



---

**rf engines**

---

(v1.00) 28-Nov-2003

---

Product Specification and User Guide  
Integer To Single Precision Format Converter Core

---

---

PART NUMBER  
FC-64IS-XV2-4-100

---

© rf engines limited

all rights reserved

<http://www.rfel.com>

Tel: +44(0)1983 550330



**TABLE OF CONTENTS**

Overview..... 3  
Core Version..... 3  
General description..... 3  
Core Output Format..... 3  
Core Latency ..... 4  
Core Interface Description ..... 4  
Place and Route Map report..... 5  
Place and Route Timing Report..... 6  
Simulated Power Dissipation Report ..... 7  
Verification..... 7  
Tools ..... 7  
Delivered File Directory Structure..... 8  
Using The Core..... 9  
Using The VHDL Test Bench..... 9



---

**OVERVIEW**

This document describes the RF Engines Ltd (RFEL) 64-bit Signed Integer to IEEE-754 32-bit Single Precision Floating Point Format Converter core. The core is provided in EDIF netlist form as a component.

---

**CORE VERSION**1.00

---

**GENERAL DESCRIPTION**

The core converts 64-bit Signed Integer input values into 32-bit IEEE-754 Single Precision output values. The conversion process is pipelined so that a continuous series of values can be converted at the clock rate applied to the core. A 'nan' control input is provided to force the output of the core to a 'Not A Number' (NaN) value of all ones. This feature may be used to provide a unique 'sync' or 'idle' value that can be easily interpreted by down-stream processes.

---

**CORE OUTPUT FORMAT**

The output of the core is compliant with IEEE-754 Single-Precision format, and is shown in Table 1 below.

Bit Range	31	30:23	22:0
Purpose	Sign 'S'	Exponent 'E'	Mantissa 'M'

**Table 1: 'integer\_to\_single' Output Bit Format**

The Floating-Point value 'V' represented by the 32-bit output vector can be determined as follows:

- If all bits are '1', then  $V = \text{NaN}$ .
- If  $0 < E < 255$ , then  $V = (-1)^S * 2^{(E-127)} * (1 + M / 2^{23})$
- If  $E = 0$  and  $M = 0$  and  $S = 0$ , then  $V = 0$

Note: Many other output values are valid within the IEEE-754 Single-Precision format, but are not shown since they cannot be output by the core.

## CORE LATENCY

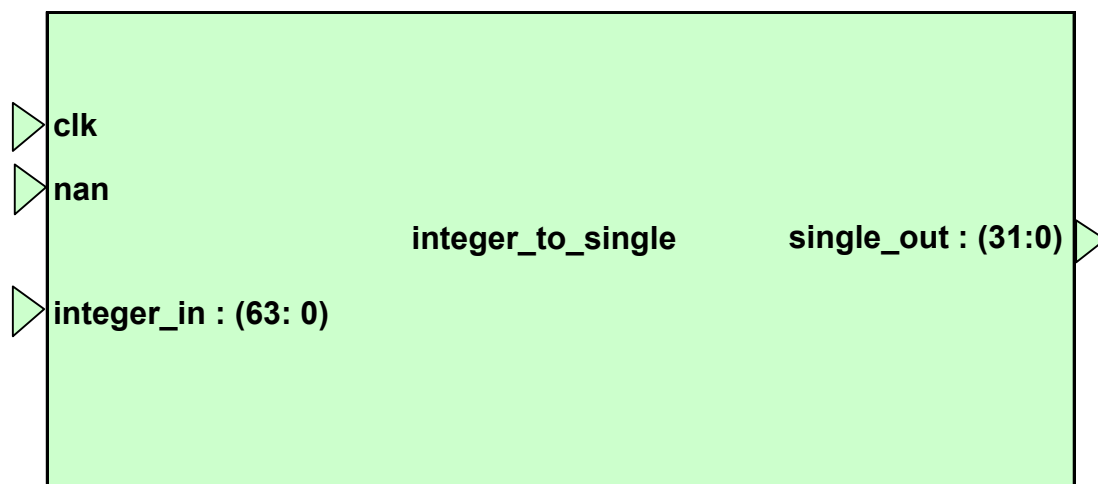
Latency, defined as the time from when a 64-bit Signed Integer value is clocked into the core to the time when the corresponding 32-bit Single-Precision converted result is output from the core, is 5 clock periods. This 5-clock period latency also applies to the 'nan' input signal.

