File
Read Save File Ctrl+O
Write Save File Ctrl+S
Write Save File As Shift+Ctrl+S
Read Sim Logfile L
Read Verilog Stemsfile
Read Tcl Script File
Quit Ctrl+Q
Cycle: 23
From: 2 ps To: 460002 ps Marker: -- | Cursor: 172 ns
99 ns 199 ns 299 ns 399 ns
xxxxxxxx 00000010

# CWB Debugger Video

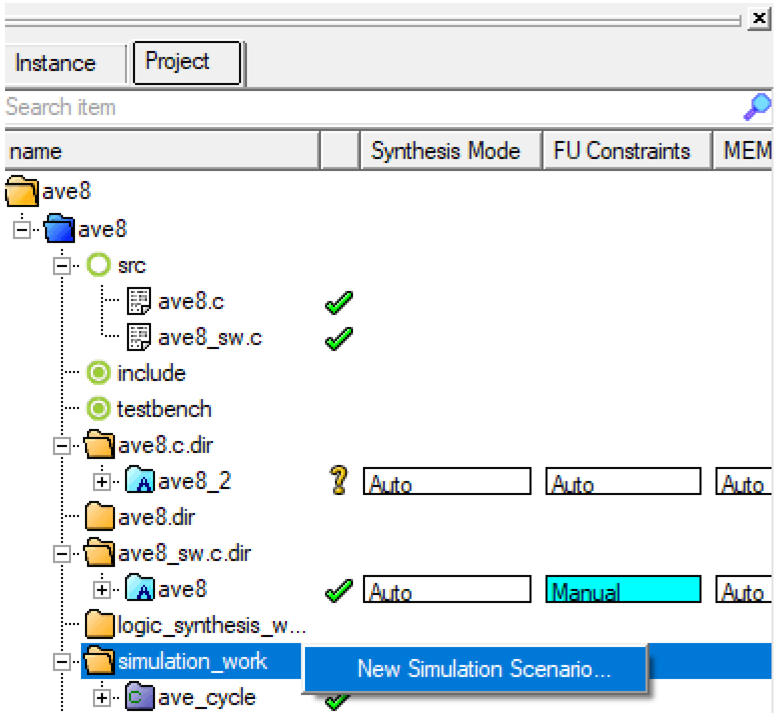


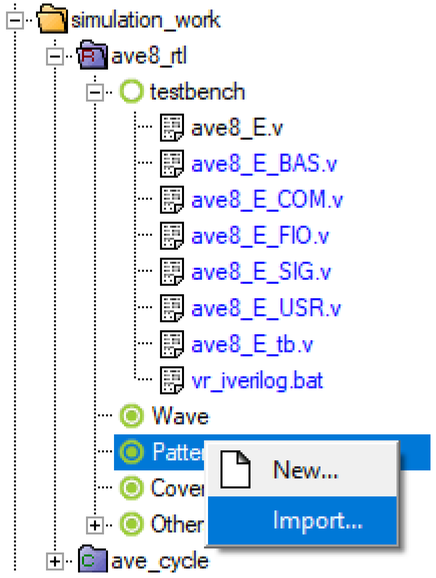
```
/*************************************************************
 * NEC CONFIDENTIAL AND PROPRIETARY
 *
 * Copyright (C) NEC Corporation 2011  All Rights Reserved.
 *
 * [Abstract]: Average of last 8 values
 *
 * [Author]
 * Jul/29/98 Original

 *
 * [Function Detail]:

 * [Compile Option]:
 *   BDL:  For Cyber. (bdlpars -DBDL foo.c)
 *     C:  For C simulation with ANSI C compiler. (gcc -DC foo.c)
 * [Notice]:
 *************************************************************/
#include <bdlstd.h>

#define pragma1 cyber unroll_times = all
#define pragma2 array=REG

in  ter(0:8)  in0;
out ter(0:8)  out0;
var(0:8)  file[8]  /* pragma2*/ = {0, 0, 0, 0, 0, 0, 0, 0};


process ave8(){
```

