

# Nexys 2 Built In Self Test Manual



Turnu Roșu, 19-21, 400388, Cluj-Napoca,  
ROMANIA,  
Phone: +40-364-101806, 101807

Revision: February 20, 2008

## Overview

This document describes project used to compile the Nexys 2 Built In Self Test (BIST) configuration file. This configuration is loaded on the Nexys 2 platform flash during manufacturing and serves for manufacturing testing. After testing, the configuration file remains in the platform Flash on the Nexys 2 boards (until erased/replaced by user) and serves as Demo and self test for final users.

This document does not describe the PC software used for manufacturing tests. However, it explains the communication interface implemented in the FPGA project. Based on information revealed by this document, the user can communicate with the internal components of the BIST project. Digilent Adept Suite Software needs to be used on the PC to communicate with the FPGA BIST project.

The Manufacturing BIST provides access to the following hardware components on the Nexys 2 board:

1. Input devices
  - a. 8x switch (SW7, SW6, SW5, SW4, SW3, SW2, SW1, SW0)
  - b. 4x button (BTN3, BTN2, BTN1, BTN0)
2. Output devices
  - a. 8x User LEDs (LD7, LD6, LD5, LD4, LD3, LD2, LD1, LD0)
  - b. 4 digit 7-segment display (DISP1)
3. Connectors
  - a. 4x 12-pin headers (JA, JB, JC, JD)
  - b. FX2 connector
  - c. VGA connector
  - d. PS2 connector
  - e. RS232 connector
4. Memory devices
  - a. Strata Flash 28F128J3 (28F320J3)
  - b. Cellular RAM MT45W8MW16BGX

The Manufacturing BIST configuration file provides an interface for the PC running application to:

1. Read logical values of all pins of Input devices (for behavioral testing)
2. Write/Read logical values of all pins of connectors (for connectivity and shortcut testing)
3. Write/read data to/from memory devices (for connectivity and behavioral testing)

The Manufacturing BIST configuration file also contains behavioral Demo components for:

1. Switches and LEDs.
2. Buttons and 7 segment display.
3. A mouse attached to the PS/2 connector and a VGA display attached to the VGA connector.

On power up, the Manufacturing BIST configuration file also performs Strata Flash and Cellular RAM auto tests (without needing a PC connection) and displays the test results on the 7 segment display.



Figure 1 shows the Nexys 2 BIST project block diagram. The behavior is explained below, with each block.

### Port definition

Clock signal

- clk : in; system clock (50MHz)

Epp/BlockRamCtrl bus signals

- EppAstb : in; Address strobe
- EppDstb : in; Data strobe
- UsbFlag : in; write signal (EppWr for EppCtrl and EppSyncMux, UsbFlag for BlockRamCtrl)
- UsbMode : in; EppRst reset signal for EppCtrl, UsbMode for EppSyncMux and BlockRamCtrl
- UsbDB : inout; 8 bit data bus (EppDB for EppCtrl, UsbDB for EppSyncMux and BlockRamCtrl)
- EppWait : out; wait signal (EppWait for EppCtrl, UsbRD for EppSyncMux and BlockRamCtrl)

BlockRamCtrl bus signals

- UsbClk : in;
- UsbDir : in;
- UsbOE : out;
- UsbWR : out;
- UsbPktEnd: out;
- UsbAdr : out 2 bit;

Memory bus signals

- MemDB : inout 16 bit; Memory data bus
- MemAdr : out 23 bit; Memory Address bus
- FlashByte : out; Byte enable('0') or word enable('1')
- RamCS : out; RAM CS
- FlashCS : out; Flash CS
- MemWR : out; memory write
- MemOE : out; memory read (Output Enable), also controls the MemDB direction
- RamUB : out; RAM Upper byte enable
- RamLB : out; RAM Lower byte enable
- RamCre : out; Cfg Register enable
- RamAdv : out; RAM Address Valid pin
- RamClk : out; RAM Clock
- RamWait : in; RAM Wait pin
- FlashRp : out; Flash RP pin
- FlashStSts: in; Flash ST-STs pin
- MemCtrlEnabled: out MemCtrl takes bus control