## CORE INTERFACE DESCRIPTION

All core inputs should be synchronised to the 'clk' signal. All core outputs are synchronised by the 'clk' signal.

Signal	Direction	Type	Width	Function
clk	IN	Std logic	1 bit	Core clock.
nan	IN	Std logic	1 bit	Active-high signal, forces ' <i>single_out</i> ' to be all ones (defined as NaN in IEEE-754 standard).
integer_in	IN	Std logic vector	(63 downto 0)	Signed 2's complement input.
single_out	OUT	Std logic vector	(31 downto 0)	IEEE-754 Single Precision Floating-Point output.

**Table 2: 'integer\_to\_single' Interface Specification**



**Figure 1: 'integer\_to\_single' Symbol**



**PLACE AND ROUTE MAP REPORT**

Release 6.1.02i Map G.25a  
Xilinx Mapping Report File for Design 'integer\_to\_single'

Design Information  
-----

Command Line : map -u -p xc2v80-4fg256 pandr\_dump\integer\_to\_single.ngd -o  
pandr\_dump\integer\_to\_single\_map.ncd pandr\_dump\integer\_to\_single.pcf  
Target Device : x2v80  
Target Package : fg256  
Target Speed : -4  
Stepping Level : 1  
Mapper Version : virtex2 -- \$Revision: 1.16 \$  
Mapped Date : Tue Nov 25 12:15:04 2003

Design Summary  
-----

Number of errors: 0  
Number of warnings: 142  
Logic Utilization:  
Number of Slice Flip Flops: 228 out of 1,024 22%  
Number of 4 input LUTs: 469 out of 1,024 45%  
Logic Distribution:  
Number of occupied Slices: 315 out of 512 61%  
Number of Slices containing only related logic: 315 out of 315 100%  
Number of Slices containing unrelated logic: 0 out of 315 0%  
\*See NOTES below for an explanation of the effects of unrelated logic  
Total Number 4 input LUTs: 475 out of 1,024 46%  
Number used as logic: 469  
Number used as a route-thru: 4  
Number used as Shift registers: 2

Total equivalent gate count for design: 5,555  
Peak Memory Usage: 65 MB

NOTES:

Related logic is defined as being logic that shares connectivity - e.g. two LUTs are "related" if they share common inputs. When assembling slices, Map gives priority to combine logic that is related. Doing so results in the best timing performance.

Unrelated logic shares no connectivity. Map will only begin packing unrelated logic into a slice once 99% of the slices are occupied through related logic packing.

Note that once logic distribution reaches the 99% level through related logic packing, this does not mean the device is completely utilized. Unrelated logic packing will then begin, continuing until all usable LUTs and FFs are occupied. Depending on your timing budget, increased levels of unrelated logic packing may adversely affect the overall timing performance of your design.



**PLACE AND ROUTE TIMING REPORT**

-----  
Release 6.1.02i Trace G.25a  
Copyright (c) 1995-2003 Xilinx, Inc. All rights reserved.

trce -e 3 -l 3 -xml pandr\_dump\integer\_to\_single  
pandr\_dump\integer\_to\_single.ncd -o pandr\_dump\integer\_to\_single.twr  
pandr\_dump\integer\_to\_single.pcf

Design file: integer\_to\_single.ncd  
Physical constraint file: integer\_to\_single.pcf  
Device, speed: xc2v80,-4 (PRODUCTION 1.116 2003-09-30, STEPPING level 1)  
Report level: error report

Environment Variable	Effect
NONE	No environment variables were set

-----  
INFO:Timing:2752 - To get complete path coverage, use the unconstrained paths option. All paths that are not constrained will be reported in the unconstrained paths section(s) of the report.