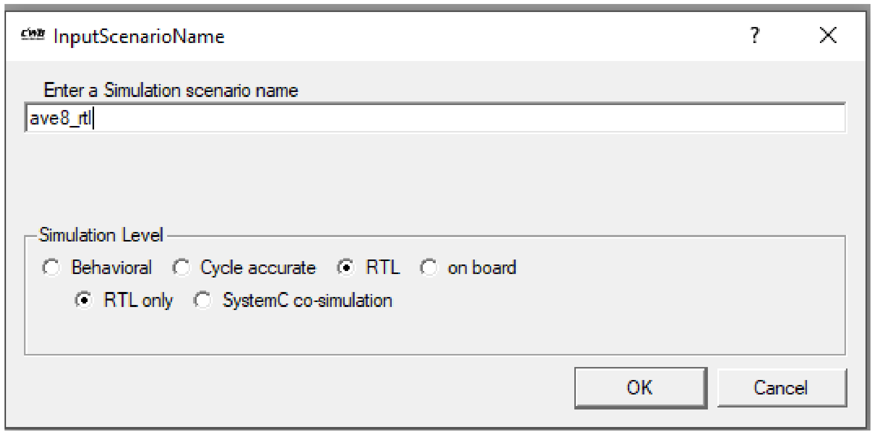
```
<<<<<<<<<< 2013/04/23 10:40:36.640 BUILD START
Target [/home/bcarri/ben/cwb/examples/BDL/ave8_single/ave8/ave8.c] is latest already. skipped
Target [/home/bcarri/ben/cwb/examples/BDL/ave8_single/ave8/ave8.c.dir/ave8/ave8.IFF] is latest already. skipped
Target [/home/bcarri/ben/cwb/examples/BDL/ave8_single/ave8/simulation_work/sim_cycle_ave8/ave8_C.Makeinfo] is latest already. skipped
Target [/home/bcarri/ben/cwb/examples/BDL/ave8_single/ave8/simulation_work/sim_cycle_ave8/Makefile.GNU] is latest already. skipped
2013/04/23 10:40:36 Execute:
           make -f Makefile.GNU
make: `sim_cycle_ave8.exe' is up to date.
>>>>>>>>>> 2013/04/23 10:40:36.656 BUILD END
2013/04/23 10:41:11 Added a file into simulation scenario [Simulation scenario: [/ave8_single/ave8/simulation_work/sim_cycle_ave8] File: [
```

# RTL Simulation

- Same process as for cycle-accurate simulation
- Takes as inputs also .clv and .tlv files






InputScenarioName

Enter a Simulation scenario name

ave8_rtl

Simulation Level

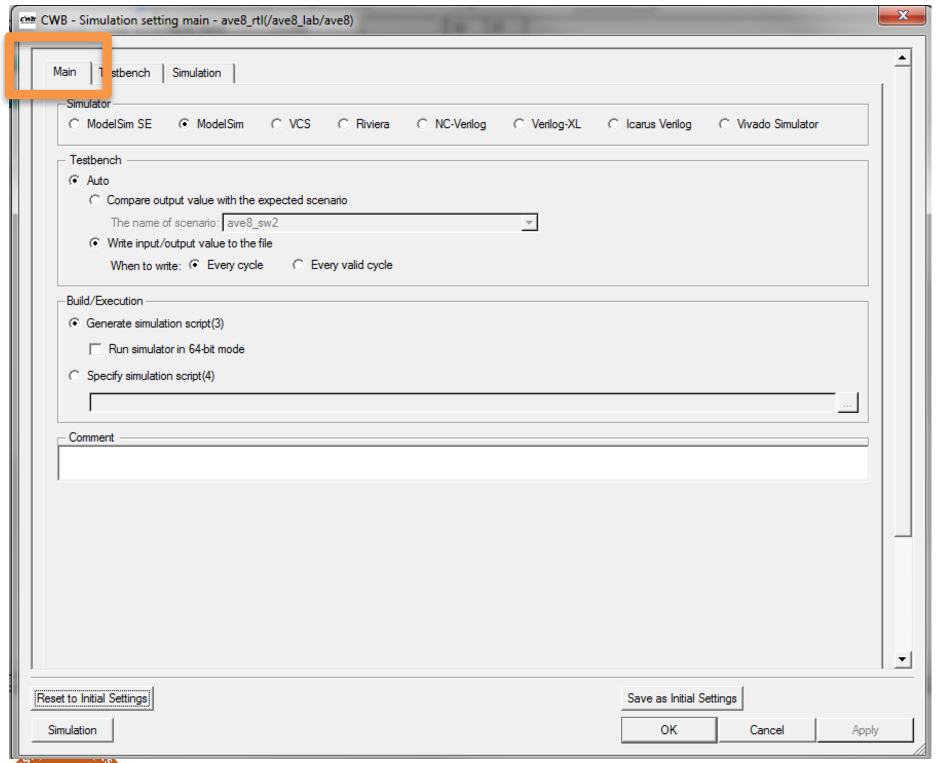
Behavioral [✓ RTL] Cycle accurate on board

