ISP Architecture and Programming

Introduction

This section describes how to program Lattice Semiconductor Corporation's (LSC) ISP™ devices once the JEDEC standard fuse map file has been generated. It is divided into two subsections. The first subsection "Getting Started Fast" is intended to give the reader enough ISP hardware information to easily implement LSC's ISP solutions using the LSC ISP tools, which are briefly described at the end of this section. The second subsection "ISP Expert" gives more details on low-level, device-specific programming algorithms. Since these algorithms are transparently handled by LSC's programming tools, the second subsection is intended for those readers who want a thorough understanding of the programming procedures, which would be required for any custom implementation of ISP.

Subsection I — Getting Started Fast

In-System Programming (ISP) Interface
LSC ISP State Machine
TAP Controller State Machine
ISP Device Programming Configurations
Hardware Considerations
Hardware Programming Tools
ISP Programming Software
ISP Programming Times
User-Programmable ID Registers

Subsection II — ISP Expert

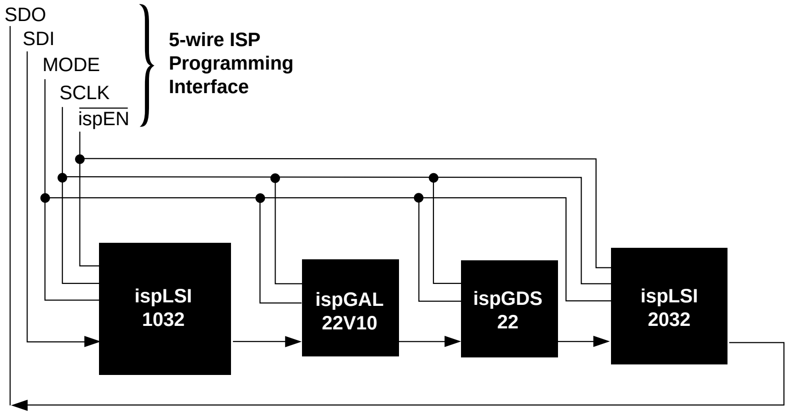
ispLSI® Programming Details
Boundary Scan (ispLSI 3000 & 6000 Families)
ispGDS™ Programming Details
ispGAL® Programming Details
ISP Daisy Chain Details

In-System Programming (ISP) Interface

Lattice's ISP devices utilize nonvolatile E^2CMOS® technology and require only 3.3V or 5V, TTL-level programming signals. An integrated state machine controls the sequence of programming operations such as identifying the ISP device, shifting in the appropriate data and commands, and controlling internal signals to program and erase the E^2 Cells in the device. Programming consists of serially shifting the logic implementation stored in a JEDEC file into the device along with appropriate address and commands, programming the data into the E^2CMOS logic elements, and shifting the data from the logic array out for device programming verification.

The ISP programming interface to Lattice Semiconductor ISP devices is controlled by either the LSC proprietary ISP programming interface or the IEEE Std 1149.1-1990 boundary-scan Test Access Port (TAP). The LSC ISP programming interface controls an integrated three-state programming state machine while the boundary-scan TAP controls programming through an IEEE1149.1 specified state machine called the TAP controller. Lattice

Figure 1. Multiple ISP Device Programming Interface



Semiconductor's ispGDS, ispGAL22V10B, ispGAL22V10C, ispLSI 1000/E families, ispLSI 2000 family, ispLSI 3000 family, and ispLSI 6000 family are all programmed through the LSC ISP programming interface. ispLSI 2000LV family devices are programmed exclusively through the boundary-scan TAP controller.

The basic elements of the LSC ISP programming interface are the mode control (MODE), serial data in (SDI), serial data out (SDO), and serial clock (SCLK) signals (Figure 1). The internal three-state state machine, which determines whether the device is in the normal operation state or the programming states, is controlled by the four ISP programming pins. MODE and SDI furnish control inputs to the state machine, SDI and SDO make up the programming data inputs and outputs to and from the internal shift register, and SCLK provides the clock. ispLSI devices 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 TAP controller interface as specified in the IEEE Std 1149.1-1990 document must include the Test Mode Select (TMS), Test Data In (TDI), Test Data Out (TDO), and Test Clock (TCK) signals. These signals perform similar duty for state machine control as the LSC ISP connections MODE, SDI, SDO, and SCLK respectively. However, the TAP controller state machine is exclusively controlled by the TMS signal, and TDI is only used for shifting in data and instructions. ispLSI devices use a fifth pin, $\overline{\text{ispEN}}$ to multiplex the functions of the TDI, TDO, TCK and TMS pins between TAPcontroller functions and either user-defined logic functions or other programming functions. An optional Test Reset ($\overline{\text{TRST}}$) pin is used to asynchronously reset the TAP controller.

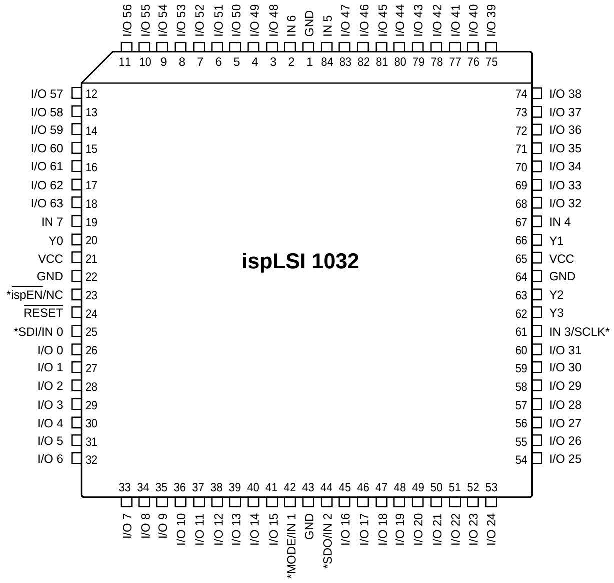
LSC ISP State Machine

LSC ISP State Machine Programming Pins

The programming pins used to program Lattice Semiconductor devices that use the LSC ISP state machine are each described in detail in this section. Figure 2 shows the ispLSI 1032 84-Pin PLCC device pinout.