=====  
Timing constraint: TS\_clk = PERIOD TIMEGRP "system\_clk" 8.333 nS HIGH  
50.000000 % ;

15623 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)  
Minimum period is 8.220ns.

-----  
All constraints were met.

Data Sheet report:

-----  
No constraints were found to generate data for the Data Sheet Report section. Use the Advanced Analysis (-a) option or generate global constraints for each clock, its pad to setup and clock to pad paths, and a pad to pad constraint.

Timing summary:

-----  
Timing errors: 0 Score: 0

Constraints cover 15623 paths, 0 nets, and 1752 connections

Design statistics:  
Minimum period: 8.220ns (Maximum frequency: 121.655MHz)

Analysis completed Tue Nov 25 12:16:00 2003



**SIMULATED POWER DISSIPATION REPORT**

-----  
Release 6.1.02i - XPower SoftwareVersion:G.25a  
Copyright (c) 1995-2003 Xilinx, Inc. All rights reserved.  
Design: integer\_to\_single\_wrapper  
Preferences: working\pandr\integer\_to\_single\_wrapper.pcf  
VCD File: working/vital\_sim/integer\_to\_single.vcd  
Part: 2v80fg256-4  
Data version: ADVANCED,v1.0,07-31-02

Power summary:	I (mA)	P (mW)
-----		
Total estimated power consumption:		495
---		
Vccint 1.50V:	108	161
Vccaux 3.30V:	100	330
Vcco33 3.30V:	1	3
---		
Clocks:	0	0
Inputs:	6	9
Logic:	37	56
Outputs:		
Vcco33	0	0
Signals:	14	21
---		
Quiescent Vccint 1.50V:	50	75
Quiescent Vccaux 3.30V:	100	330
Quiescent Vcco33 3.30V:	1	3

Note: The core power is indicated by the 'Logic' power only (56mW in this case). All other power contributions are due to the core being placed within a chip wrapper (Inputs, Outputs, Quiescents etc). The power simulation shown above was run using a clock period of 10 ns (100MHz) and 1000 random input samples.

**VERIFICATION**

Verification of the core is achieved using the supplied VHDL test bench, and compiled behavioural and VITAL models.

The test bench applies random stimulus to the core, and automatically checks that the converted results are correct. An error flag is latched active if any converted result is not correct.

