Using Proprietary Lattice ISP Devices

August 2001

Introduction

This document describes how to program Lattice's In-System Programmable (ISP™) devices that utilize the proprietary Lattice ISP State Machine for programming, rather than the IEEE 1149.1 Test Access Port (TAP) controller. Most of these details are transparent to the user if Lattice in-system programming software and hardware are used.

There are two basic groups of devices that include the proprietary Lattice ISP state machine for programming. The first group includes those devices that can only be programmed using this state machine while the second group can use either this state machine or the TAP controller. The ispLSI® 1000/E, 2000/A, ispGAL®22V10 and ispGDS® devices fall into the first group while the ispLSI 3000, 8000 and ispGDX® devices are in the second group. All other Lattice ISP devices utilize only the TAP controller for programming and boundary scan test.

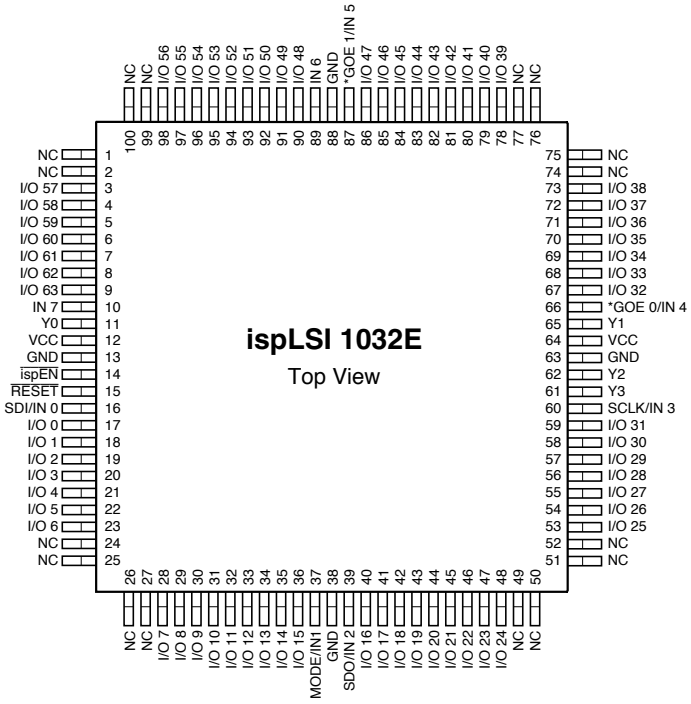
The following information is a supplement to the standard in-system programming design and usage guidelines available on the Lattice web site (www.latticesemi.com) or CD-ROM. The proprietary ISP state machine is described in detail. Board layout recommendations are discussed for combining a standard JTAG chain in parallel with a proprietary Lattice ISP device chain to use a single programming connector.

Lattice ISP State Machine

Lattice ISP State Machine Programming Pins

The programming pins used to program Lattice devices that use the proprietary Lattice ISP state machine are each described in detail in this section. Figure 1 shows the ispLSI 1032E 100-pin TQFP device pinout. The pins of interest are SDI, MODE, SCLK and SDO. Depending on the device, the $\overline{\text{ispEN}}$ or BSCAN/$\overline{\text{ispEN}}$ pin may be used to enter or select the programming mode. The various choices and methods for programming are discussed for each particular architecture group later in this document.

Figure 1. ispLSI 1032E 100-Pin TQFP Pinout Diagram



* Pins have dual function capability which is software selectable.

The Serial Data In (SDI) pin performs two different functions. First, it acts as the data input to the serial shift register built inside each Lattice ISP device. Second, it functions as one of two control pins for the programming state machine. Because of this dual role, the function of SDI is controlled by the MODE pin. When MODE is low, SDI becomes the serial input to the shift register. When MODE is high, SDI becomes a control signal for the programming state machine. Internally, the SDI signal is multiplexed to various shift registers in the device. The different shift instructions of the state machine determine which of these shift registers receives input from SDI.

The MODE signal, combined with the SDI signal, controls the programming state machine, as described in the following section about the Lattice ISP state machine operation.