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 ISP device. Second, it functions as one of the 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, and when

Figure 2. ispLSI 1032 84-Pin PLCC Pinout Diagram



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.

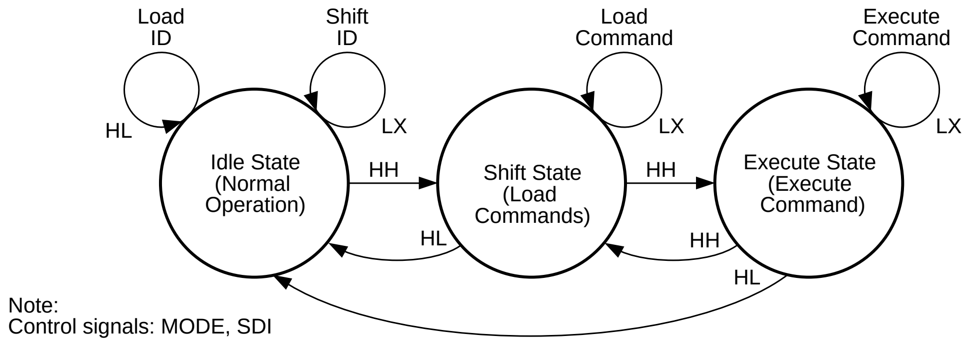
The MODE signal, combined with the SDI signal, controls the programming state machine, as described in theww which follows.

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 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, and 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 ISP state machine's shift instruction. When MODE is driven high, SDO connects directly to SDI, bypassing the device's shift registers.

The $\overline{\text{ispEN}}$ pin, only utilized on the ispLSI devices, determines which mode the device is in, namely *ISP programming mode* or *normal mode* (normal device operation mode). When $\overline{\text{ispEN}}$ is driven low on an LSC

Figure 3. LSC ISP Programming State Machine



ISP state machine-programmable device, the device I/O pins are put into a high impedance state (by internal active pull-up resistors equivalent to 100KΩ) and the device enters the programming mode.

LSC ISP State Machine Operation

The programming state machine controls which mode the device is in, and provides the means to read and write data to the device (Figure 3). The three states defined in the LSC 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. In the SHIFT and EXECUTE states, all the I/O pins are 3-stated. To transition between states, MODE is held high, SDI is set to the appropriate level, and SCLK is clocked. The ispGAL22V10B/C and ispGDS devices, unlike ispLSI devices which employ an $\overline{\text{ispEN}}$ input pin, rely on the state machine to put the device I/O pins in a high impedance state. The IDLE state puts the ispGAL and ispGDS devices into normal mode, and the remaining two states put the devices into ISP programming mode, which places the device I/O pins in the high impedance state.

Idle/ID State

The Idle/ID state is the first state activated when the device is powered-up or the $\overline{\text{ispEN}}$ pin is driven low. The state machine is in the Idle/ID state when the user needs to read the device identification (each ISP device type is assigned a unique identification code—see the ISP Expert section). The eight-bit device identification is loaded into the shift register by driving MODE high, SDI low, and clocking the ISP state machine with SCLK. Once the 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”) and the state machine needs only seven clocks to read out eight bits of device ID. The default state for the control signals is MODE high and SDI low. State transition to the Command Shift State occurs when both MODE and SDI are high while the ISP state machine gets a clock transition. As with most shift registers, the Least Significant Bit (LSB) of the ID gets shifted out from SDO first.

Command Shift State

This state is strictly used for shifting instructions into the state machine. The entire LSC ISP state machine instruction sets for the ispLSI, ispGDS, and ispGAL devices are listed in the ISP Expert section. When MODE is low and SDI is “don’t care” in the Command 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 Command Shift State to the Execute State. If needed, the state machine can move from the Command Shift State to the Idle/ID 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 Command 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 appropriate data sheets.

To execute a command, MODE is driven low and SDI is "don't care." For multiple clock instructions, the control signals must remain in the same state throughout the duration of the execution. MODE high and SDI high will take the state machine back to the Command Shift State and MODE high and SDI low will take the state machine to the Idle/ID State.

TAP Controller State Machine

TAP Controller State Machine Programming Pins

The programming pins used to interface to the TAP controller are described in detail in this section. Figure 4 shows the ispLSI 2032LV device pinout.

The TDI pin is the input pin to load instructions and data into the device. Internally, the TDI pin is multiplexed to different shift registers in the device. The instructions loaded determine which internal shift register will receive input from the TDI pin.

The TMS pin is used to control the state of the TAP controller. The signal present on TMS is sampled on the rising edge of TCK.

The TDO pin is the output of the serial shift register. Data from internal registers will be available at TDO on the falling edge of TCK. The instruction loaded into the TAP controller selects which shift register will be active.

The TCK pin is used to clock the internal serial shift registers and the TAP controller state machine. State changes and shifting data in are performed on the rising edge of TCK, while data is clocked out on the falling edge of TCK.

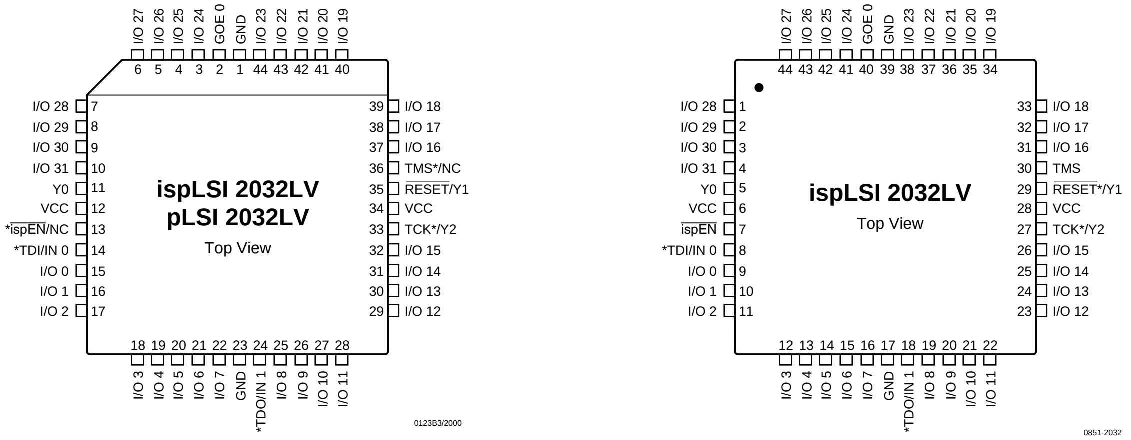
The $\overline{\text{ispEN}}$ pin multiplexes functionality of programming pins for ispLSI devices. For the ispLSI 2000LV family, when $\overline{\text{ispEN}}$ is driven low, the TAP controller pins are enabled and the TAP controller may be accessed. For the ispLSI 3000 and ispLSI 6000 families, when the $\overline{\text{ispEN}}$ pin is low, the LSC ISP state machine is active. While the $\overline{\text{ispEN}}$ pin is high, the TAP controller is active. With the exception of the ispLSI 3256, all of the ispLSI 3000 and ispLSI 6000 families of devices have the ability to be programmed through either the LSC ISP state machine or the TAP controller state machine. The ispLSI 3256 is programmed through the LSC ISP state machine only.

