

Overview

This document describes a project used to compile the Basys Rev. E Built In Self Test (BIST) configuration file. This configuration is loaded on the Basys Rev. E platform flash during manufacturing and serves for manufacturing testing. After testing, the Basys Demo configuration file is loaded in the platform Flash on the Basys Rev. E boards (until erased/replaced by user) and serves as the Demo. For more details about the Basys Demo Project, see the Digilent *Basys Demo Reference Project*.

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 this information, users 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 Basys board:

Input Devices

- 8x switch (SW7, SW6, SW5, SW4, SW3, SW2, SW1, SW0)
- 4x button (BTN3, BTN2, BTN1, BTN0)

Output Devices

- 8x User LEDs (LD7, LD6, LD5, LD4, LD3, LD2, LD1, LD0)
- four digit seven-segment display (DISP1)

Connectors

- four 6-pin headers (JA, JB, JC, JD)
- VGA connector
- PS2 connector

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

- read logical values of all pins of input devices (for behavioral testing)
- write/read logical values of all pins of connectors (for connectivity and shortcut testing).

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

- switches and LEDs
- buttons and seven-segment display
- a VGA display attached to the VGA connector.

Functional Description

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

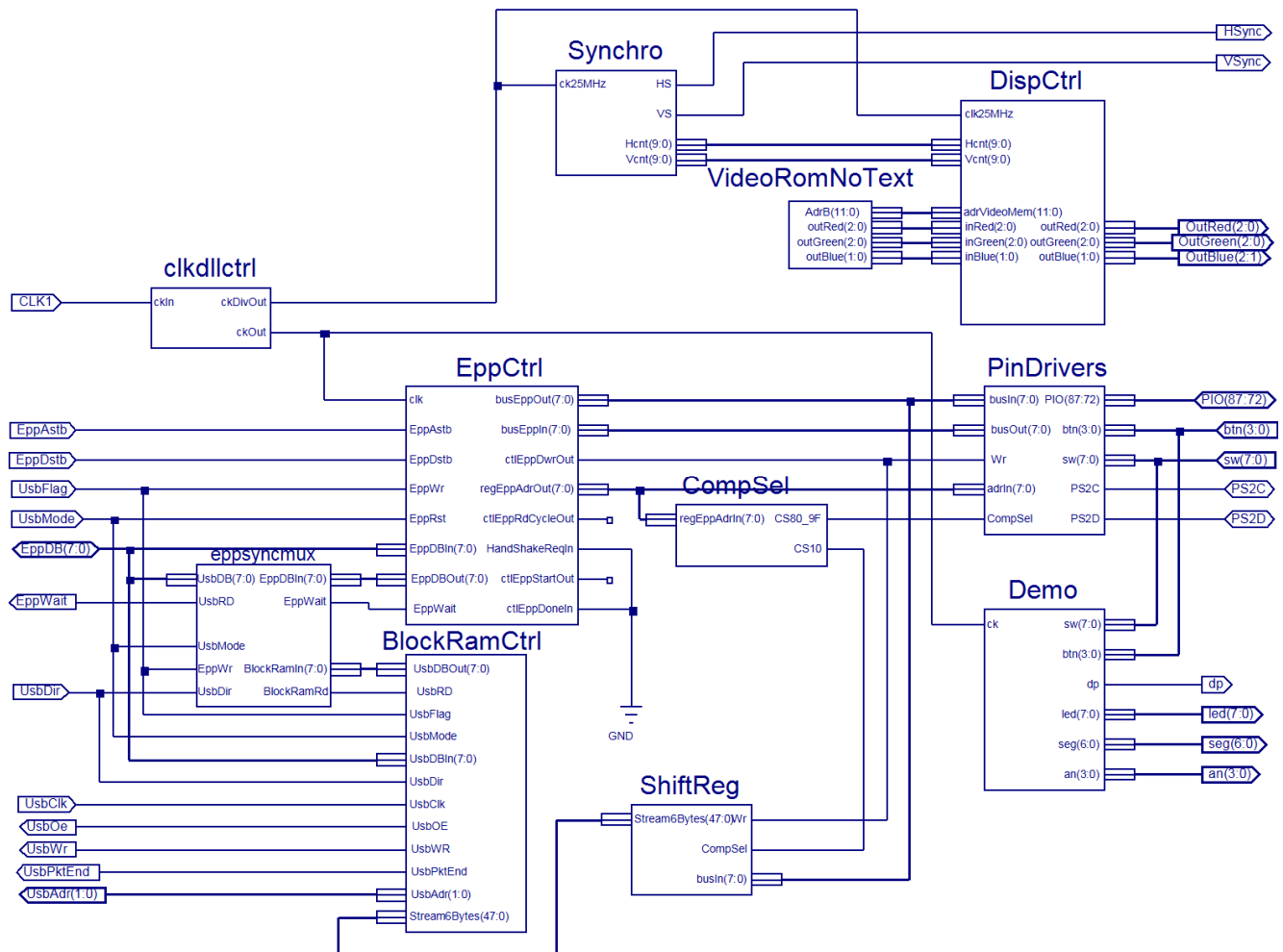


Figure 1 The Basys BIST Project

Port Definitions

Clock Signals

clk1 in, system clock (100MHz from IC6)

