

### Key Design Features

- Synthesizable, technology independent VHDL Core
- Test patterns generated as 24-bit RGB-video
- Supports all video resolutions up to  $2^{16} \times 2^{16}$  pixels
- Colour, greyscale or monochrome outputs
- Bars, squares, lines and 'bouncing ball' display
- Option to generate 'blank' output video
- Programmable pattern width and line spacing
- Simple valid-ready output flow control
- Fully configurable output video resolution
- Output pixels generated at 1 pixel/clock
- Compatible with all Zipcores video IP cores

### Applications

- Digital video testing and prototyping
- Generation of a blank video background as a 'canvas' for video overlays
- Simple screen savers

### Generic Parameters

Generic name	Description	Type	Valid Range
tpg_wait	Start up wait time (in clock cycles) before output video is generated	integer	$< 2^{16}$
tpg_mode	Test pattern colour mode	integer	0 : monochrome 1 : greyscale 2 : colour
tpg_type	Test pattern type	integer	0 : bars 1 : squares 2 : hatch 3 : bouncing ball 4 : blank display
tpg_log2w	Log2(width) of bars, squares, ball, hatch spacing etc.	integer	$< 2^{16}$
tpg_ppl	No. of pixels in each output video line	integer	$< 2^{16}$
tpg_lpf	No. of lines in each output video frame	integer	$< 2^{16}$

### Block Diagram

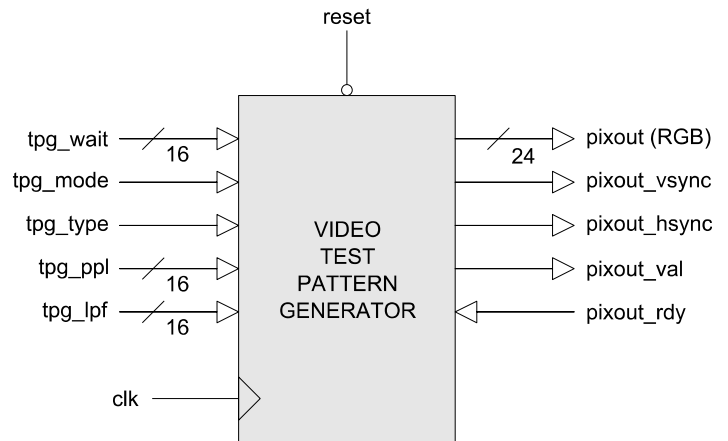


Figure 1: Video test pattern generator

### Pin-out Description

Pin name	I/O	Description	Active state
clk	in	Synchronous clock	rising edge
reset	in	Asynchronous reset	low
pixout [23:0]	out	24-bit pixel out	data
pixout_vsync	out	Vertical sync out	high
pixout_hsync	out	Horizontal sync out	high
pixout_val	out	Output pixel valid	high
pixout_rdy	in	Ready to accept output pixel (handshake signal)	high

### General Description

The TPG module (Figure 1) is a versatile test pattern generator capable of producing a range of test patterns in colour, greyscale and monochrome formats. The module is invaluable in the prototyping stages of digital video systems. In addition, the test pattern generator may be used to provide a blank background display as a 'canvas' for video overlays.

Pixels and syncs are transferred out of the module on a rising clock-edge when *pixout\_val* is high and *pixout\_rdy* is high. The signal *pixout\_vsync* is active high when the first pixel of a frame is output. The signal *pixout\_hsync* is active high when the first pixel of a line is output.

By enabling or disabling the *pixout\_rdy* signal, the flow of pixels out of the test pattern generator may be easily controlled by the downstream module. The test pattern generator has 'infinite' video bandwidth - with output pixels generated on demand.

The video output resolution is controlled by the generic parameters *tpg\_ppl* and *tpg\_lpf*. The colour, type and dimensions of the test pattern are determined by the parameters *tpg\_mode*, *tpg\_type* and *tpg\_log2w*.

### Test pattern dimensions

By controlling the parameters *tpg\_ppl* and *tpg\_lpf*, the output video resolution may be set. The parameter *tpg\_log2w* controls the width or spacing of the bars, squares or lines. For example, when the type is set to 'bars' and *tpg\_log2w* is set to '5', then the width of the bars would be  $2^5$  or 32-pixels. Figure 2 shows the relationship of these parameters to the output display.