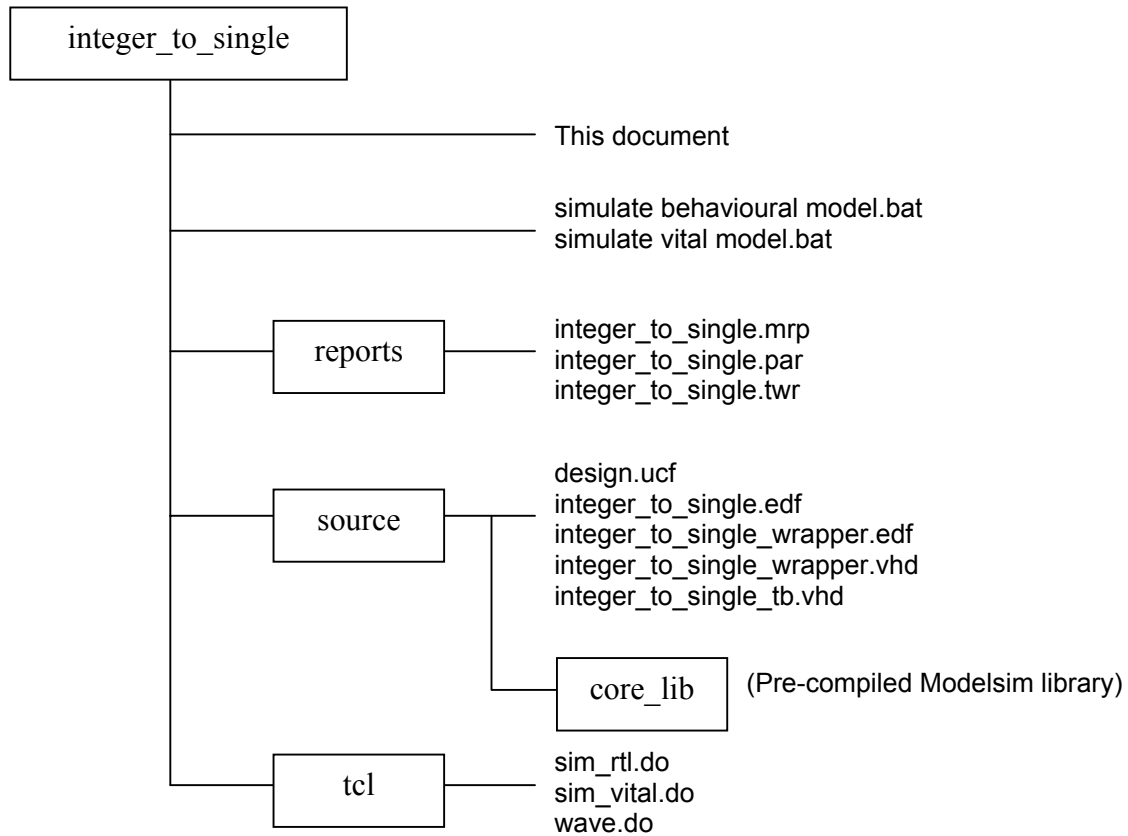
**TOOLS**

The following tools and versions were used to generate this delivery.

- Modelsim PE Version: 5.4e
- Leonardo Spectrum Level 2 Version: LS2002e\_16
- Xilinx ISE Release Version: 6.1.02i

**DELIVERED FILE DIRECTORY STRUCTURE**

The delivered file structure is shown in Figure 2 below.

**Figure 2: Delivered File Directory Structure**

## USING THE CORE

The following steps should be followed to ensure the EDIF netlist is included in the final design:

1. Include core instance(s) within the target design (using *'integer\_to\_single/source/integer\_to\_single\_wrapper.vhd'* as a reference).
  2. Place the core EDIF file *'integer\_to\_single/source/integer\_to\_single.edf'* in the same directory as the final design EDIF, or add *integer\_to\_single/source* to the macro search path in the process properties dialog box (right mouse click over implement design > properties) in XilinxISE.
  3. Use the parameters contained in the constraints file *'integer\_to\_single/source/design.ucf'* as a reference.
  4. Continue through normal place and route using the XilinxISE tools and the core will be integrated into the design.
- 

## USING THE VHDL TEST BENCH

The VHDL test bench provides control and data stimuli to either the pre-compiled behavioural VHDL model, or a VITAL VHDL model that can be created by the XilinxISE tools. Random stimuli and expected results are generated within the test bench. The test bench produces a latched pass / fail indication by automatically comparing the core outputs against the expected values during simulation.

### Running the Test Bench for RTL Simulation

Running the DOS BAT file *'integer\_to\_single/simulate\_behavioural\_model.bat'* will perform behavioural simulation. This script performs the following:

1. If the directory *'integer\_to\_single/working/behavioural\_sim'* does not already exist (as will be the case the first time the script is run), then it is created.
2. The pre-compiled behavioural VHDL model library *'integer\_to\_single/source/core\_lib'* is copied into *'integer\_to\_single/working/behavioural\_sim'*.
3. ModelSim is launched in GUI mode, and the TCL script *'integer\_to\_single/tcl/sim\_rtl.do'* is run. ModelSim's transcript path is set to *'integer\_to\_single/working/behavioural\_sim/transcript.txt'* so that it may be viewed after simulation if required.
4. When ModelSim has launched, the TCL script *'integer\_to\_single/tcl/sim\_rtl.do'* performs the following:
  - a. ModelSim's working directory is set to *'integer\_to\_single/working/behavioural\_sim'*.
  - b. A working library *'integer\_to\_single/working/behavioural\_sim/lib'* is created and mapped to as *'work'*.
  - c. The copied version of the pre-compiled behavioural VHDL model library *'core\_lib'* is mapped to as *'integer\_to\_single\_library'*, and is updated (vcom -refresh) to the current ModelSim library format from it's delivered ModelSim 5.4e format.
  - d. The VHDL core wrapper *'integer\_to\_single/source/integer\_to\_single\_wrapper.vhd'* and the VHDL test bench *'integer\_to\_single/source/integer\_to\_single\_tb.vhd'* are compiled, and the test bench is loaded.

- e. The TCL script *'integer\_to\_single/tcl/wave.do'* is run to open a Wave window with the relevant test bench signals and variables required to check the results of the simulation when completed.
- f. The simulation is run until all of the test bench processes have reached their *'wait'* states.

When the simulation is complete, the traces shown in Figure 3 should be displayed in the Wave window.