[✓ RTL only] SystemC co-simulation

OK Cancel

# Select Inputs

- Need to specify which third part RTL simulator to use
- Random
- From pattern file (clv/tlv)
- Data radix


CWB - Simulation setting main - ave8_rtl(/ave8_lab/ave8)
Main
T stbench
Simulation
Reset to Initial Settings
Save as Initial Settings
Simulation
OK
Cancel
Apply

Simulator

ModelSim SE [✓ ModelSim] VCS Riviera NC-Verilog Verilog-XL Icarus Verilog Vivado Simulator

Testbench

[✓ Auto]

Compare output value with the expected scenario

The name of scenario: ave8_sw2

[✓ Write input/output value to the file]

When to write: [✓ Every cycle] Every valid cycle

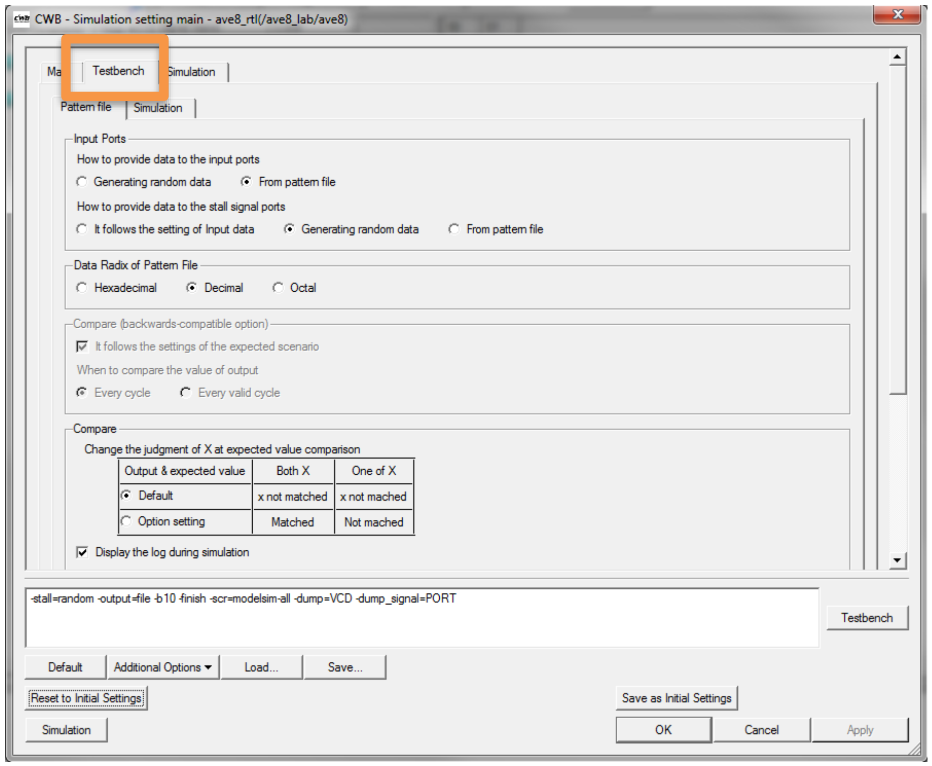
Build/Execution

[✓ Generate simulation script(3)]