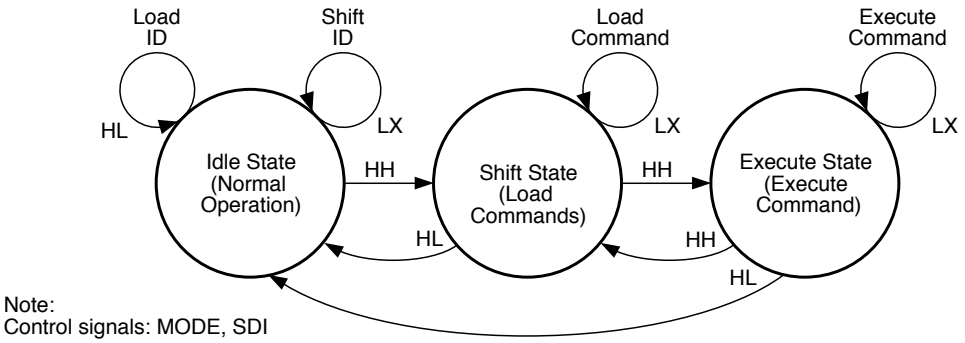
The Serial Clock (SCLK) pin provides the serial shift register with a clock. SCLK is used to clock the internal serial shift registers and clock the Lattice ISP state machine between states. State changes and shifting data in are performed on low-to-high transitions. When MODE is high, SCLK controls the programming state machine. When MODE is low, SCLK acts as a shift register clock to shift data in or out or to start an operation. When shifting data out, the data is available and valid on SDO only after a subsequent high-to-low transition of SCLK.

The Serial Data Out (SDO) pin is connected to the output of the internal serial shift registers. As previously stated, the selection of which shift register to output is determined by the state machine's shift instruction. When MODE is high, SDO connects directly to SDI, bypassing the device's shift registers.

Lattice ISP State Machine Operation

The Lattice ISP programming state machine controls which mode the device is in and provides the means to read and write data to the device (Figure 2). The three states defined in the Lattice ISP state machine diagram are the Idle state, Shift state and Execute state. Instruction codes, which are shifted into the device in the Shift state, control which instruction is to be executed in the Execute state. To transition between states, MODE is held high, SDI is set to the appropriate level, and SCLK is clocked. There is a setup time required for values placed on SDI and MODE before a valid SCLK.

Figure 2. Lattice ISP Programming State Machine



Idle State

The Idle state is the first state activated when the device is powered-up or the device enters programming mode. The state machine is in the Idle state when the user needs to read the device identification (Each Lattice ISP device type is assigned a unique identification code. See the corresponding device details section). The eight-bit device identification is loaded into the shift register by driving MODE high, SDI low, and clocking the Lattice ISP state machine with SCLK. Once the device ID is loaded, it is read out serially by driving MODE low. Notice that when the device ID is read serially, SDI can either be high or low (a logical "don't care"). When MODE is driven low after loading the ID, SDO is directly connected to the ID shift register, so the state machine needs only seven clocks to read out eight bits of device ID. As with most shift registers, the Least Significant Bit (LSB) of the device ID gets shifted out from SDO first. State transition to the Shift state occurs when both MODE and SDI are high while the Lattice ISP state machine gets a clock transition.

Shift State

This state is strictly used for shifting instructions into the state machine. The different Lattice ISP state machine instruction sets are listed for each ISP device in the corresponding device detail section. When MODE is low and SDI is “don’t care” in the Shift state, SCLK shifts the instruction into the state machine. Once the instruction is shifted into the state machine, the state machine must transition to the Execute state to perform the instruction. Driving both MODE and SDI high and applying the clock transfers the state machine from the Shift state to the Execute state. If needed, the state machine can move from the Shift state to the Idle state by driving MODE high and SDI low.

Execute State

In the Execute state, the state machine executes instructions that are loaded into the device in the Shift state. For some instructions, the state machine requires more than one clock to execute the command. An example of this multiple clock requirement is the address or data shift instruction. The number of clock pulses required for these instructions depends on the device shift register sizes. When executing instructions such as Program, Verify, or Bulk Erase, the necessary timing requirements must be followed to make sure that the commands are executed properly. For specific timing information, refer to the corresponding device detail section.

To execute a command, MODE is driven low and SDI is “don’t care.” For multiple clock instructions, MODE must remain low throughout the duration of the execution. MODE high and SDI high will take the state machine back to the Shift state and MODE high and SDI low will take the state machine to the Idle state.

$\overline{\text{ispEN}}$ and BSCAN/$\overline{\text{ispEN}}$ Function

The ispLSI 1000/E and 2000/A families use a fifth programming pin, $\overline{\text{ispEN}}$, to multiplex the functions of the SDI, SDO, SCLK and MODE pins between ISP functions during programming and user-defined logic functions during normal PLD operations.