7 segment display signals

- seg : out 7 bit; cathode
- an : out 4 bit; anode
- dp : out; decimal point cathode

LED, switch, button signals

- led : out 7 bit; cathode
- sw : in 8 bit; switch input

- btn : in 4 bit; button input

#### VGA signals

- HSYNC : out; horizontal sync
- VSYNC : out vertical sync;
- vgaRed : out 3 bit Red;
- vgaGreen : out 3 bit Green;
- vgaBlue : out 2 bit Blue;

#### RS232 signals

- RsTx : inout; transmit data
- RsRx : in receive data;

#### PS/2 signals

- PS2D : inout; PS/2 data
- RS2C : inout PS/2 clock;

#### Connector signals

- PIO : inout; 68 bit IO drivers

## The EppCtrl interface

This file contains the design for an EPP interface controller. This configuration, in conjunction with a communication module, (Digilent USB, Serial, Network or Parallel module or DigilentOnBoard USB circuitry) allows the user to interface some other FPGA implemented "client" components (Digilent Library components or user generated ones) to a PC application program (a Digilent utility or user generated).

In the Nexys 2 BIST project the communication module is the DigilentOnBoard USB circuitry, and the client components are: NexysOnBoardMemCtrl, NexysOnBoardMemTest, PinDrivers, ShiftReg, and ConfReg.

A detailed description of the EppCtrl component can be found in the **Digilent OnBoard Memory Controller Reference Manual**. Compared to this document, in the Nexys 2 BIST project there are some slight changes, to allow compatibility with the BlockRamCtrl:

- the original bidirectional EppDB was split to unidirectional busses EppDBOut and EppDBIn and the three-state feature was removed. The busses are combined and three-stated in the EppSyncMux component.
- the original three-state feature of signal EppWait was removed and added back in the EppSyncMux component.

## NexysOnBoardMemCtrl

This component, in conjunction with a communication module (Nexys OnBoard USB controller), a PC application program (a Digilent utility or user generated) and the EppCtrl Digilent Library component allows the user to read or write the On Board RAM and Flash memory chips on the Digilent boards (Nexys, Nexys 2).

A detailed description of the NexysOnBoardMemCtrl component can be found in the **Digilent OnBoard Memory Controller Reference Manual**. A detailed description of the usage of NexysOnBoardMemCtrl component can be found in the **Digilent OnBoard Memory Controller User Manual**.

## NexysOnBoardMemTest

This file contains the design for a memory test controller. After (Power On) reset, this component performs automatic tests for the RAM and Flash memory chips on the Digilent Nexys 2 board and displays the result on the 7-seg display. This component, in conjunction with a communication module (Nexys OnBoard USB controller), a PC application program (a Digilent utility or user generated) and the EppCtrl Digilent Library component allows the user to read the test status.

### **Behavioral description**

PhoenixOnBoardMemTest implements a set of 7 Epp registers (read only) for PC interface (via USB):

Register	Function
0	Test Status register
1	Memory address bits 7 - 0
2	Memory address bits 15 - 8
3	Memory address bits 23 - 16
4	Expected Data bits 7 - 0
5	Expected Data bits 15 - 8
6	Read Data bits 7 - 0
7	Read Data bits 15 - 8

The registers can be read at any time, but registers 1 - 7 only provide steady values after the test ends (see test description below)

There are three sub-tests implemented:

#### *The RAM test:*

- Write RAM (status = 0x00); Writes the whole RAM on the Nexys 2 board with a sequence of 16 bit words, as follows: 0x0000, 0x0001, 0x0002, ... 0xfffe, 0x0000, ...(skipping value 0xffff, avoids address - data matching for all the memory range).
- Read RAM (status = 0x02); Reads back the whole RAM and compares to the data pattern above. If no differences, continues with the Flash QRY test. If a difference, the test ends with an RAM error message:

RAM error message (status alternates 0x03, 0x01); Test is ended and the 7-segment display alternates the messages: FAIL, 1.

In this state:

- o registers 1, 2, 3 show the ADR+2 (ADR = mismatch address)
- o registers 4, 5 show the ExpData+1 (ExpData = expected data)
- o registers 6, 7 show the actual RAM data.

