Using ispGDX® to Replace Boundary Scan Bus Devices

Introduction

Boundary Scan Bus Devices are available from Texas Instruments (TI) and National Semiconductor (NS)/ Fairchild Semiconductor for enhancing board-level test of bus interfaces. These TI SCOPE™ and NS SCAN devices are essentially updated versions of standard bus devices that also offer boundary scan test capabilities. Lattice's high I/O ispGDX Family can integrate many 8- to 20-bit standard interface devices, including these JTAG bus devices, as shown in Figure 1. The 5V ispGDX and 3.3V ispGDXV devices have 24mA I_{OL} drive which allows them to replace many standard bus transceivers, registers, switches, multiplexers and de-multiplexers. Built-in JTAG test circuitry and In-System Programmability (ISP™) allow ispGDX devices to go above and beyond the board test flexibility promoted by the TI SCOPE and National/Fairchild SCAN products. Table 1 shows a list of such devices easily integrated into ispGDX devices.

In addition, applications calling for single or banks of multiple inverting and non-inverting buffers can be combined into one device. The ispGDX architecture has outputs that can be configured as combinatorial, D-type flip flop or D-type latch. Using the ispGDX device's in-system programmable on-chip interconnect, shift registers with different combinations of parallel and serial in and out are possible in user selectable lengths. Schmitt-trigger inputs combined with pin selectable slew rate control outputs help ispGDX devices to tolerate noisy input signals. High drive and individual tristate control for the I/O cells make almost any TTL-bus application possible, especially with the 3.5ns Tpd and Tco of the 3.3V ispGDXV devices. After a brief overview of the ispGDX architecture, several examples illustrating the use of the ispGDX devices for boundary scan bus devices are presented. For more detailed information on the ispGDX or ispGDXV devices, refer to the ispGDX or ispGDXV Family data sheets.

ispGDX Architecture Overview

ispGDX devices are in-system programmable generic digital crosspoint devices, including four device densities ranging from 80 to 160 programmable I/O pins for the ispGDX devices and 80 to 240 programmable I/O pins for the ispGDXV devices. The I/O pins are divided into four banks. Figure 2 shows the individual I/O cell structure with routing for the 160 I/O device. Each of the four MUX inputs into the 4-to-1 I/O MUX can be connected to an I/O pin in its respective bank. The signal can be registered, latched or passed directly through to an output pin in its true or inverted form. More advanced functions can make use of the register and pin feedback paths. Control signals for MUX selects, clocks and output enables are routed from specific I/O pins based on the user's design. Half the I/O pins are available as MUX selects while the remaining I/O pins are divided equally between clocks and output enables. Global clocks are also available. Table 2 provides a summary of the features available in the various ispGDX devices. The ispGDX is used to describe the general ispGDX architecture throughout