TAP Controller State Machine Operation

The boundary scan TAP controller is the IEEE 1149.1 specified state machine. Four control pins are used to load and unload data, TMS, TDI, TDO, and TCK. For a detailed description and specifications for each state consult the IEEE Standard Test Access Port and Boundary-Scan Architecture document. A diagram of the Tap Controller is shown in Figure 5.

While the test access port and boundary-scan architecture was developed to standardize system testing, the TAP controller can be used for in-system programming. The TAP controller is used for programming by loading instructions in the Shift-IR state, loading address information and programming data in the Shift-DR state, and

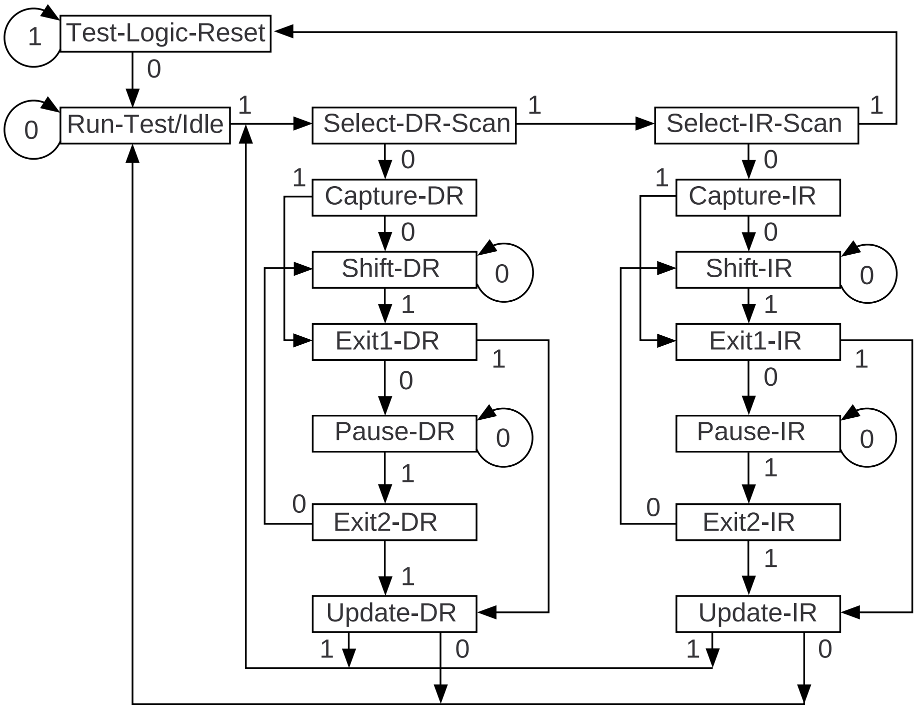
Figure 4. ispLSI 2032LV 44-Pin PLCC and TQFP Pinout Diagrams



* Pins have dual function capability for ispLSI 2032LV only (except pin 13, which is $\overline{\text{ispEN}}$ only).

* Pins have dual function capability.

Figure 5. The TAP Controller



carrying out programming instructions in the Run-Test/Idle state. To enter the programming mode in ispLSI boundary-scan TAP ISP devices, the Program Enable (ProgEN) instruction is loaded three times in succession. After the first time it is loaded, the functional I/Os enter the high-impedance state. After the third time it is loaded, the part enters the programming mode. Once in the programming mode, the part is ready for programming instructions. After loading an instruction and associated data, the programming pulse is applied. The TAP controller is clocked to the Run-Test/Idle state and the programming pulse starts on the first rising edge of the clock with TMS low. The programming pulse is terminated by exiting the Run-Test/Idle state.

ISP Device Programming Configurations

Serial Daisy Chain

Advantages

One of the main advantages of daisy-chained ISP programming is the simplified hardware interface. The number of ISP devices that can be connected to the same serial interface is limited only by the signal drive capability of the ISP programming control logic. One serial daisy chain is capable of providing the necessary programming interface, minimizing the hardware overhead for in-system programming. Software controls generated from PCs, microcontrollers, and test equipment can program and reconfigure ISP devices during various board-level design, test, and manufacturing stages.