#### *Flash QRY test:*

- Set Flash chip in "read QRY" mode (status = 0x06)
- Read Flash QRY (status = 0x04)
- Read Flash addresses 0x000020, 0x000022, 0x000024 and compare to the string "0x0051, 0x0052, 0x0059" (ASCII codes for "QRY"). If no differences, continues with the Flash IDC test. If a difference, the test ends with an QRY error message:

QRY error message (status alternates 0x05, 0x07); Test is ended and the 7-segment display alternates the messages: FAIL, 2

In this state:

- o registers 1, 2, 3 show the ADR+2 (ADR = mismatch address)

- registers 4, 5 show the ExpData (ExpData = expected data)
- registers 6, 7 show the actual Flash data.

#### Flash IDC test:

- Set Flash chip in "read IDC" mode (status = 0x0c)
- Read Flash IDC (status = 0x0e)
- Read Flash addresses 0x000000, 0x000001 and compare to the string "0x0089, 0x00yy, (Intel ID and Flash size codes):
  - yy = 16 - 32Mbits
  - yy = 17 - 64Mbits
  - yy = 18 - 128Mbits
  - yy = 1d - 256Mbits)

If no differences, continues with the set Flash in Array mode. If a difference, the test ends with an IDC error message:

IDC error message (status alternates 0x0f, 0x0d)

Test is ended and the 7-segment display alternates the messages: FAIL, 3

In this state:

- registers 1, 2, 3 show the ADR+2 (ADR = mismatch address)
- registers 4, 5 show the ExpData (ExpData = expected data)
- registers 6, 7 show the actual Flash data.

- Set Flash chip in "read array" mode (status = 0x0a)
  - PASS message (status alternates 0x0b, 0x09)
- Test is ended and the 7-segment display alternates the messages: PASS, zzzz (zzzz show the Flash size (in Mbits)). In this state:
- registers 1, 2, 3 show the 0x000006
  - registers 4, 5 show the ExpData (0x00ff)
  - registers 6, 7 show the actual Flash data (0x00yy).

Test is running while the status is any even value (bit 0x01 = 0):  
0x00, 0x02, 0x04, 0x06, 0x08, 0x0a, 0x0c, 0x0e

Test is done while the status is any odd value (bit 0x01 = 1):  
0x01, 0x03, 0x05, 0x07, 0x09, 0x0b, 0x0d, 0x0f

Reading from any register generates a simple Register Transfer (see the EppCtrl.vhd header); the HandShakeReqOut is inactive.

## BlockRamCtrl and ShiftReg

BlockRamCtrl component contains the design for a High Speed Communication Interface. ShiftReg component is a companion for BlockRamCtrl. It gets the transfer parameters via the EppCtrl and delivers them to the BlockRamCtrl. As a client of EppCtrl, it uses the Epp Address 0x10 ("00010000").

All the FPGA code, protocol and Adept functions which implement the High Speed Communication are currently beta versions. They are not documented nor supported by Digilent. When the final version of the High Speed Communication will be available, it will replace the beta version and it will be published and documented on the Digilent web site: [www.digilentinc.com](http://www.digilentinc.com).

## ConfReg

ConfReg gets a configuration byte via the EppCtrl. The only used bit is the least significant one, DemoEnable. This bit is sent to the VGA test components to Enable the PS/2-VGA Demo (when '0'), or release the PS/2 and VGA lines for the PinDrivers Component (when '1').

As a client of EppCtrl, it uses the Epp Address 0xB0 ("10110000").