Figure 1. Typical ispGDX Application in System Environment

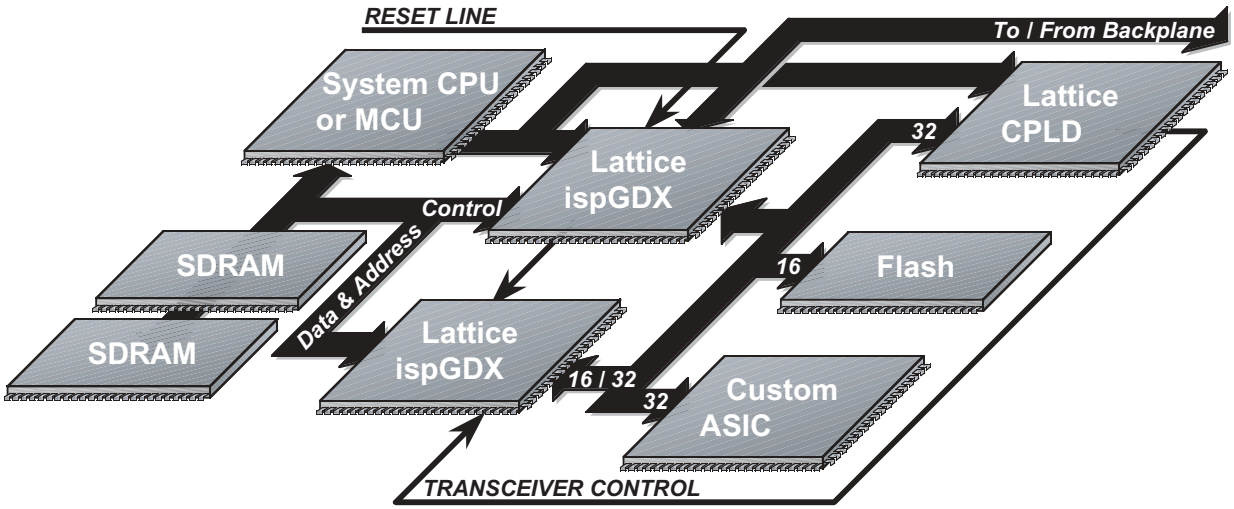

Table 1. Texas Instruments and National/Fairchild Scan Bus Devices Replaceable with ispGDX

TI 5V Scan Devices

Device	Description
SN54ABT8245	8-BIT SCAN TEST, BUS TRANSCEIVER
SN54ABT8543	8-BIT SCAN TEST, BUS TRANSCEIVER
SN54ABTH18502A	18-BIT SCAN TEST, BUS TRANSCEIVER
SN54BCT8240A	8-BIT SCAN TEST, BUS BUFFER
SN54BCT8244A	8-BIT SCAN TEST, BUFFER
SN54BCT8245A	8-BIT SCAN TEST, BUS TRANSCEIVER
SN54BCT8373A	8-BIT SCAN TEST, LATCH
SN54LVT18502	18-BIT SCAN TEST, BUS TRANSCEIVER
SN74ABT18245A	18-BIT SCAN TEST, BUS TRANSCEIVER
SN74ABT18502	18-BIT SCAN TEST, BUS TRANSCEIVER
SN74ABT18504	20-BIT SCAN TEST, BUS TRANSCEIVER
SN74ABT18640	18-BIT SCAN TEST, TRANSCEIVER
SN74ABT8245	8-BIT SCAN TEST, TRANSCEIVER
SN74ABT8543	8-BIT SCAN TEST, REGISTERED TRANSCEIVER
SN74ABT8952	8-BIT SCAN TEST, REGISTERED TRANSCEIVER
SN74ABTH182502A	18-BIT SCAN TEST, TRANSCEIVER
SN74ABTH182504A	20-BIT SCAN TEST, TRANSCEIVER
SN74ABTH18502A	18-BIT SCAN TEST, TRANSCEIVER
SN74ABTH18504A	20-BIT SCAN TEST, TRANSCEIVER
SN74BCT8240A	8-BIT SCAN TEST, BUFFER
SN74BCT8244A	8-BIT SCAN TEST, BUFFER
SN74BCT8245A	8-BIT SCAN TEST, BUS TRANSCEIVER
SN74BCT8373A	8-BIT SCAN TEST, LATCH
SN74BCT8374A	8-BIT SCAN TEST, D-TYPE FLIP-FLOP
SN74LVT18512	18-BIT SCAN TEST, BUS TRANSCEIVER

TI 3.3V Scan Devices

Device	Description
SN74LVTH182502A	18-BIT UNIVERSAL SCAN TEST, BUS TRANSCEIVER
SN74LVTH182504A	20-BIT UNIVERSAL SCAN TEST, TRANSCEIVER
SN74LVTH182512	18-BIT UNIVERSAL SCAN TEST, TRANSCEIVER
SN74LVTH182514	20-BIT UNIVERSAL SCAN TEST, BUS TRANSCEIVER
SN74LVTH18502A	18-BIT SCAN TEST, BUS TRANSCEIVER
SN74LVTH18504A	20-BIT SCAN TEST, BUS TRANSCEIVER
SN74LVTH18512	18-BIT SCAN TEST, UNIVERSAL BUS TRANSCEIVER
SN74LVTH18514	20-BIT SCAN TEST, UNIVERSAL BUS TRANSCEIVER