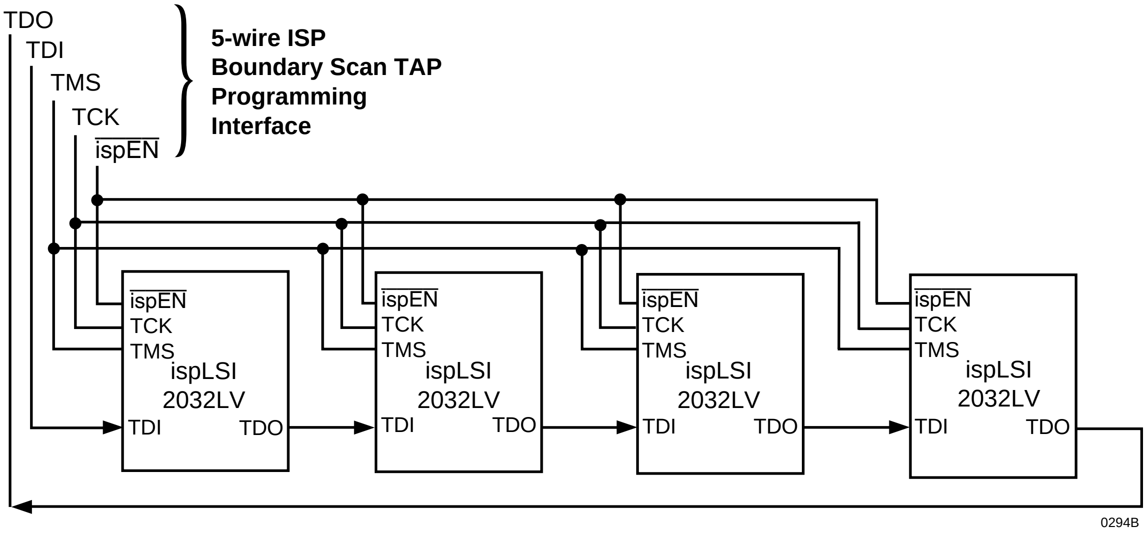
Programming Configuration

As shown previously in Figure 1, all the MODE, SCLK, and $\overline{\text{ispEN}}$ (if using ispLSI devices) pins of the LSC ISP devices are connected to the ISP interface, and the first device's SDO is connected to the second device's SDI and each following SDO to the SDI of the next ISP device. For boundary-scan TAP ISP programmable devices the TMS, TCK and $\overline{\text{ispEN}}$ pins are shared and TDO is connected to TDI of the next device. Figure 6 shows the serial boundary-scan test access port programming interface. This configuration allows a large string of ISP devices to be programmed in-system, in a serial daisy chain.

Parallel

For low-density ISP devices daisy chain programming is the most common configuration, but for high-density devices, with multiplexed programming and logic pins controlled by $\overline{\text{ispEN}}$, other programming configurations are also common. ISP devices can be programmed in one of two parallel configurations. The first parallel con-

Figure 6. Serial Boundary-Scan TAP Programming Interface



figuration, called Dedicated ISP Pins, dedicates all ISP programming pins to programming. The second parallel configuration, called Parallel Multiplex, allows the functions of the ISP programming pins to be multiplexed between acting as programming pins and acting as inputs for normal logic functions. Both the dedicated ISP pins and Parallel Multiplex configurations can be used with parallel combinations of ispLSI 1000 family, ispLSI 2000 family, and ispLSI 2000LV family devices. The ispLSI 3000 and ispLSI 6000 family devices use ISP pins for both the LSC ISP state machine interface and the TAP controller interface and therefore cannot be put in the dedicated ISP pins or Parallel Multiplex configurations due to interference with TAP controller operations.

Dedicated ISP Pins

Figure 7 illustrates one configuration for programming multiple ISP devices, where the ISP programming pins (MODE, SDI, SDO, and SCLK) are dedicated to programming functions. For the boundary-scan TAP interface the corresponding pins (TMS, TDI, TDO, and TCK) can be connected in the same configuration. Although this scheme precludes the use of the ISP programming control signal pins as separate dedicated inputs for system logic functions on ispLSI devices, it is the easiest to implement. Each of the four programming control signal pins in each ISP device is connected (i.e. SDI of the ispLSI 1032 is connected to SDI of the ispLSI 1048 and SDI of the ispLSI 1016; MODE of the ispLSI 1032 is connected to MODE of the ispLSI 1048 and MODE of the ispLSI 1016; etc.). With this scheme, the $\overline{\text{ispEN}}$ signal for each ispLSI device is enabled ($\overline{\text{ispEN}}$ low) independently, and one device is placed in the programming mode at a time. With one device in the programming mode, the other devices will be in normal mode and can continue to perform normal system logic functions.

Parallel Multiplex

Figure 8 illustrates a multiplexing scheme which allows the user to control the ISP programming through multiple

Figure 7. Dedicated ISP Pins Configuration

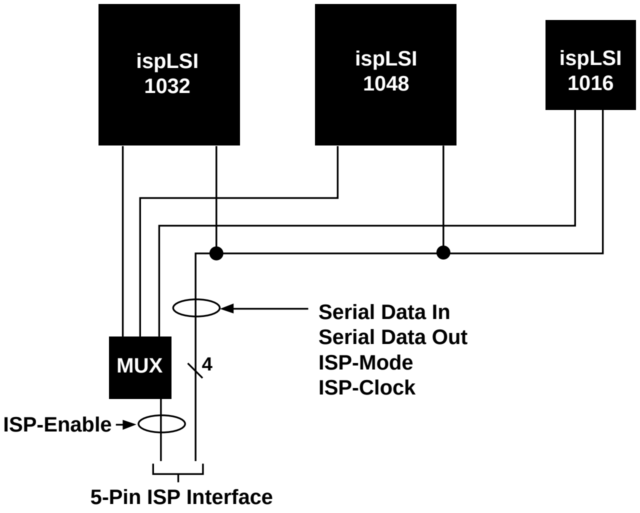
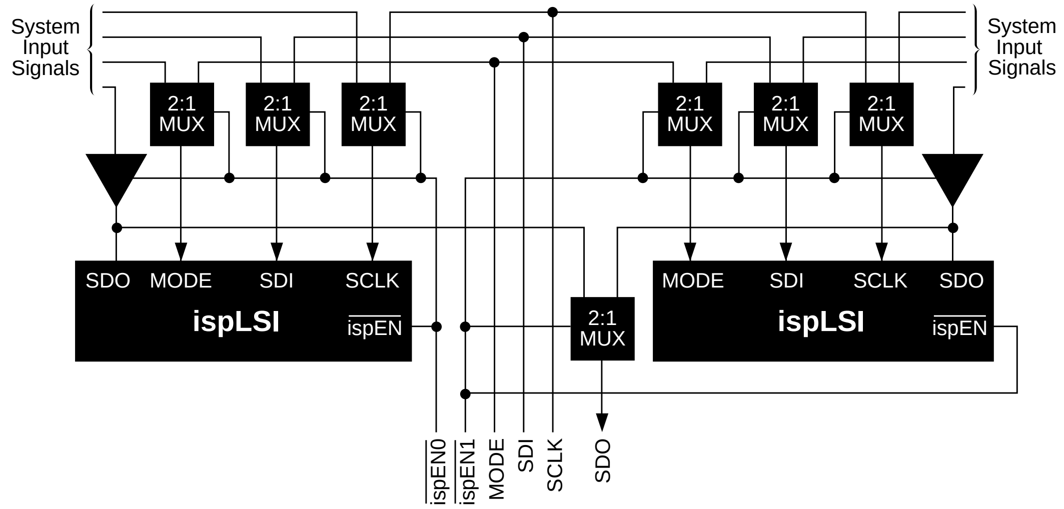