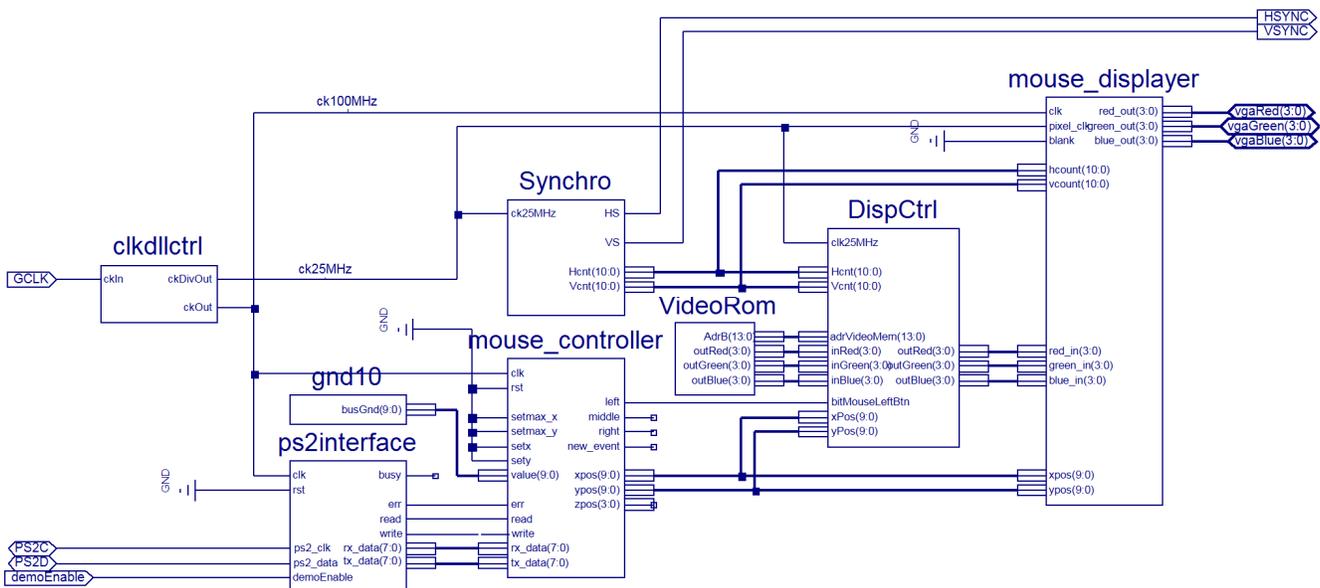
## VgaPs2Demo

VgaPs2Demo component contains the design for a VGA image generator. A moving colored pattern is shown on the VGA display (640X480, 60Hz mode). Red bars slides from lower-right corner toward upper-left one, green ones from left to right, and blue double sided bars fall from upper edge. For each color, the intensity decreases in steps from brightest to black. There are 8 intensity bars for red and green (the VGA interface handles three bit for each) and just 4 levels for blue (just 2 bits in the VGA simplified "analog to digital converter"). While the fundamental colors move, all the combinations could be observed, generating all the colors that can be generated on the 8 bit VGA interface.

A Digilent Logo is also shown on the VGA screen with black lines, white surfaces, except the ones in shadow, which degrade in 5 grey levels.

A mouse is initialized first time when connected after the project was launched. It controls a mouse cursor on the VGA screen. The mouse cursor is prevented to leave the active VGA screen. At left click, the Digilent Logo is moved at the cursor position.

Figure 2 shows the structure of the VgaPs2demo component.



**Figure 2**  
The VgaPs2Demo component block diagram

## Port definition

Clock signal

- GCLK : in; system clock (50MHz)

VGA signals

- HSYNC : out; horizontal sync
- VSYNC : out vertical sync;
- vgaRed : out 4 bit Red;
- vgaGreen : out 4 bit Green;
- vgaBlue : out 4 bit Blue;

Enable signal

- demoEnable: in; enables demo ('0') or release the PS/2 lines for other components.

PS/2 signals

- PS2D : inout; PS/2 data
- RS2C : inout PS/2 clock;

## ***ClkDIICtrl***

ClkDIICtrl component instantiates a CLKDLL Xilinx Primitive. The input signal ckIn (50MHz) is used to generate two output clock signals: ckOut (100MHz) and ckDivOut (25MHz).

## ***Synchro***

Synchro component uses the 25MHz clock signal ck25MHz to generate the HS and VS VGA synchro signals and two counters: Hcnt (pixel counter) and Vcnt (line counter). VGA mode implemented is: 640x480, 60Hz frame frequency.

