

\*\*\*\*\*DRAFT\*\*\*\*\*

**Counter Test SpinalHDL**  
**Verilator Simulation**  
**02/28/21**

\*\*\*\*\*DRAFT\*\*\*\*\*

```
~/SpinalTemplateSbt/src/main/scala/mylib/Counter.scala
```

```
package mylib
```

```
import spinal.core._
```

```
class Counter(width : Int) extends Component{  
  val io = new Bundle{  
    val clear = in Bool  
    val value = out UInt(width bits)  
  }  
  val register = Reg(UInt(width bits)) init(0)  
  register := register + 1  
  when(io.clear){  
    register := 0  
  }  
  io.value := register  
}
```

```
object CounterVerilog {  
  // Let's go  
  def main(args: Array[String]) {  
    SpinalVerilog(new Counter(6))  
  }  
}
```

```
~/SpinalTemplateSbt/
```

```
sbt run
```

```
[5] mylib.CounterVerilog
```

```
~/SpinalTemplateSbt/
```

```
// Generator : SpinalHDL v1.4.3   git head : adf552d8f500e7419fff395b7049228e4bc5de26
```

```
// Component : Counter
```

```
// Git hash : 95d6bcd599277f65c1cb6acfe7a971d8d9b90c2f
```

```
module Counter (  
  input      io_clear,  
  output [5:0] io_value,  
  input      clk,  
  input      reset  
);
```

```

reg    [5:0]  register_1;

assign io_value = register_1;
always @ (posedge clk or posedge reset) begin
    if (reset) begin
        register_1 <= 6'h0;
    end else begin
        register_1 <= (register_1 + 6'h01);
        if(io_clear)begin
            register_1 <= 6'h0;
        end
    end
end
end

endmodule

/////////////////////////////////////////////////////////////////
//
// Filename:   helloworld_tb.cpp
//
// Project:    Verilog Tutorial Example file
//
// Purpose:    To demonstrate a Verilog main() program that calls a local
//              serial port co-simulator.
//
// Creator:    Dan Gisselquist, Ph.D.
//              Gisselquist Technology, LLC
//
/////////////////////////////////////////////////////////////////
//
// Written and distributed by Gisselquist Technology, LLC
//
// This program is hereby granted to the public domain.
//
// This program is distributed in the hope that it will be useful, but WITHOUT
// ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
// FITNESS FOR A PARTICULAR PURPOSE.
//
/////////////////////////////////////////////////////////////////
//
//
#include <verilatedos.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include <signal.h>
#include "verilated.h"
#include "VCounter.h"

```

```
#include "testb.h"
```

```
int    main(int argc, char **argv) {
    Verilated::commandArgs(argc, argv);
    TESTB<VCounter> *tb
        = new TESTB<VCounter>;

    Verilated::traceEverOn(true); // Verilator must compute traced signals
    VL_PRINTF("Enabling waves...\n");

    tb->opentrace("Counter.vcd");

    for(unsigned clocks=0;clocks < 10000;clocks++) {

        if (clocks==30) tb->m_core->io_clear=1;
        if (clocks==31) tb->m_core->io_clear=0;

        tb->tick();

    }

    printf("\n\nSimulation complete\n");
}
```

```
~/SpinalTemplateSbt/test_Counter
```