National Semiconductor/Fairchild Semiconductor 5V Scan Devices

Device	Description
SCAN182245A	18-BIT SCAN TEST, TRANSCEIVER w/SERIES TERMINATION
SCAN182373A	18-BIT ACTIVE LOW LATCH ENABLE LATCH/BUFFER w/SERIES TERMINATION
SCAN182374A	18-BIT ACTIVE HIGH CLOCK REGISTER/BUFFER w/SERIES TERMINATION
SCAN18245T	18-BIT SCAN TEST, TRANSCEIVER
SCAN182541A	18-BIT SCAN TEST, TRANSCEIVER w/DUAL OUTPUT ENABLE
SCAN18373T	18-BIT ACTIVE LOW LATCH ENABLE LATCH/BUFFER
SCAN18374T	18-BIT ACTIVE HIGH CLOCK REGISTER/BUFFER
SCAN18540T	18-BIT INVERTING BUFFER WITH DUAL OUTPUT ENABLE
SCAN18541T	18-BIT NON-INVERTING BUFFER WITH DUAL OUTPUT ENABLE

Figure 2. ispGDXV I/O Cell and GRP Detail (160-I/O Device)

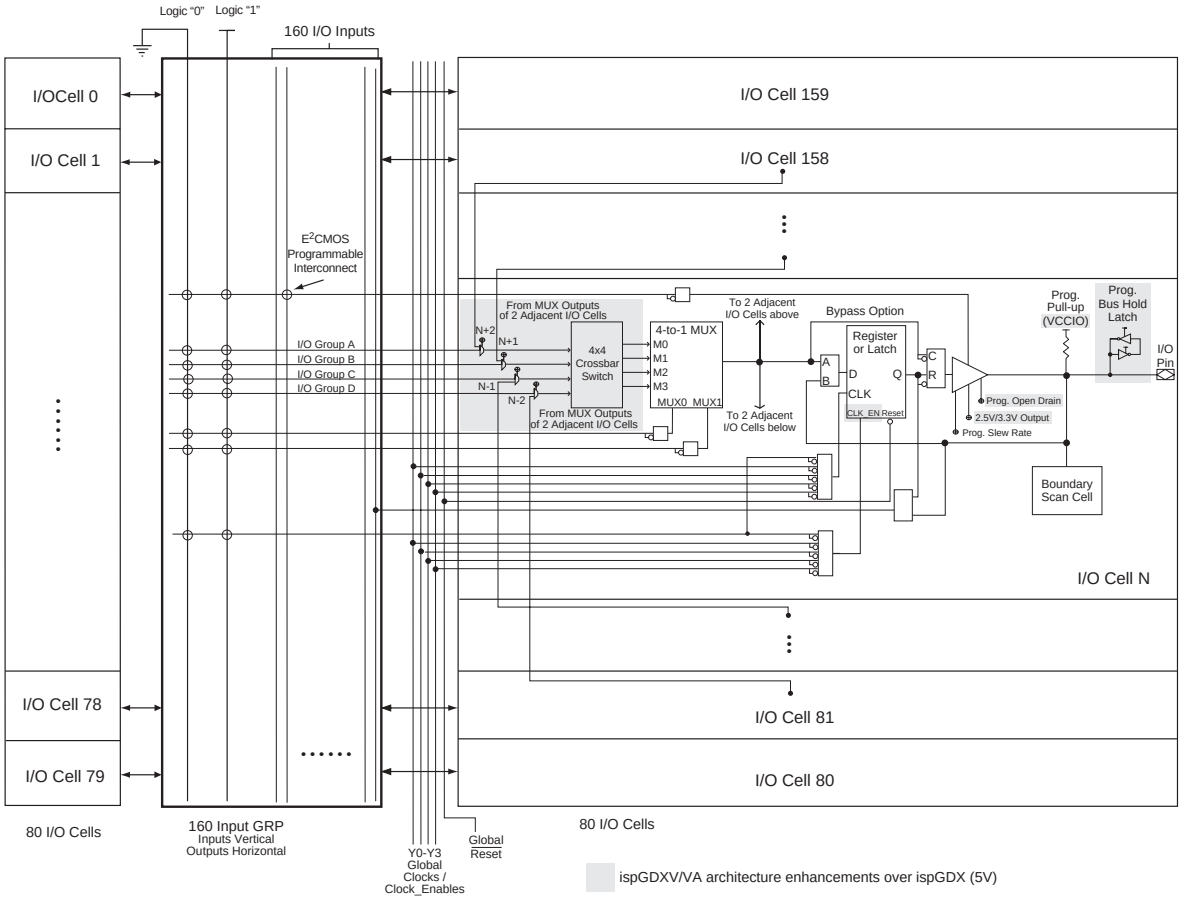

Table 2. ispGDX Family Feature Summary

	ispGDX DEVICE FAMILY			
	ispGDX80A/VA	ispGDX120A	ispGDX160/A/VA	ispGDX240VA
I/O Pins	80	120	160	240
I/O-OE Inputs*	20	30	40	60
I/O-Clk Inputs*	20	30	40	60
I/O-MUXsel1 Inputs*	20	30	40	60
I/O-MUXsel2 Inputs*	20	30	40	60
Dedicated Clock Pins	2	4	4	4

*CLK, OE, MUX0 and MUX1 can each access 25% of the I/Os.

this document. ispGDXV is used to specify the 3.3V devices and ispGDX to specify the 5V devices.

Bus Device Examples

Texas Instruments' boundary scan bus device family is the old architecture with the addition of boundary scan test circuitry. The flexibility of the Lattice ispGDX device allows many of these architectures to be emulated. In addition, in-system programmability allows these architectures to be customized for individual board test and/or functional operation. Listed below are four examples based on existing Texas Instruments devices.

Example 1. 8-Bit Bus Transceiver (SN74ABT8245)

This TI device is functionally equivalent to the 'F245 and 'ABT245 octal bus transceivers. The DIR control chooses the direction of the data flow. The OE control activates the output chosen by DIR (refer to Function Table 1).

Function Table 1

INPUTS		OPERATION
OE	DIR	
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

To emulate this architecture with an ispGDX device, some logic is required to create the individual bus OE controls from the different combinations of the OE and DIR controls. The logic is simple and can be implemented inside the ispGDX device using MUX logic. Listing 1 provides the ispGDX design file. The ispGDX device also supports boundary scan test, allowing a snapshot of the bus to be taken using the built-in test hardware.