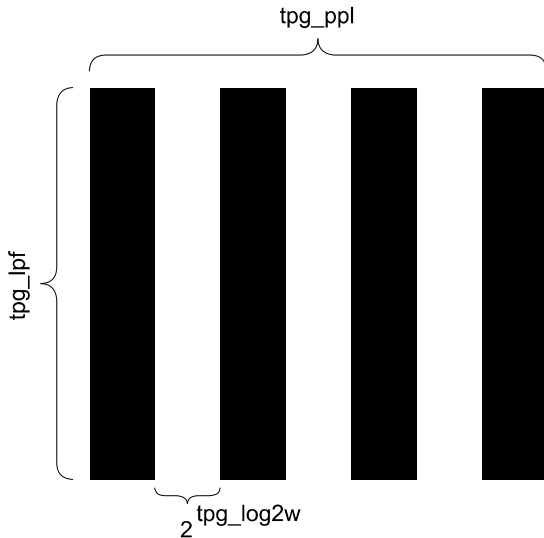


Figure 2: Test pattern dimensions

### Test pattern mode and type

By modifying the test pattern mode and type, the colour and appearance of the test pattern may be controlled. Figure 3 shows the types of test pattern available.

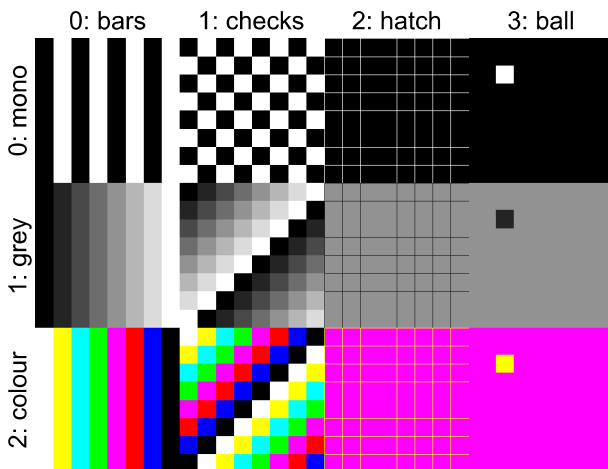


Figure 3: Different test pattern types and colours

In summary, the bar display is a series of vertical bars that extend the width of the display. The width of the bars is determined by the parameter *tpg\_log2w*. The checker-board display is a series of squares, with the width of each square controlled by *tpg\_log2w*. The hatch test pattern features a number of horizontal and vertical lines of 1 pixel in width. In this instance, the spacing between lines is controlled by *tpg\_log2w*. The 'bouncing ball' test pattern is an animated 'ball' that bounces randomly and at varying speeds. The ball test pattern is especially useful for detecting movement artefacts such as motion blur and 'mouse-teeth' in digital video systems. Finally, the type '4' test pattern (not shown) is a plain black background display.

### Functional Timing

RGB output pixels are sampled according to the valid-ready pipeline protocol<sup>1</sup>. Figure 4 shows the signalling at the output of the test pattern generator at the start of a new frame. The first pixel of a new frame begins with *pixout\_vsync* and *pixout\_hsync* asserted high together with the first pixel. The first pixel of a new line begins with *pixout\_hsync* asserted only.

After reset, and after the startup wait time has been satisfied, valid output pixels are generated. Pixels may be held off by asserting *pixout\_rdy* low. As an example, the diagram shows what happens when *pixout\_rdy* is deasserted for one clock cycle. In this case, the output pixels (and syncs if present) are stalled until the ready signal is asserted again.

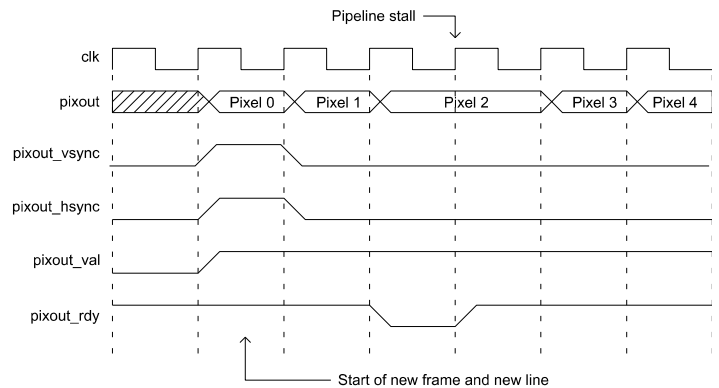


Figure 4: First output pixel of a new frame

### Source File Description

All source files are provided as text files coded in VHDL. There are only two source files required in the test pattern generator design.