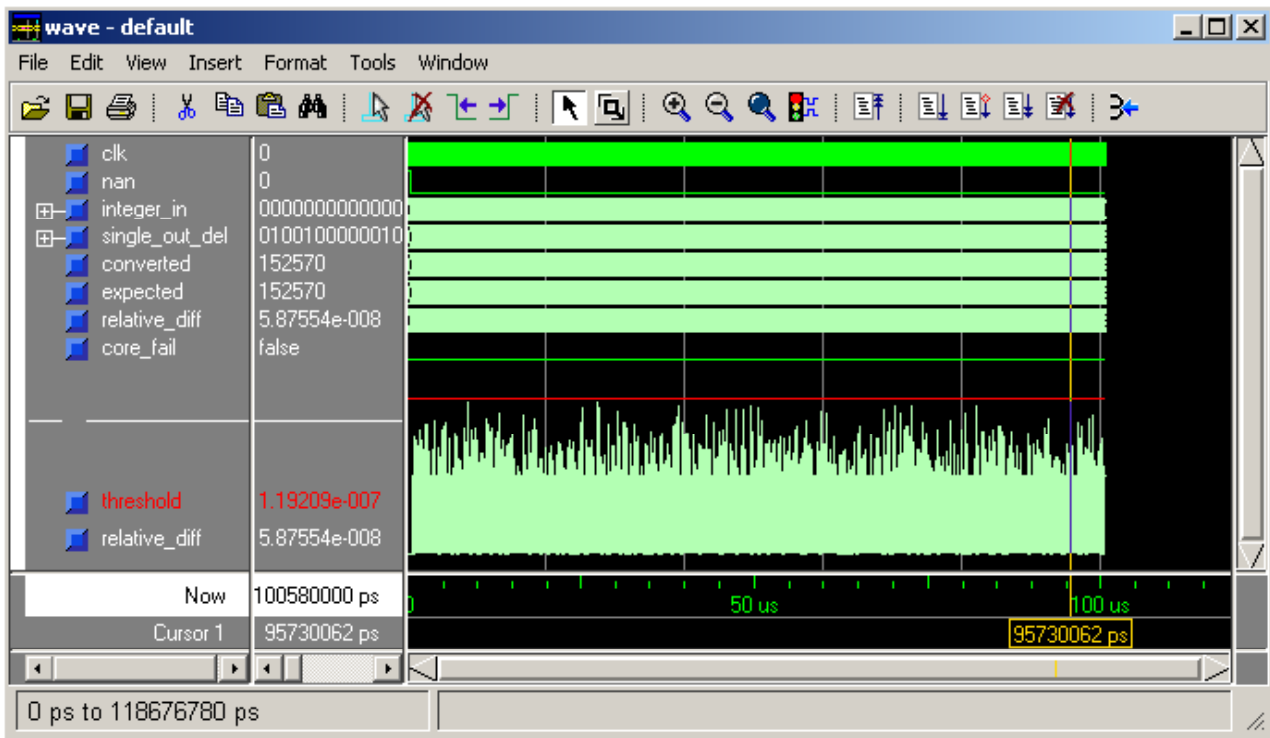


Figure 3: Behavioural Simulation Wave Window

The traces show that the *'core\_fail'* signal is *'false'* at the end of the simulation, indicating a *'pass'* condition. Also shown are all of the core inputs and outputs, as well as the relative difference between the single precision result of the core and the double precision expected result derived within the test bench. The red trace shows the relative difference *'threshold'* used within the test bench to establish a pass or failure.

### Running the Test Bench for VITAL Simulation

Before VITAL simulation can take place, the user is required to edit and update the script *'integer\_to\_single/tcl/sim\_vital.do'* to reflect the location of the compiled Xilinx *'simprim'* library. The following text within the script should be edited appropriately:

```
#####
#### MAP TO YOUR COMPILED SIMPRIM LIBRARY HERE ....
vmap simprim E:/xilinxise6/vhdl/src/compiled_simprim
#####
```

When this is done, running the DOS BAT file *'integer\_to\_single/simulate\_vital\_model.bat'* will perform VITAL simulation. This script performs the following:

1. If the directory '*integer\_to\_single/working/pandr*' does not already exist (as will be the case the first time the script is run), then the script performs the following:
  - a. The directory '*integer\_to\_single/working/pandr*' is created. This is where all of the wrapped core place and route files will be written to.
  - b. '*integer\_to\_single/source/integer\_to\_single\_wrapper.edf*' is implemented using the XilinxISE tools. This netlist is the core wrapper design, synthesised with I/O pads, I/O registers and a clock buffer. The core '*integer\_to\_single/source/integer\_to\_single.edf*' is automatically included within the wrapper because '*integer\_to\_single/source*' has been specified as a macro search path within the script.
  - c. After implementation, the script checks to see which version of XilinxISE is currently installed. If version 5 is present, then the '*ngd2vhd*' function is used to create the VHDL VITAL simulation model, otherwise it is assumed that the user has version 6 or later, and the newer '*netgen*' function is used instead.
2. If the directory '*integer\_to\_single/working/vital\_sim*' does not already exist (as will be the case the first time the script is run), then it is created.
3. ModelSim is launched in GUI mode, and the TCL script '*integer\_to\_single/tcl/sim\_vital.do*' is run. ModelSim's transcript path is set to '*integer\_to\_single/working/vital\_sim/transcript.txt*' so that it may be viewed after simulation if required.
4. '*simulate vital model.bat*' now waits until the file '*integer\_to\_single/working/vital\_sim/done*' is written by ModelSim to indicate that simulation is complete.
5. When ModelSim has launched, the TCL script '*integer\_to\_single/tcl/sim\_vital.do*' performs the following:
  - a. ModelSim's working directory is set to '*integer\_to\_single/working/vital\_sim*'.
  - b. A working library '*integer\_to\_single/working/vital\_sim/lib*' is created and mapped to as '*work*'.
  - c. The compiled '*simprim*' library is mapped as specified by the user.
  - d. The VITAL model '*integer\_to\_single/working/pandr/integer\_to\_single\_wrapper\_vital.vhd*' and the VHDL test bench '*integer\_to\_single/source/integer\_to\_single\_tb.vhd*' are compiled.
  - e. The SDF file '*integer\_to\_single/working/pandr/integer\_to\_single\_wrapper\_vital.sdf*' is applied to the unit under test '*uut*' when the test bench is loaded for simulation.
  - f. The TCL script '*integer\_to\_single/tcl/wave.do*' is run to open a Wave window with the relevant test bench signals and variables required to check the results of the simulation when completed.
  - g. The simulation is run for 1 $\mu$ s to flush the core with random data, then ModelSim is instructed to create a VCD file containing signal state information of all signals within the unit under test '*uut*'. The simulation is run for a further 10 $\mu$ s to allow 1000 samples of random data to pass through the core and be logged within the VCD file. When the simulation is complete, the file '*integer\_to\_single/working/vital\_sim/done*' is created to tell '*integer\_to\_single/simulate vital model.bat*' to proceed with power analysis.
6. When '*integer\_to\_single/working/vital\_sim/done*' has been written, the XilinxISE '*xpwr*' tool is launched to provide power analysis figures for the simulated VITAL design. The results of this power analysis are written to '*integer\_to\_single/working/vital\_sim/integer\_to\_single.pwr*', and are displayed on the Command Prompt window.