With the ispGDX, the function can be expanded to handle bus widths of up to 76 bits with the ispGDX160 device. In contrast, multiple 8-bit bus transceiver chips can be combined into one ispGDX device. For example, the equivalent of eight discrete SN74ABT8245 devices fit inside of one ispGDX160VA device. If any of the control signals are common, more may fit. Distributor 1000+ piece pricing is $7.50 for the SN74ABT8425DW, so the ispGDX devices represent a very cost-effective alternative at a cost savings of 60% to replace eight discrete devices (compared to 1000+ quantity pricing). In addition, there is a significant board space savings with the advanced fine pitch packaging of the ispGDX device.

Example 2. 8-Bit Bus Transceiver with Registers (SN74ABT8952)

This TI device is functionally equivalent to the 'BCT2952 and 'ABT2952 octal registered bus transceivers. There are individual bus output enables, clocks and clock enables (Function Table 2). The ispGDX Family can emulate this architecture without requiring any extra logic. Listing 2 provides the ispGDX design file.

For this architecture, the bus width can be expanded up to 77 bits with the ispGDX160. Or, the equivalent of seven discrete SN74ABT8952 devices can fit inside one ispGDX160VA. If any of the control signals are common, more may fit. The distributor 1000+ piece pricing is $7.50 for the SN74ABT8952DW.

Function Table 2

INPUTS				OUTPUTS
OEAB	CLKENAB	CLKAB	A	B
L	L	↑	L	L
L	L	↑	H	H
L	H	X	X	B_0
L	X	L	X	B_0
H	X	X	X	Z

INPUTS				OUTPUTS
OEAB	CLKENBA	CLKBA	B	A
L	L	↑	L	L
L	L	↑	H	H
L	H	X	X	A_0
L	X	L	X	A_0
H	X	X	X	Z

Example 3. 8-Bit Buffer/Driver (SN74BCT8244A)

This TI device is functionally equivalent to the 'F244 and 'BCT244 octal buffers. There are two OE controls for the lower and upper half of the buffered bus output (Function Table 3). This architecture is one of the easiest to implement in the ispGDX. No extra logic is required and the OE signals are the only control required. Listing 3 provides the ispGDX design file.