The ispLSI 3000, 8000 and ispGDX families use a fifth pin, BSCAN/$\overline{\text{ispEN}}$, to select either the Lattice ISP interface or the ispJTAG™ interface.

Lattice ISP Daisy Chain Details

This section provides a detailed look at the issues associated with using proprietary Lattice ISP device programming. The ispLSI 1000/E, 2000/A, ispGAL22V10 and ispGDS devices can only be programmed using this method, but the ispLSI 3000, 8000 and ispGDX devices can be programmed with this method or by the TAP controller interface. Table 1 compares these devices.

Table 1. Lattice ISP Daisy Chain Comparison

	ispLSI 1000/E and 2000/A Families	ispLSI 3000 Family	ispLSI 8000 and ispGDX Families	ispGDS Family and ispGAL22V10
Lattice ISP Interface	Yes	Yes	Yes	Yes
ispJTAG Interface	No	Yes	Yes	No
Boundary Scan Test	No	Yes	Yes	No
Device ID	8-Bit ISP ID	8-Bit ISP ID	8-Bit ISP ID and 32-Bit Boundary Scan IDCODE	8-Bit ISP ID

The ispLSI 1000/E and 2000/A families are programmed exclusively through the Lattice ISP interface. They also use the dedicated $\overline{\text{ispEN}}$ pin to enable the programming mode: by driving $\overline{\text{ispEN}}$ low, all of the device I/O pins are put into a high-impedance state and the programming functions for SDI, SDO, MODE and SCLK are enabled. The ispGDS family and ispGAL22V10 use only the Lattice ISP interface, but they do not have or use a dedicated $\overline{\text{ispEN}}$ pin to enter programming mode. The I/O pins of ispGDS and ispGAL22V10 devices are put into a high impedance state when the programming state machine is not in the idle state and the ISP programming pins are always active.

The ispLSI 3000 family is programmable through either the Lattice ISP interface or the ispJTAG interface. By driving BSCAN/$\overline{\text{ispEN}}$ low, all the device I/O pins are put into the high-impedance state, the programming functions for SDI, SDO, MODE and SCLK are enabled and the device enters the programming mode. When BSCAN/$\overline{\text{ispEN}}$ is high, the TAP controller is active and the functions for TDI, TDO, TMS and TCK are enabled. With the TAP controller active, these devices enter and exit programming mode using private ispJTAG instructions. Device I/O pins go to a high-impedance state while in programming mode. In addition, boundary scan test operations are available for the ispLSI 3000 devices when the TAP controller is active. In summary, BSCAN/$\overline{\text{ispEN}}$ should be not connected or connected to Vcc in order to use the ispJTAG interface. The active pull-up will activate the ispJTAG interface if BSCAN/$\overline{\text{ispEN}}$ floats. To program using the Lattice ISP interface, BSCAN/$\overline{\text{ispEN}}$ needs to be brought out to the interface connector.

The ispLSI 8000 and ispGDX families are also programmable through either the Lattice ISP interface or the ispJTAG interface. Unlike the ispLSI 3000 family, these devices have both the JTAG 32-bit IDCODE and the Lattice 8-bit ID. By driving BSCAN/$\overline{\text{ispEN}}$ low, all device I/O pins are put into the high-impedance state, the programming functions for SDI, SDO, MODE and SCLK are enabled and the device enters the programming mode. When BSCAN/$\overline{\text{ispEN}}$ is high, the TAP controller is active and the functions for TDI, TDO, TMS and TCK are enabled. With the TAP controller active, these devices enter and exit programming mode using private ispJTAG instructions. The ispGDX I/O pins go to a high impedance state while programming mode. When the ispLSI 8000 devices are in ispJTAG programming mode, the I/O pins are under the control of the boundary scan test output latches. This allows the user to drive signals to other devices on the board while programming these devices. The optional JTAG pin, TOE, can be used to tri-state the I/O pins when the device is functional. In addition, boundary scan test operations are available for these devices when the TAP controller is active. In summary, BSCAN/$\overline{\text{ispEN}}$ should be not connected or connected to Vcc in order to use the ispJTAG interface. The active pull-up will activate the ispJTAG interface if BSCAN/$\overline{\text{ispEN}}$ floats. To program using the Lattice ISP interface, BSCAN/$\overline{\text{ispEN}}$ needs to be brought out to the programming interface connector.