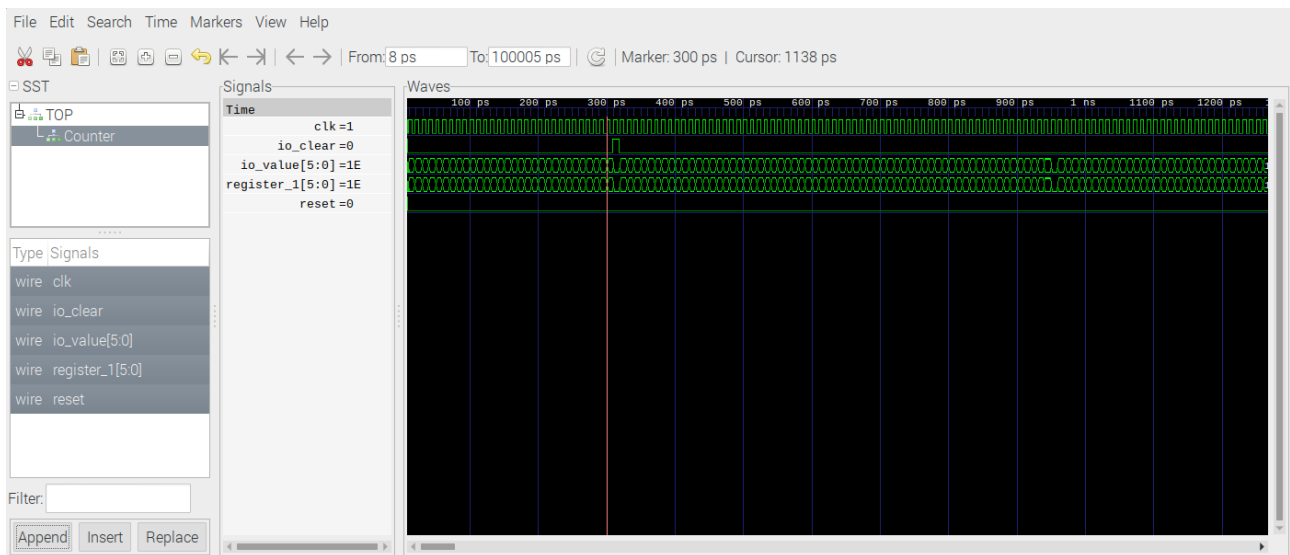
```
verilator -Wall --trace -cc Counter.v --exe --build sim_main.cpp
```

```
~/SpinalTemplateSbt/test_Counter/obj_dir
./VCounter
```

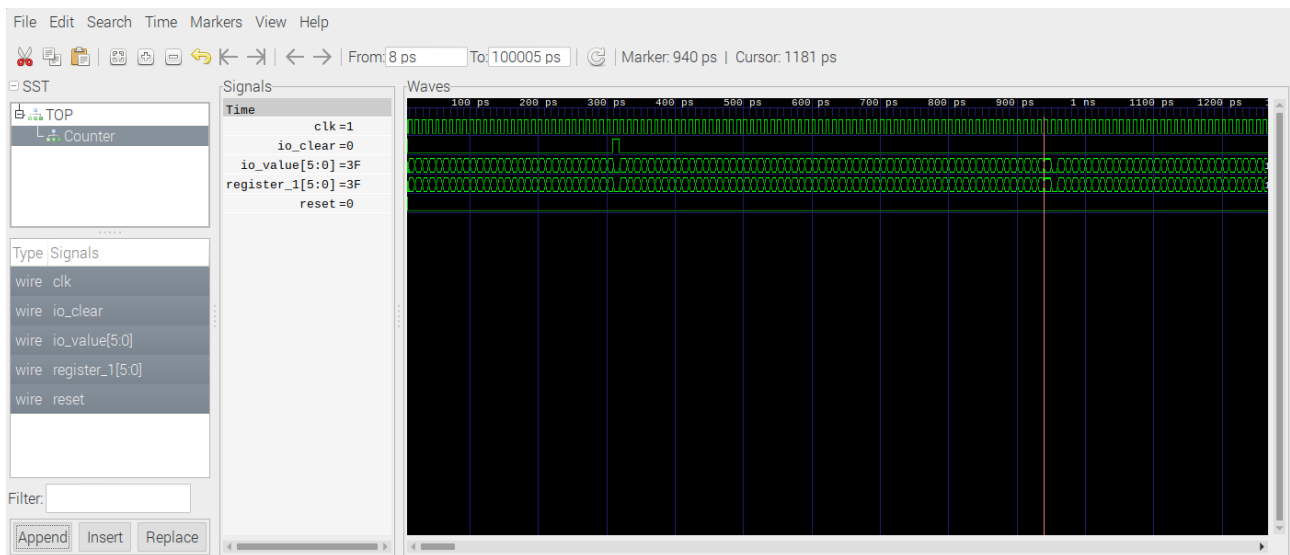
Enabling waves...

Simulation complete

```
gtkwave Counter.vcd
if (clocks==30) tb->m_core->io_clear=1;
if (clocks==31) tb->m_core->io_clear=0;
```



When no io.clear is valid the io.value counts to 0x3F 6 bits.



Setting the reset hi and lo