Run simulator in 64-bit mode

Specify simulation script(4)

Comment



Input Ports

How to provide data to the input ports

Generating random data [✓ From pattern file]

How to provide data to the stall signal ports

It follows the setting of Input data [✓ Generating random data] From pattern file

Data Radix of Pattern File

Hexadecimal [✓ Decimal] Octal

Compare (backwards-compatible option)

[✓ It follows the settings of the expected scenario]

When to compare the value of output

[✓ Every cycle] Every valid cycle

Compare

Change the judgment of X at expected value comparison

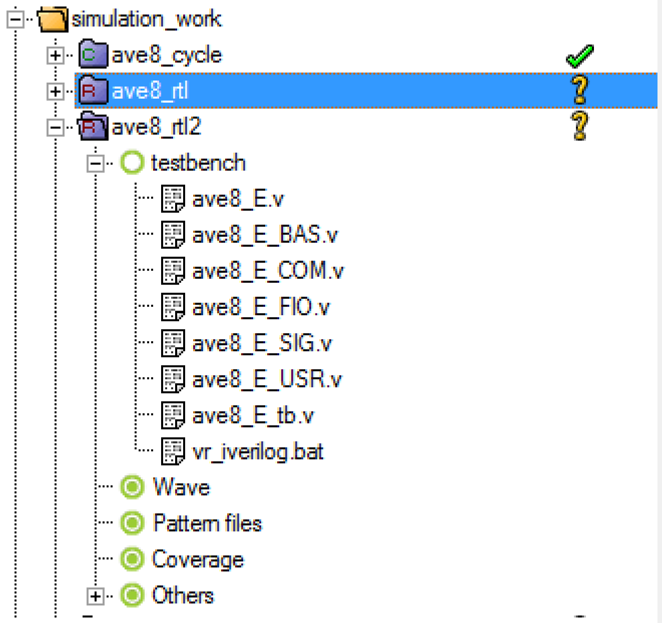
| Output & expected value | Both X | One of X |
|---|---|---|
| [✓ Default] | x not matched | x not mached |
| Option setting | Matched | Not mached |

[✓ Display the log during simulation]