ISP Programming for Mixed Lattice ISP Interface and ispJTAG Interface Systems

This section discusses the hardware interface when proprietary Lattice ISP interface devices are mixed with IEEE 1149.1 compliant devices on the same board. Following a few simple procedures will result in first time success for programming all ISP devices. Described here are the most typical configurations for system design based on common ISP and testability goals.

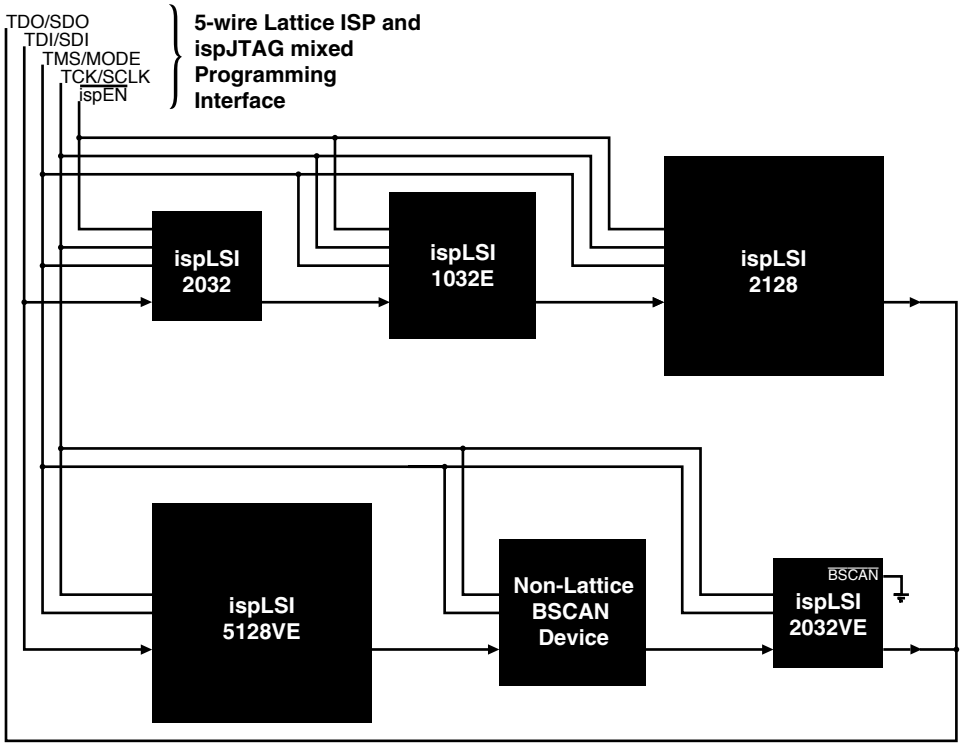
In general, most of the signals from the Lattice ISP interface can be common with the corresponding ispJTAG interface signals. This includes SDI and TDI, MODE and TMS, SDO and TDO, and SCLK and TCK. A parallel programming configuration can be used, where the proprietary Lattice ISP interface devices and the IEEE 1149.1 compliant devices are in two separate chains, both fed by SDI/TDI and combining SDO/TDO at the end of the chains (Figure 2). The $\overline{\text{ispEN}}$ pin can be used to disable the proprietary Lattice ISP interface chain. When these devices are not in programming mode, the ISP pins are in a high impedance state and will not affect the programming of ispJTAG devices. When programming the Lattice ISP interface devices, the TDO pin from ispJTAG devices needs to be in the high-impedance state so programming of the Lattice ISP interface daisy chain will not be affected. Using the appropriate hardware and software from Lattice will remove this programming challenge.

Tristatable buffers must be inserted before and after each ispGAL22V10. The ispGAL22V10 does not have an $\overline{\text{ispEN}}$ pin which is needed when programming a mixed ISP and ispJTAG chain.

Software Support

Lattice's ispVM™ System software and ISP Daisy Chain Download (ispDCD) software support mixed-chain programming. When creating the chain file, a marker is used between the ISP and ispJTAG chains to indicate which devices follow the Lattice proprietary ISP flow and the ispJTAG flow. More information about mixed-chain programming with ispVM System can be found in the ispVM System help menu.

Figure 2. Mixed Lattice ISP and ispJTAG Programming Interfaces



Board Layout Considerations

All ISP devices are shipped from Lattice with a fuse pattern that will put all I/O pins in the high impedance state prior to programming. This configuration prevents the ISP devices from driving unwanted signals to other devices in the system before they can be properly programmed.