The ModelSim Wave window traces should look like those shown in Figure 4 below.

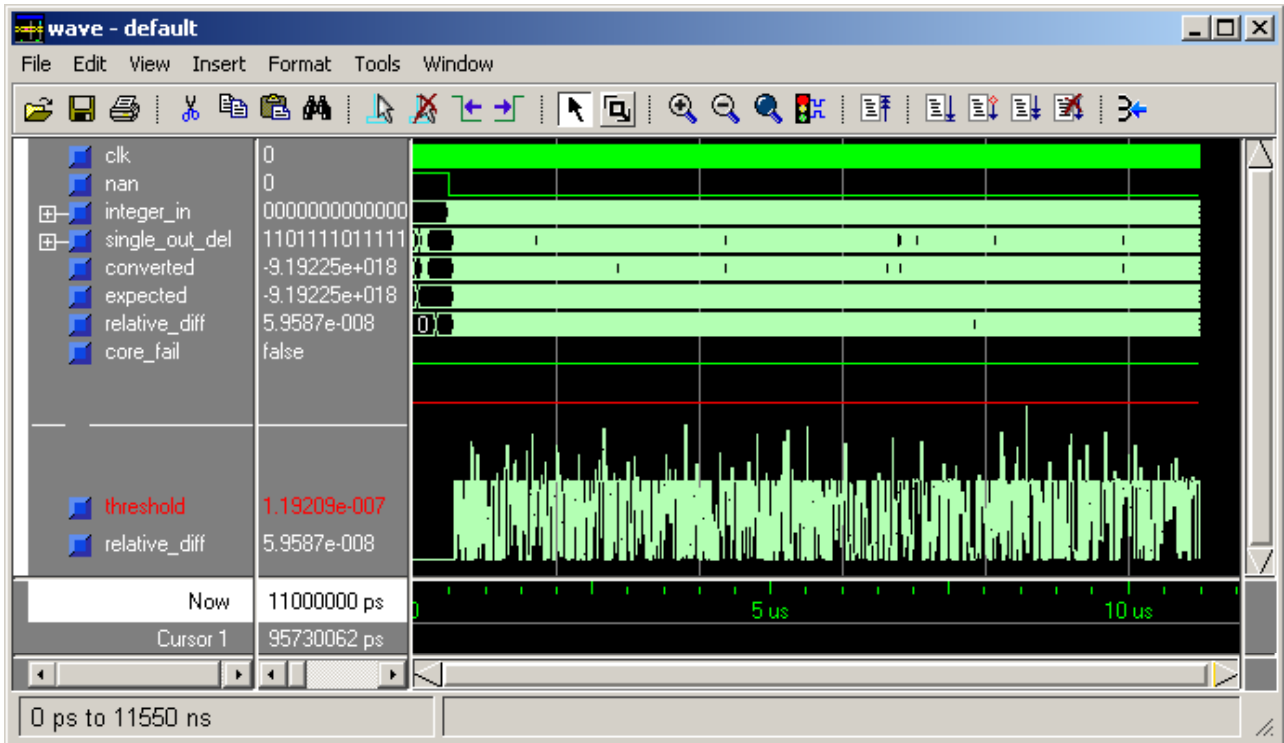


Figure 4: VITAL Simulation Wave Window