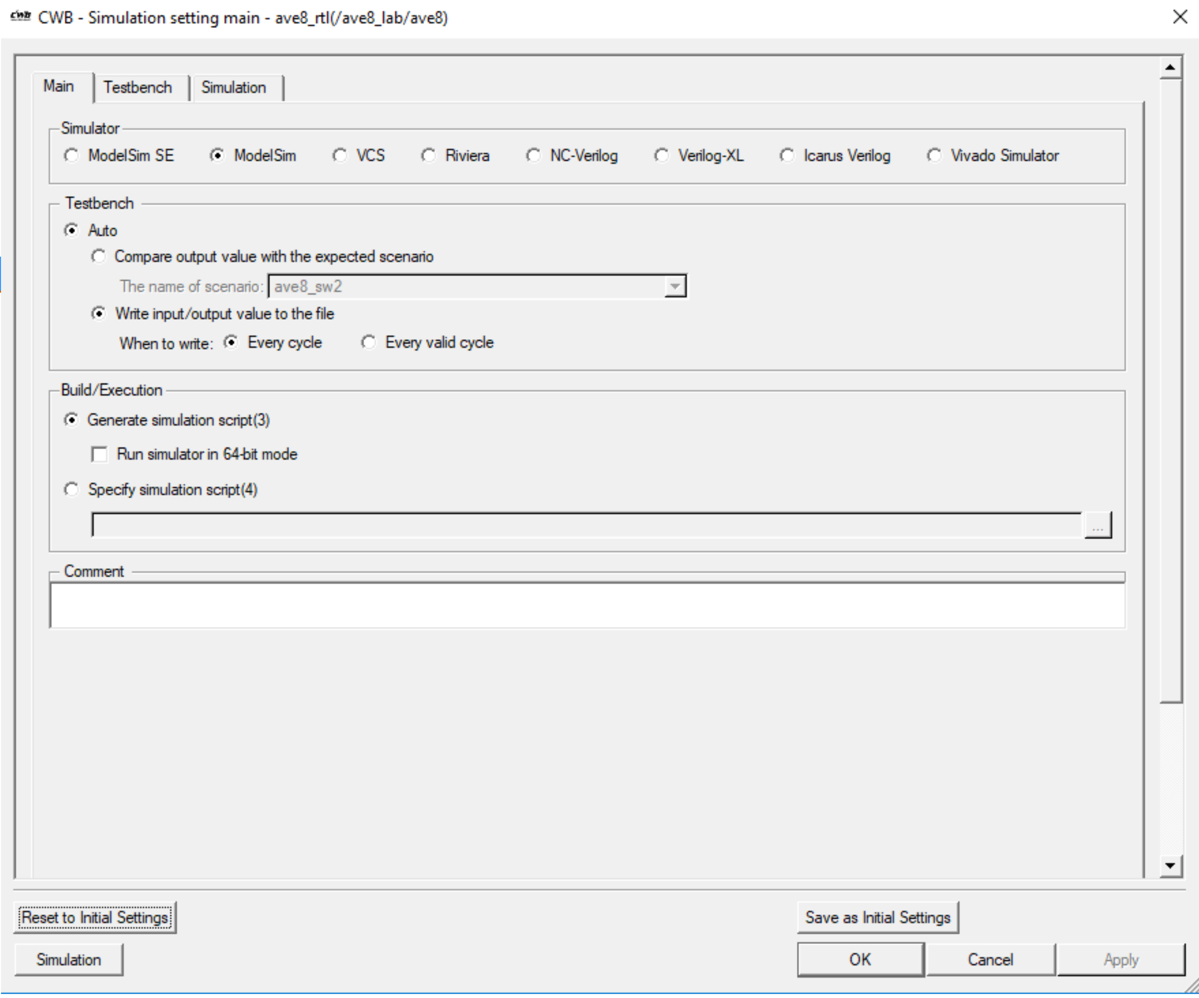
```
-stall=random -output=file -b10 -finish -scr=modelsim-all -dump=VCD -dump_signal=PORT
```

# RTL Testbench

- Generates the RTL testbench
- Script to run RTL simulator (based on simulator selected)
  - E.g., vr_iverilog.bat



```
simulation_work
  ave8_cycle        ✓
  ave8_rtl          ?
  ave8_rtl2         ?
    testbench
      ave8_E.v
      ave8_E_BAS.v
      ave8_E_COM.v
      ave8_E_FIO.v
      ave8_E_SIG.v
      ave8_E_USR.v
      ave8_E_tb.v
      vr_iverilog.bat
    Wave
    Pattern files
    Coverage
    Others
```



CWB - Simulation setting main - ave8_rtl(/ave8_lab/ave8)

Main | Testbench | Simulation

Simulator: ModelSim SE [✓ ModelSim] VCS Riviera NC-Verilog Verilog-XL Icarus Verilog Vivado Simulator

Testbench:
- [✓ Auto]
  - Compare output value with the expected scenario
    - The name of scenario: ave8_sw2
  - [✓ Write input/output value to the file]
    - When to write: [✓ Every cycle] Every valid cycle

Build/Execution:
- [✓ Generate simulation script(3)]
  - ☐ Run simulator in 64-bit mode
- Specify simulation script(4)

Comment:

## Conclusions

- CyberWorkBench
  - Parsers
    - Cpars
    - Bdlpars
    - Scpars
  - Bdltran
    - Clock period (-c)
    - Scheduling mode (-s, -sN, -Zzpipeline)
  - RTL generator
    - Veriloggen
    - Vhdlgen
  - Logic Synthesis using third party tools
    - Synopsys DC
    - Intel Quartus
  - Verification
    - Cycle-accurate model generator (cmscgen)
    - Makefile generator (mkmfsim)
    - .clv and .tlv files
    - Source code debugger
    - RTL testbench generator (tbgen)