The ispLSI 3000, 8000 and ispGDX families can exist in either a boundary scan or a Lattice ISP interface serial daisy chain for test and programming purposes. The configuration choice for these devices depends on the boundary scan test operations needed, programming requirements and device combinations. For both boundary scan test operations and programming through the ispJTAG interface, these devices may be put in a chain including non-Lattice boundary scan devices. If a mixed chain programming configuration is used, these devices need to be in ispJTAG interface mode.

In addition to using good PCB layout practices, including the use of decoupling capacitors between Vcc and ground on each ISP device and minimizing trace lengths wherever possible, additional care must be taken to ensure programming interface signal integrity. A filtering capacitor (.01 μF) must be provided between the $\overline{\text{ispEN}}$ signal and ground. This filtering capacitor must be located as close as possible to the ISP connector on the PCB, in order to filter out any noise during programming. During programming, the $\overline{\text{ispEN}}$ signal is driven low. Without the capacitor, noise can couple into the $\overline{\text{ispEN}}$ signal during programming and could interrupt the programming sequence. This applies to ispLSI 3000, 8000 and ispGDX devices for the BSCAN/$\overline{\text{ispEN}}$ signal when programming in Lattice ISP mode.

When long serial daisy chains of devices are used, special consideration needs to be placed on the MODE/TMS, SCLK/TCK, and $\overline{\text{ispEN}}$ signals. These signals are connected in parallel to all devices and may require that buffers be used. No buffer is necessary on the SDI/TDI line since only the first device in the chain is driven. SDI/TDI for each subsequent device is driven from SDO/TDO of the previous device. High speed CMOS (MCMOS) or TTL (AS, ALS) devices may be used as additional buffers. The additional buffers should be placed in parallel in order to minimize the timing skew between the programming signals. With additional buffers, it is recommended that a filtering

capacitor (.01 μF) be added on the output side of the buffers for the $\overline{\text{ispEN}}$ signal, in addition to the filtering capacitor (.01 μF) on the input side of the buffer.

The last device in the chain must drive SDO/TDO. The output drive of that device pin should be considered if a long programming cable is used. It may be necessary to add a line driver with a higher current drive to the PCB.

ISP programming signal default states must be maintained during normal device operation. The $\overline{\text{ispEN}}$ pin on the ispLSI 1000/E and ispLSI 2000/A devices has an internal pull-up to place the devices in normal functional mode when the pin is not driven externally. The $\overline{\text{ispEN}}$ and BSCAN pin for the ispLSI 2000E, 2000VE, 2000VL and 2000V devices has an internal pull-up that will cause the TAP inputs to be disabled if it is allowed to float. The BSCAN/ $\overline{\text{ispEN}}$ pin on the ispLSI 3000, 8000 and ispGDX devices has an internal pull-up that puts the devices into ispJTAG interface mode when the pin is not driven externally. The EPEN pin for the ispLSI 8000V and ispGDXV devices has an internal pull-up that enables the TAP controller when it is not driven externally. The ispGAL22V10B and ispGDS devices' MODE or SDI signals must be tied low through a 1.2KW pull-down resistor during normal functional mode. It is not acceptable to let these pins float during normal operation. However, the ispGAL22V10C devices provide an internal pull-down on SDI to maintain socket compatibility with the standard 22V10 in the PLCC package.

User Electronic Signature

The UES is incorporated on all ISP devices in the ispLSI 1000/E, 2000/A, 3000, 8000, ispGDX, ispGAL and ispGDS families. The UES is part of the JEDEC file fusemap for ispGAL and ispGDS devices and is contained in the U-field for ispLSI devices. Physically, the UES is an extra row that is appended to the programmable array. The size of the UES varies by device type.

The UES may be accessed (read or write) through one of three methods. First, most third-party programmers support the UES option for the ispGAL and ispGDS devices through the programmer's user interface, so programming or verifying the UES is as simple as programming or verifying any other array. Second, the UES may be embedded within the JEDEC file directly, the Lattice design software or the ispVM™ System download software. And third, the UES can be written or read using Lattice embedded source code. Further information on using Lattice software to access the UES can be found in the individual software data sheets in the Lattice Data Book CD-ROM or the Lattice Semiconductor web site at www.latticesemi.com.

Technical Support Assistance

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