Function Table 3

INPUTS		OUTPUT
OE	A	Y
H	X	Z
L	L	L
L	H	H

For this architecture, the bus width can be expanded up to 79 bits with the ispGDX160. Or, the equivalent of eight discrete SN74BCT8244A devices can fit inside one ispGDX160. If any of the control signals are common, more may fit. The distributor 1000+ piece pricing is $5.00 for the SN74BCT8244ADW.

Example 4. 8-Bit Bus Interface with FF (SN74BCT8374A)

This TI device is functionally equivalent to the 'F374 and 'BCT374 octal D-type flip-flops. There is an OE control for the output bus. Bus data is clocked into the D-type registers on the rising edge of CLK (refer to Function Table 4). Listing 4 includes the ispGDX design file.

Function Table 4

INPUTS			OUTPUTS
OE	CLK	D	Q
L	↑	H	H
L	↑	L	L
L	H or L	X	Q_0
H	X	X	Z

For this architecture, the bus width can be expanded up to 79 bits when using the ispGDX160. Or, the equivalent of eight discrete SN74BCT8374A devices fit inside of one ispGDX160VA. If any of the control signals are common, more may fit. The distributor 1000+ piece pricing is $5.00 for the SN74BCT8374ADW.

Board Test Implications

The examples above illustrate data bus manipulation. Tapping into a bus is a good way to find out whether a board is functioning correctly. Using the IEEE 1149.1 Sample/Preload instruction allows the user to take a snapshot of the data bus without disturbing the board function. In addition, by utilizing the Extest instruction, known values can be put on the bus to see if the desired result is found.

An extension of the basic test operation is to use a data path boundary scan device to isolate certain portions of the board. It is common practice to put simple JTAG buffer devices on a board, solely to aid board test. These devices can simply break a bus or drive enable signals to other devices or busses. They are especially useful in adding board testability to devices without built-in boundary scan test.

The ispGDX Family offers the best of both worlds: the flexibility to emulate standard bus device operation, to add user configuration and to aid in board test. In addition to boundary scan test, the use of in-system programmability does not dedicate the ispGDX device to functional or test operation, but allows both.

Summary

While the above examples illustrate how to implement single components, it is possible to create any combination of the above. By listing the required data and control signals for each component, the best mapping to the ispGDX device can be found. Resources can be shared for common signals. The limiting factor can be either data or control pins. Table 2 provides a summary of the resources required for each of the 8-bit examples above. If a wider bus is used, the data signals used increase while the control signals stay the same. Common signals between different components will not use any additional resources. Some of the architectures use extra I/O or control pins to implement required logic.

Lattice's ispLEVER™ Development System software supports ispGDX and ispGDXV design. This design tool is available for download from the Lattice web site at www.latticesemi.com.

Technical Support Assistance

Hotline: 1-800-LATTICE (Domestic)
1-408-826-6002 (International)
e-mail: techsupport@latticesemi.com

Table 3. Resource Usage Based on ispGDXVA Architecture

	I/O PINS		CONTROL PINS		
Component	Data Signals	Logic Creation	Clocks	MUX Selects	Output Enables
8245	16	2	0	0	2
8952	16	0	2	2	2
8244	16	0	0	0	2
8374	16	0	1	0	1

Listing 1. 8-Bit Bus Transceiver

```
// 8-bit Bus transceiver;
// Functions like TI SN74ABT8245;
//
//
DESIGN ti8245;

PART ispGDX160VA-3Q208;

// Set declarations
SET busA [A0..A7];  // Bus can be 1 to 39 signals
SET busB [B0..B7];  // Bus can be 1 to 39 signals

// Pin assignments
BIDI busA {A0..A7};    // Bus signals
BIDI busB {B0..B7};

INPUT OE_AB {D3};      // Output enable vs isolation
INPUT DIR_AB {D2};     // Direction control

BIDI OE_A {D33};       // Mux logic to create busA OE, device pin needs to be floating
BIDI OE_B {D37};       // Mux logic to create busB OE, device pin needs to be floating

BEGIN
        busA.m1 = busB;
        busA.s0 = VCC;
        busA.s1 = GND;
        busA.oe = OE_A;

        busB.m0 = busA;
        busB.s0 = GND;
        busB.s1 = GND;
        busB.oe = OE_B;

        OE_A.m0 = VCC;
        OE_A.m1 = GND;
        OE_A.m2 = GND;
        OE_A.m3 = GND;
        OE_A.s0 = DIR_AB;
        OE_A.s1 = OE_AB;
        OE_A.oe = VCC;

        OE_B.m0 = GND;
        OE_B.m1 = VCC;
        OE_B.m2 = GND;
        OE_B.m3 = GND;
        OE_B.s0 = DIR_AB;
        OE_B.s1 = OE_AB;
        OE_B.oe = VCC;

END
```