Source file	Description
tpg.vhd	Text overlay top-level component
tpg_bench.vhd	Top-level test bench

<sup>1</sup> See application note: app\_note\_zc001.pdf on the Zipcores website for more examples of the valid-ready pipeline protocol

## Functional Testing

An example VHDL testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. tpg.vhd
2. tpg\_bench.vhd

The VHDL testbench instantiates the TPG component and the user may modify the generic parameters as required. In the example testbench, the test pattern generator is configured to give a basic hatched line output at 256x256 pixels in resolution with a line spacing of 32 pixels.

The simulation must be run for at least 10 ms during which time the output pixels and syncs from the test pattern generator are captured in the output file *video\_out.txt*.

The output text file follows a simple format which defines the state of signals: *pixout\_val*, *pixout\_vsync*, *pixout\_hsync* and *pixout* on a clock-by-clock basis. An example file might be the following:

```

1 1 1 FF FF FF # pixel 0 line 0 (start of frame)
1 0 0 FF FF FF # pixel 1 line 0
1 0 0 FF FF FF # pixel 2 line 0
.
.
1 0 1 FF FF 00 # pixel 0 line 1 (start of line)
1 0 0 FF FF 00 # pixel 1 line 1 etc..
  
```

The video output of the simulation is shown in Figure 5. The generic parameters are set to: *tpg\_mode* = 0, *tpg\_type* = 2, *tpg\_log2w* = 5, *tpg\_ppl* = 256 and *tpg\_lpf* = 256.

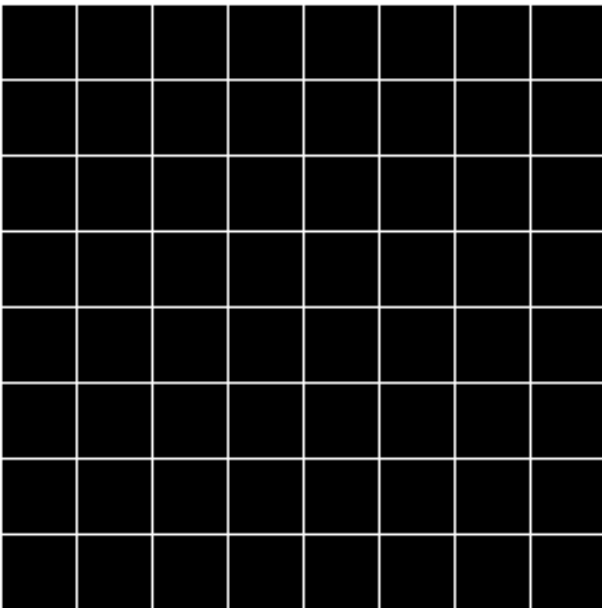


Figure 5: 256x256 hatched line simulation output

## Synthesis

The only file required for synthesis is the file 'tpg.vhd'. The module is totally self contained with no sub-modules.

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx Virtex 5 and the Altera Stratix III series of FPGA devices. The lowest and highest speed grade devices have been chosen in both cases for comparison.

Trial synthesis results are shown in the following tables. The design was synthesized with the generic parameters set as follows: *tpg\_wait* = 100, *tpg\_mode* = 2, *tpg\_type* = 0, *tpg\_log2w* = 4, *tpg\_ppl* = 640, *tpg\_lpf* = 480.

Resource usage is specified after Place and Route.

### VIRTEX 5

Resource type	Quantity used
Slice register	50
Slice LUT	103
Block RAM	0
DSP48	0
Clock frequency (worst case)	300 MHz
Clock frequency (best case)	410 MHz

### STRATIX III

Resource type	Quantity used
Register	51
ALUT	68
Block Memory bit	0
DSP block 18	0
Clock frequency (worse case)	350 MHz
Clock frequency (best case)	415 MHz

## Revision History

Revision	Change description	Date
1.0	Initial revision	09/12/2009
1.1	Updated synthesis results	02/02/2012