## ***ps2interface, mouse\_controller and gnd10***

These files implement a mouse component, as described in the Mouse Component Reference manual. Since Value input is not used, a 10 bit GND component is used to connect it to "0000000000". The ps2interface was slightly modified in this project, adding the demoEnable input. When demoEnable is '0', ps2interface drives PS2C and PS2D, otherwise they are driven HighZ.

## ***VideoRom***

This file implements a ROM memory which encodes the Digilent Logo for VGA representation. Each line in constant Rom defines a line in the logo. There are 64 lines each with 128 pixels. A 4 bit palette is used to encode 15 grey levels and green.

The colors are encoded:

- 0 = white
- 1...13 = decreasing intensity grey levels
- 14 = black
- 15 = green background (green = 50%, blue = 25%)

Output outData decodes the colors on 12 bit:

outData(7 downto 0) = red(3 downto 0) & green(3 downto 0) & blue(3 downto 0).

## **DispCtrl**

This file generates Red, Green and Blue color bars (decreasing intensity) which move on the screen on three directions. It also overlaps the Digilent Logo at the position shown by the mouse\_controller, when bitMouseLeftBtn input is active.

### ***mouse\_displayer***

This file overlaps the mouse cursor over the image of dispCtrl. For details about mouse\_displayer component, see the Mouse Displayer Reference Manual.

## **VgaSelect**

The VgaSelect component is a mux for the VGA connector: when demoEn input is '0', the demo VGA signals are sent to the VGA connector pins. Otherwise, it sends the VGA signals generated by the PinDrivers component.

## **ClkDIIIDiv16**

ClkDIIIDiv16 component instantiates a CLKDLL Xilinx Primitive. The input signal ckIn (50MHz) is used to generate an output clock signal: ckDivOut divided by 16 (3.125MHz).

## **Demo**

Demo file contains the demo for the 7 Segment Display and the LEDs on the Nexys 2 Board. Each switch controls the state of an LED (SW0 ->LED0, etc.) When the switch is "1" (UPPER position), the corresponding LED gets ON, and reverse. A "fade" effect is added: when changing the switch position to "1", the corresponding LED increases intensity gradually from OFF to ON. This is done by supplying the LED with a Pulse Width Modulated signal which starts at 0 duty factor (LED off), increases the duty factor (increasing the LED light intensity) until reaching the duty factor 1 (LED full ON). When turning the switch to "0", the duty factor (and the LED intensity) decreases..

The Snake is the demo on the 7 Segment Display. A curled up snake is represented on one of the 7-segment display digits. The segment light intensity is maximal for the snake head and decreases towards the tail (again PWM is used to control the light intensity). The snake likes to rest on the digit where the decimal dot is lit (target digit). Pressing a button moves the target to the corresponding digit (BTN0 -> Digit 0, etc.). Immediately, the snake begins to crawl towards the new target: it uncurls, slides through the midline of the display (the G segments of the intermediate digits) and curls back on the target digit.

## **DispMux**

The DispMux component is a mux for the 7 segment display pins: after (Power On) reset, the NexysOnBoardMemTest signals are sent to the 7 segment display pins. When any switch or button changes state, the mux switches so that the Demo signals are sent to the 7 segment display pins.

## **PinDrivers**

The file implements IO drivers for the FPGA IO pins. The SW and BTN state can be read with this module. The file is used together with communication modules and PC software to implement a shortcut test for the target board. The PC software has to:

- set a test vector of 68 bits (all '1's)
- send the test vector (byte-wise) to registers 0x80...0x9b,
- read same registers and compare to the original test vector.
- loop the previous steps 64 more times, with a moving '0' as test vector.

In case of a shortcut between two output pins which drive different values, there are two conditions to meet:

- the conflict should not be destructive for the FPGA
- the conflict should be visible for the PC software.