Figure 8. Parallel Multiplex Configuration



independent $\overline{\text{ispEN}}$ signals for the ispLSI devices. The multiple $\overline{\text{ispEN}}$ signals not only control the $\overline{\text{ispEN}}$ inputs of the ispLSI devices, but also act as the control signals for multiplexing the functional and ISP programming signals. This scheme differs from the previous one in that the ISP programming signals are not dedicated to programming. Instead, the ISP programming signals function as ISP state machine inputs and outputs for programming mode functions and dedicated inputs (if available) for normal functional mode. Figure 8 also shows the difference in controlling these different programming signals. When multiplexing the programming interface signals, the input driving the SDO/TDO pin must be put into high-impedance state during programming to avoid signal contention. As previously stated, the ISP programming pins on the ispGAL and ispGDS devices are dedicated to ISP programming, so this configuration is not utilized often for the ispGAL and ispGDS devices. The concept can be modified to multiplex the MODE pin instead of the $\overline{\text{ispEN}}$ pin and becomes useful in some ispGAL and ispGDS applications.

Hardware Considerations

Lattice Semiconductor's ISP technology makes the use of programmable logic incredibly simple. Using ISP, multiple devices can be programmed using a single serial daisy chain programming loop. However, as with any high performance semiconductor component, systems must be designed to insure good signal integrity without signal conflicts between components. By doing so, reliable operation can be obtained over a wide range of operating conditions. This section discusses some basic programming hardware issues which should be considered when implementing a system using ISP.

All ISP programming specifications such as the programming cycle and data retention are guaranteed when programming ISP devices over the commercial temperature range (0 to 70° C). It is critical that the programming and bulk erase pulse width specifications are met by the programming platform to insure proper in-system programming. LSC's ISP Daisy Chain Download and ispCODE™ software ensures that these specifications are met when using a PC programming platform.

When using the ispDOWNLOAD™ cable in a daisy-chained configuration, Lattice recommends using a maximum of eight ISP devices in a single chain. This is to ensure proper programming signal integrity (pulse width, shape, etc.) at the ISP devices. The recommended number of devices is based on a typical system board environment with proper signal terminations and typical trace lengths. The actual number of devices that can be programmed in a serial chain may vary according to the system board environment. When using more than eight devices, additional buffering of the ISP programming signals is recommended. Alternatively, multiple programming loops can be employed which are electrically isolated from one another.

I/O pins on ISP devices may be defined as inputs once the devices are programmed. As a result, they typically will be driven by the outputs of other components once mounted on the board. Care must be taken to ensure that I/O pins are not enabled prematurely during programming. To do so when the device is partially programmed can cause contention with other signal drivers since I/O pins destined to be configured as inputs may not be 3-stated yet. This conflict can cause improper device programming and potential damage (Figure 9).

All ISP devices are shipped from Lattice Semiconductor with a fuse pattern that will put all I/O pins in the high impedance state prior to programming. For ispLSI 1000 family and ispLSI 2000 family devices the I/O pins are put into the high-impedance state by asserting the $\overline{\text{ispEN}}$ pin low. For the ispLSI 2000LV family of devices the $\overline{\text{ispEN}}$ pin must be asserted low and one Program Enable instruction executed to set the I/O pins into the high-impedance state. To put the I/O pins in the high-impedance state for the ispLSI 3000 and ispLSI 6000 family devices either the $\overline{\text{ispEN}}$ pin can be asserted low or one Program Enable instruction loaded in the TAP controller state machine (the ProgEN instruction is not supported for the ispLSI 3256). For the ispGAL and ispGDS devices, the output 3-state is controlled by the programming state machine (Shift and Execute states put I/O pins in the high-impedance state). When implementing custom ISP programming code, it is important for the ispGAL and ispGDS that the ISP state machine be kept within the Shift and Execute states until the completion of programming. This procedure keeps the partially programmed device or devices from conflicting with other components on the board.

ISP programming signal default states must be maintained during normal device operation. The $\overline{\text{ispEN}}$ pin on the ispLSI devices has an internal pull-up to place the devices in normal functional mode when the pin is not driven externally. The ispGAL22V10B and ispGDS devices' MODE or SDI signals must be tied low through a 1.2KΩ 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. In addition, it is recommended that the ispDOWNLOAD cable have its $\overline{\text{ispEN}}$ signal tied to a decoupling capacitor (.01µF) to ground on the system board.

Hardware Programming Tools

isp Engineering Kit

Lattice Semiconductor provides a PC-based (Model 100) isp Engineering Kit. The isp Engineering Kit functions as a stand-alone device programmer for prototyping.

isp Engineering Kit Model 100

The isp Engineering Kit Model 100 provides designers with a quick and inexpensive means of evaluating and