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

EppDB in/out, 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

Seven-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
OutRed	out 3-bit, red
OutGreen	out 3-bit, green
OutBlue	out 2-bit, blue

PS/2 Signals

PS2D	inout, PS/2 data
RS2C	inout, PS/2 clock

Connector Signals

PIO	inout, 16-bit I/O 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 Digilent OnBoard 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 (either a Digilent- or user-generated utility).

In the Basys BIST project, the communication module is the Digilent OnBoard USB circuitry, and the client components are PinDrivers and ShiftReg.

A detailed description of the EppCtrl component can be found in the Digilent *OnBoard MemCfg Reference Project* document. The Basys BIST project incorporates the following changes to allow compatibility with the BlockRamCtrl.

- The original bi-directional 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.

BlockRamCtrl and ShiftReg

The BlockRamCtrl component contains the design for a high-speed communication interface. The 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 becomes available, it will replace the beta version and will be published and documented on the Digilent web site: www.digilentinc.com.

ClkDllCtrl

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

VGA PS/2 Demo

The Basys BIST project contains the design for a VGA image generator. A moving colored pattern is shown on the VGA display (640X480, 60Hz mode). Red bars slide from the lower-right corner toward the upper-left one, green ones from left to right, and blue double-sided bars fall from the upper edge. For each color, the intensity decreases in steps from brightest to black. There are eight intensity bars for red and green (the VGA interface handles three bits for each) and just four levels for blue (just two 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 and white surfaces, except the ones in shadow, which degrade in five grey levels.

Synchro, VideoRom, and DispCtrl components are involved in the VGA PS/2 Demo.

The image generated by the Basys BIST project is poor quality. The reason for that is the unstable frequency generated by the RC oscillator IC6. This clock signal (CLK1) is good for applications which do not require a very precise and stable frequency, but is not appropriate for more advanced projects (video, asynchronous serial transmissions, etc.). See more about this issue in the *Basys Demo Reference Project*.

Synchro

The 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). The VGA mode implemented is 640x480, 60Hz frame frequency.

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 64 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 8-bit: outData(7 downto 0) = red(2 downto 0) & green(2 downto 0) & blue(1 downto 0).

DispCtrl

This file generates red, green, and blue color bars (decreasing intensity) which move on the screen in three directions. It also overlaps the Digilent logo. The green color on the Digilent logo is replaced by "transparency."

Demo

The Demo file contains the demo for the seven-segment display and the LEDs on the Basys board. Each switch controls the state of an LED (SW0 -> LED0, etc.). When the switch is '1' (UPPER position), the corresponding LED is ON, and vice versa. 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 (PWM) signal which starts at '0' duty factor (LED OFF), then 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 seven-segment display. A curled up snake is represented on one of the seven-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 (the 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.

PinDrivers

This file implements I/O drivers for the FPGA I/O 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 18 bits (all 1's)
- send the test vector (byte-wise) to registers 0x80, 0x82, 0x86, 0x88
- read the same registers and compare to the original test vector
- loop the previous steps 18 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 BTN_s, RS232, PS/2

Register 0x82 (or 0x83)
User Switches

I/O PIN	DD (0x81)	Bit		I/O PIN	DD (0x83)	Bit
PS2D	In/Out	0x01		SW0	Input	0x01
PS2C	In/Out	0x02		SW1	Input	0x02
		0x04		SW2	Input	0x04
		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 0x86 (or 0x87)
12-pin header JB

Register 0x88 (or 0x89)
12-pin header JC

IO PIN	DD (0x95)	Bit		IO PIN	DD (0x97)	Bit
R-JA1	In/Out	0x01		R-JC1	In/Out	0x01
R-JA2	In/Out	0x02		R-JC2	In/Out	0x02
R-JA3	In/Out	0x04		R-JC3	In/Out	0x04
R-JA4	In/Out	0x08		R-JC4	In/Out	0x08
R-JB1	In/Out	0x10		R-JD1	In/Out	0x10
R-JB2	In/Out	0x20		R-JD2	In/Out	0x20
R-JB3	In/Out	0x40		R-JD3	In/Out	0x40
R-JB4	In/Out	0x80		R-JD4	In/Out	0x80