To meet these conditions, PullUp primitives are instantiated for each pin and the output drivers convert logical levels as follows:

- '1' → HiZ with PullUp
- '0' → I (weak low)

The pins can be written or read as bytes to or from registers 0x80 - ...0x96, as follows:

Register 0x80 (or 0x81)

*User BTNs, RS232, PS/2*

Register 0x82 (or 0x83)

*User Switches*

IO PIN	DD (0x81)	Bit		IO PIN	DD (0x83)	Bit
PS2D	In/Out	0x01		SW0	Input	0x01
PS2C	In/Out	0x02		SW1	Input	0x02
RxRx	Input	0x04		SW2	Input	0x04
RxTx	In/Out	0x08		SW3	Input	0x08
BTN0	Input	0x10		SW4	Input	0x10
BTN1	Input	0x20		SW5	Input	0x20
BTN2	Input	0x40		SW6	Input	0x40
BTN3	Input	0x80		SW7	Input	0x80

Register 0x88 (or 0x89)

*FX2*

Register 0x8a (or 0x8b)

*FX2*

IO PIN	DD (0x89)	Bit		IO PIN	DD (0x8b)	Bit
IO1	In/Out	0x01		IO2	In/Out	0x01
IO3	In/Out	0x02		IO4	In/Out	0x02
IO5	In/Out	0x04		IO6	In/Out	0x04
IO7	In/Out	0x08		IO8	In/Out	0x08
IO9	In/Out	0x10		IO10	In/Out	0x10
IO11	In/Out	0x20		IO12	In/Out	0x20
IO13	In/Out	0x40		IO14	In/Out	0x40
IO15	In/Out	0x80		IO16	In/Out	0x80

Register 0x8c (or 0x8d)

*FX2*

Register 0x8e (or 0x8f)

*FX2*

IO PIN	DD (0x8d)	Bit		IO PIN	DD (0x8e)	Bit
IO17	In/Out	0x01		IO18	In/Out	0x01
IO19	In/Out	0x02		IO20	In/Out	0x02
IO21	In/Out	0x04		IO22	In/Out	0x04
IO23	In/Out	0x08		IO24	In/Out	0x08
IO25	In/Out	0x10		IO26	In/Out	0x10
IO27	In/Out	0x20		IO28	In/Out	0x20
IO29	In/Out	0x40		IO30	In/Out	0x40
IO31	In/Out	0x80		IO32	In/Out	0x80

Register 0x90 (or 0x91)

FX2

IO PIN	DD (0x91)	Bit		IO PIN	DD (0x93)	Bit
IO33	In/Out	0x01		R-JA1	In/Out	0x01
IO35	In/Out	0x02		R-JA2	In/Out	0x02
IO37	In/Out	0x04		R-JA3	In/Out	0x04
IO39	In/Out	0x08		R-JA4	In/Out	0x08
IO34	In/Out	0x10		R-JA7	In/Out	0x10
IO36	In/Out	0x20		R-JA8	In/Out	0x20
IO38	In/Out	0x40		R-JA9	In/Out	0x40
IO40	In/Out	0x80		R-JA10	In/Out	0x80

Register 0x92 (or 0x93)

12-pin header JA

Register 0x94 (or 0x95)

12-pin header JB

IO PIN	DD (0x95)	Bit		IO PIN	DD (0x97)	Bit
R-JB1	In/Out	0x01		R-JC1	In/Out	0x01
R-JB2	In/Out	0x02		R-JC2	In/Out	0x02
R-JB3	In/Out	0x04		R-JC3	In/Out	0x04
R-JB4	In/Out	0x08		R-JC4	In/Out	0x08
R-JB7	In/Out	0x10		R-JC7	In/Out	0x10
R-JB8	In/Out	0x20		R-JC8	In/Out	0x20
R-JB9	In/Out	0x40		R-JC9	In/Out	0x40
R-JB10	In/Out	0x80		R-JC10	In/Out	0x80

Register 0x96 (or 0x97)

12-pin header JC

Register 0x98 (or 0x99)

12-pin header JD

IO PIN	DD (0x95)	Bit		IO PIN	DD (0x??)	Bit
R-JD1	In/Out	0x01		BLU1	Output	0x01
R-JD2	In/Out	0x02		BLU2	Output	0x02
R-JD3	In/Out	0x04		GRN0	Output	0x04
R-JD4	In/Out	0x08		GRN1	Output	0x08
HSYNC	Output	0x10		GRN2	Output	0x10
VSYNC	Output	0x20		RED0	Output	0x20
		0x40		RED1	Output	0x40
		0x80		RED2	Output	0x80

Register 0x9A (or 0x9b)

VGA connector