Figure 9. ISP Serial Daisy Chain

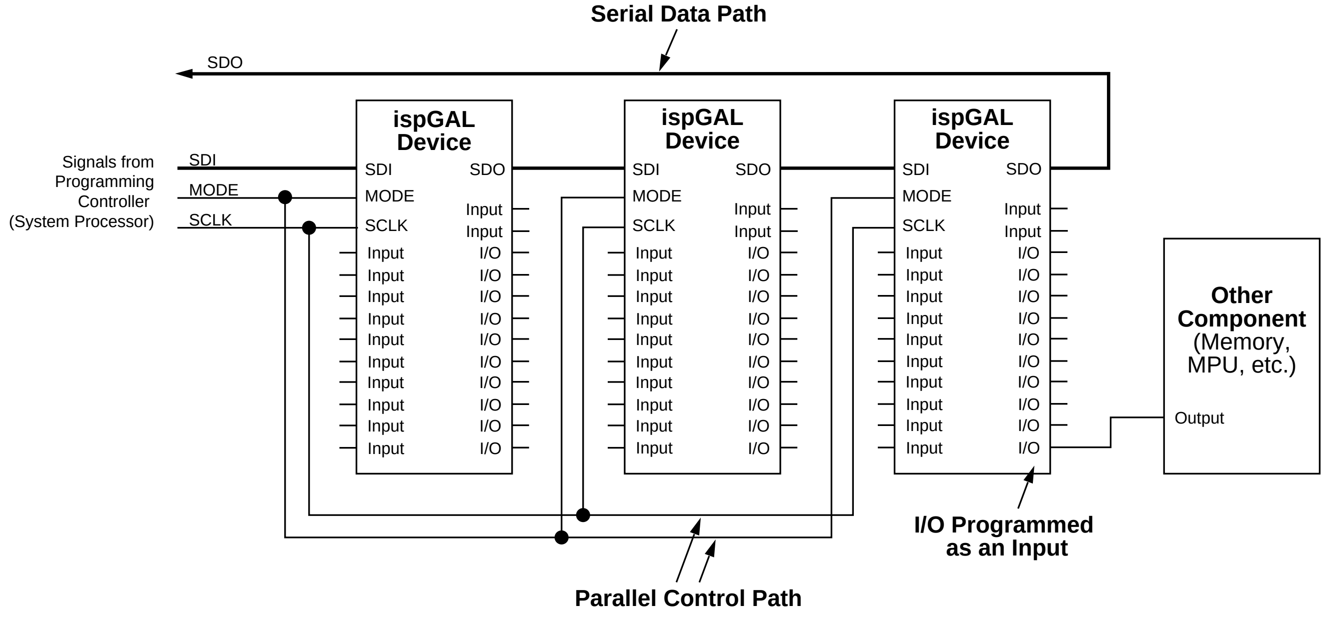
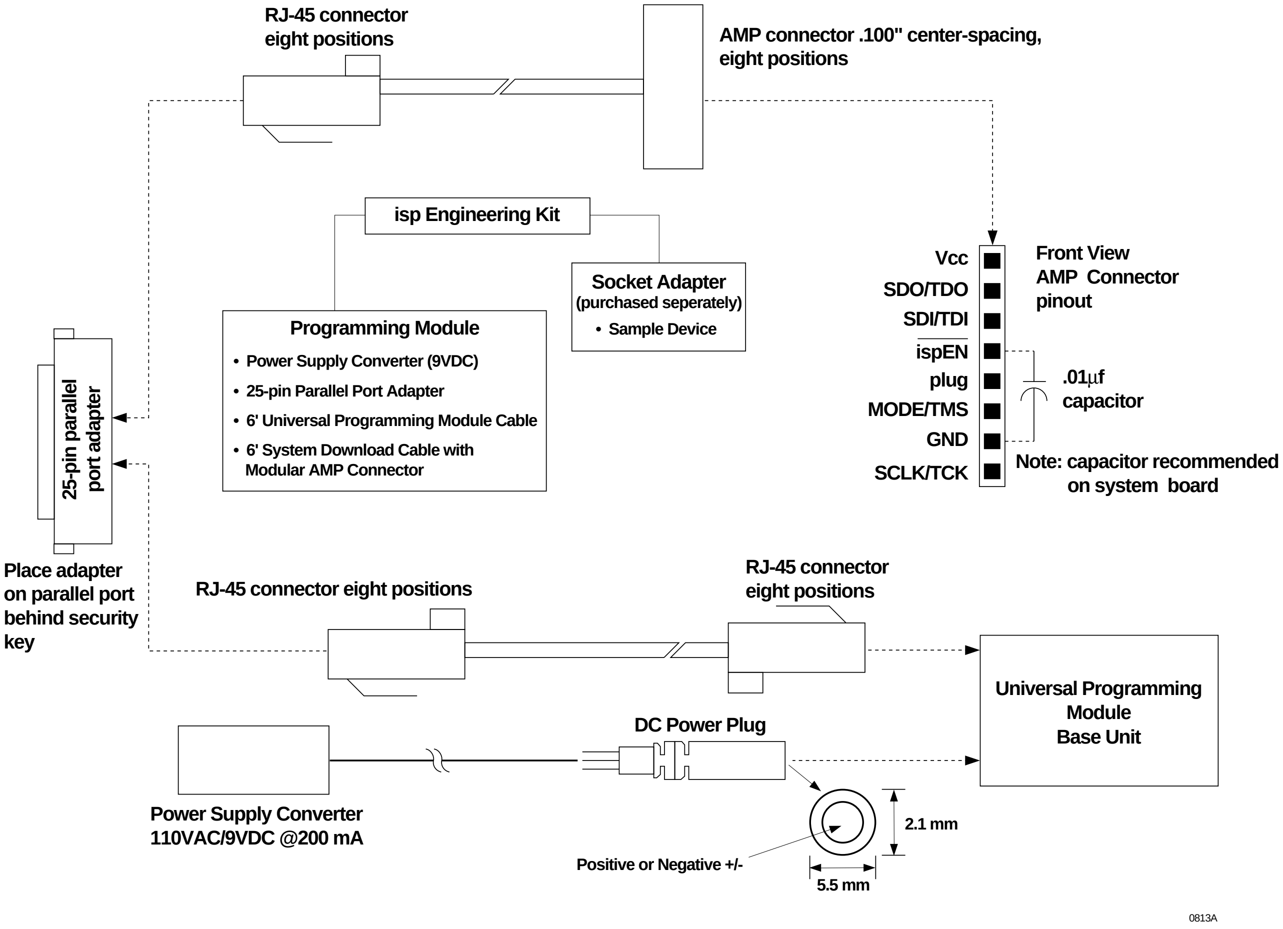

Serial Data Path
SDO
Signals from Programming Controller (System Processor)
SDI
MODE
SCLK
ispGAL Device
SDI
SDO
MODE
SCLK
Input
I/O
Other Component (Memory, MPU, etc.)
Output
I/O Programmed as an Input
Parallel Control Path