Listing 2. 8-Bit Bus Transceiver with Registers

```
// 8-bit Bus transceiver with registers;
// Functions like TI SN74ABT8952;
//
//
// NOTE: Feedback is taken from the pin, not internal, to implement clock enable.
// When the clock is not enabled, oe not enabled and CLK,
// then register loaded from the bus.
//

DESIGN ti8952;

PART ispGDX160VA-3Q208;

// Set declarations
SET busA [A0..A7];  // Bus can be 1 to 40 signals
SET busB [B0..B7];  // Bus can be 1 to 40 signals

// Pin assignments
BIDI busA {A0..A7};   // Bus signals
BIDI busB {B0..B7};

INPUT OE_AB {D37};    // Output enable, busA (OE input)
INPUT OE_BA {D33};    // Output enable, busB (OE input)
INPUT CLKENAB {D38}; // Clock enable, busA (mux input)
INPUT CLKENBA {D39}; // Clock enable, busB (mux input)
INPUT CLKAB {D36}; // Clock, busA (CLK input)
INPUT CLKBA {D32}; // Clock, busB (CLK input)

BEGIN
        busA.m0 = busA;               // Hold when clock not enabled
        busA.m1 = busB;
        busA.s0 = !CLKENAB;
        busA.oe = !OE_AB;
        busA.clk = CLKAB;


        busB.m0 = busA;
        busB.m1 = busB;               // Hold when clock not enabled
        busB.s0 = CLKENBA;
        busB.oe = !OE_BA;
        busB.clk = CLKBA;


END
```

Listing 3. 8-Bit Buffer/Driver

```
// 8-bit Bus driver/buffer
// Functions like TI SN74BCT8244A
//

DESIGN ti8244;

PART ispGDX160VA-3Q208;

// Set Declarations
set bus1A [s_1A1..s_1A4];
set bus2A [s_2A1..s_2A4];
set bus1Y [s_1Y1..s_1Y4];
set bus2Y [s_2Y1..s_2Y4];

// Pin Declarations
INPUT bus1A {A0..A3};         //Low nibble in
INPUT bus2A {A4..A7};         //High nibble in
OUTPUT bus1Y {A8..A11};       //Low nibble out
OUTPUT bus2Y {A12..A15};      //Low nibble out

INPUT s_1OE {A37};            //Low nibble OE
INPUT s_2OE {A33};            //High nibble OE

BEGIN
        bus1Y.m0 = bus1A;
        bus1Y.s0 = GND;
        bus1Y.s1 = GND;
        bus1Y.oe = !s_1OE;

        bus2Y.m0 = bus2A;
        bus2Y.s0 = GND;
        bus2Y.s1 = GND;
        bus2Y.oe = !s_2OE;

END
```

Listing 4. 8-Bit Buffer/Driver with D-Type Registers

```
// 8-bit Bus driver/buffer with registers
// Functions like TI SN74BCT8374A
//

DESIGN ti8374;

PART ispGDX160VA-3Q208;

// Set Declarations
set busD [D1..D8];
set busQ [Q1..Q8];

// Pin Declarations
INPUT busD {A0..A7};
OUTPUT busQ {A8..A15};

INPUT busOE {A37};
INPUT busCLK {Y0};     //Use global clock or individual clock

BEGIN
        busQ.m0 = busD;
        busQ.s0 = GND;
        busQ.s1 = GND;
        busQ.oe = !busOE;
        busQ.clk = busCLK;

END
```