Figure 10. ispEngineering Kit Model 100


RJ-45 connector
eight positions
AMP connector .100" center-spacing, eight positions
isp Engineering Kit
Socket Adapter
(purchased seperately)
• Sample Device
Programming Module
• Power Supply Converter (9VDC)
• 25-pin Parallel Port Adapter
• 6' Universal Programming Module Cable
• 6' System Download Cable with Modular AMP Connector
25-pin parallel port adapter
Place adapter on parallel port behind security key
Vcc
SDO/TDO
SDI/TDI
ispEN
plug
MODE/TMS
GND
SCLK/TCK
Front View AMP Connector pinout
.01µf capacitor
Note: capacitor recommended on system board
RJ-45 connector eight positions
RJ-45 connector
eight positions
Universal Programming Module Base Unit
DC Power Plug
Power Supply Converter
110VAC/9VDC @200 mA
Positive or Negative +/-
2.1 mm
5.5 mm
0813A

prototyping new designs using LSC ispLSI devices. This kit is designed for engineering purposes only and is not intended for production use. The kit programs devices from the parallel printer port of a host PC using the LSC pDS® or pDS+™ PC-based designs tools. By connecting a system cable (included) from the host PC to the isp Engineering Kit, or connecting from the host PC to the target device on the system board, a JEDEC file can be easily downloaded into the ispLSI device(s) (Figure 10).

ispDOWNLOAD Cable

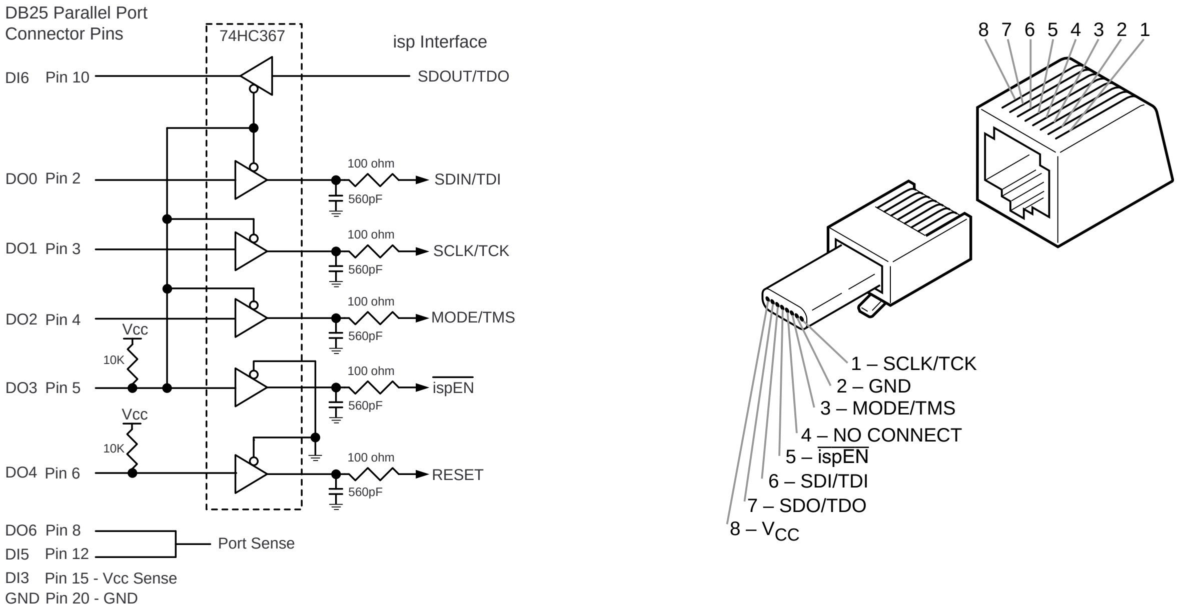
The ispDOWNLOAD cable is designed to facilitate in-system programming of all LSC ISP devices on a printed circuit board directly from the parallel port of a PC. After completion of the logic design and creation of a JEDEC file by a logic compiler such as the pDS, pDS+ Fitter or ispGDS Compiler software, Lattice's ISP Daisy Chain Download software programs devices on the end-product PC board by generating programming signals directly from the parallel port of a PC which then pass through the ispDOWNLOAD cable to the device. With this cable and a connector on the PC board, no additional components are required to program a device (Figure 11).

ISP Synario System

The ISP Synario System is designed to make Lattice's innovative in-system programmable device technology available in a single, complete package. The ISP Synario System contains all the software, hardware, device samples and information required for ISP design.

The ISP Synario System is based on the popular Synario Entry tool from Data I/O and a version of LSC's pDS+ Synario Fitter supporting ispLSI and pLSI devices up to 1024/2096 device densities. Designs may be entered via Synario Schematic Capture or using ABEL-HDL. Functional Simulation, Project Navigator and the LSC Fitter are included, along with the ispGDS Compiler, ispCODE, ISP Daisy Chain Download software and ispDOWNLOAD cables. Device samples include the ispLSI 2032, ispGAL22V10B and an ispGDS 14 device. In addition, the ISP Synario System includes GAL compiler support for all LSC GAL devices.

Figure 11. ispDOWNLOAD Cable



ISP Programming Software

Introduction

Once the JEDEC file has been generated for a given design, the design information must be downloaded into the proper device. The download method depends on the hardware available and the design stage. For example, you might program the system with ISP devices during prototyping using a PC. Then, when the system goes to full production, you can use ATE for programming. Finally, if field upgrades are necessary, you can use the system's embedded microprocessor to reprogram the ISP devices. Table 1 summarizes the download methods supported by Lattice.

ispCODE

ispCODE is C-source code that facilitates in-system programming of LSC ISP devices on UNIX systems, PCs, testers and embedded systems. The ispCODE software supplies specific routines, with extensively commented code, for incorporation into user application programs. This software is available from Lattice Semiconductor. For a more thorough description of ispCODE, refer to the ispCODE data sheet in the 1996 Lattice Semiconductor Data Book.

ISP Daisy Chain Download

ISP Daisy Chain Download software supports programming of all LSC ISP devices in a serial daisy chain programming configuration in a PC environment. Two varieties of this software exist: one for a Windows environment, the other for a DOS environment. This software is available from Lattice Semiconductor. For a more thorough description of the ISP Daisy Chain Download software, refer to the ISP Daisy Chain Download Software data sheet in the 1996 Lattice Semiconductor Data Book.

ispATE™

LSC's ispATE is a test-vector creation utility that facilitates programming of LSC ISP devices on HP, Teradyne and GenRad testers. ispATE converts a standard JEDEC file into a programming vector template that can be easily incorporated into a product's printed circuit board functional test program.

Table 1. ISP Programming Platform and Download Methods

Programming Platform	Download Methods
PC	ISP Daisy Chain Download ispCODE C Source Routines
Workstation	ispCODE C Source Routines
Embedded Processor	ispCODE Executed by Microprocessor
ATE	ispATE
Third-Party Programmer	Standard JEDEC File Download

ISP Programming Times

The ISP programming times can be approximated by the number of rows that are required to program on a given device and the programming pulse width. Assuming that the overhead of shifting data and other miscellaneous functions are an order of magnitude smaller in time duration and therefore negligible, the total programming time ranges can be calculated.

Calculating Programming Times

ISP programming times can be approximated by the number of rows that are required to program on a given device and the programming pulse width. Assuming that the overhead of shifting data and other miscellaneous functions are several orders of magnitude smaller in time duration and therefore negligible, the total programming time ranges can be calculated using this equation:

tpt = asrl * dr * tpwp (minimum)

where:

tpt: total programming time, ISP devices
asrl: address SR length from Table 10 or
ispGDS = 11
ispGAL22V10 = 44
dr: number of data registers,
ispGDS and ispGAL22V10 = 1
all other ISP devices = 2
tpwp: programming pulse width time,
see Tables 7, 9 or 15.

Example ispLSI 1016-90 total programming time:

tpt = 96 * 2 * 40 ms = 7.68 sec

To minimize the total programming time of a daisy chain of ISP devices, a programming method called ispTURBO Download can be used with Windows/DOS Daisy Chain Download or ATE to program all the ISP devices in the chain concurrently. ispTURBO Download allows programming of any number of ISP devices in the time it takes to program only the largest device. For example, a chain of three devices with programming times of ten, seven and seven seconds can be programmed with ispTURBO Download in a total of ten seconds (the time it takes to program the largest device). Serially, the programming time would be 24 seconds for all three devices. This valuable feature of Lattice's proprietary UltraMOS E^{2}CMOS technology is not available with many other ISP CPLD device technologies.

User-Programmable ID Registers

A user-programmable identification can ease problems associated with document control and device traceability. ISP devices that use the LSC ISP state machine for programming contain a register called the User Electronic Signature (UES). Lattice ISP devices that program exclusively through the boundary-scan TAP contain a 32-bit register accessible through the USERCODE instruction. The user-programmable ID register is basically a user's "notepad" provided in electrically erasable (E^2) cells on each device.

In the course of system development and production, the proliferation of PLD architectures and patterns can be significant. To further complicate the record-keeping process, design changes often occur, especially in the early stages of product development. The task of maintaining which pattern goes into what device for which socket becomes exceedingly difficult. Once a manufacturing flow has been set, it becomes important to "label" each PLD with pertinent manufacturing information, which is beneficial in the event of a customer problem or return. A user-programmable ID register was incorporated into

Table 2. UES Sizes

Device	UES Size (Bits)
ispGAL 22V10	64
ispGDS	32
ispLSI 1016/E	80
ispLSI 1024/E	120
ispLSI 1032/E	160
ispLSI 1048/C/E	240
ispLSI 2032	40
ispLSI 2064	80
ispLSI 2096	120
ispLSI 2128	160
ispLSI 3160	160
ispLSI 3192	240
ispLSI 3256/A/E	160
ispLSI 6192	160

UES Number Table 2

ISP devices to store such design and manufacturing data as the manufacturer's ID, programming date, programmer make, pattern code, checksum, PCB location, revision number, and/or product flow. This assists users with the complex chore of record maintenance and product flow control. In practice, the user-programmable ID register can be used for any of a number of ID functions.

Within the various bits available for data storage, users may find it helpful to define specific fields to make better use of the available storage. A field may use only one bit (or all bits), and can store a wide variety of information. The possibilities for these fields are endless, and their definition is completely up to the user.

Even with the device's security feature enabled, the user-programmable ID can still be read. With a pattern code stored in the user-programmable identification register, the user can always identify which pattern has been used in a given device. As a second safety feature, when a device is erased and re-patterned, the user-programmable identification is automatically erased. This prevents any situation in which an old programmable ID might be associated with a new pattern.

It is the user's responsibility to update the user-programmable ID when reprogramming. It should be noted that user-programmable identification information will be included in the checksum reading for ispGAL and ispGDS devices. Therefore, when the user-programmable ID is modified on these two device types, the checksum will also change. For ispLSI devices, the user-programmable ID will not affect the checksum.

User Electronic Signature

The UES is incorporated on all ISP devices that are programmable through the LSC ISP state machine. The UES is part of the JEDEC file 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. Table 2 indicates the various sizes of the UES.

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 or the ISP Daisy Chain Download software by selecting the UES menu option from the software for the ispGAL, ispGDS and ispLSI devices. And third, the UES can be written or read using Lattice's ispCODE software. Further information on using ispCODE software to program the UES can be found in the ispCODE Software section of the 1996 Lattice Semiconductor Data Book.

USERCODE

ISP devices programmable exclusively through the TAP controller (2000V family) contain a 32-bit, boundary-scan-compliant USERCODE. Loading of the USERCODE instruction makes the USERCODE available to be shifted out in the Shift-DR state of the TAP controller. The USERCODE instruction can be used while the device is in normal functional operation allowing the device to be scanned while operating.

ispLSI Programming Details

The following sections describe the programmable state machine instruction set, timing parameters, device layout, and programming algorithms for ispLSI devices. Programming steps and specifications for both the LSC ISP state machine and programming through the boundary-scan TAP controller will be given. The first step in programming any ISP device is to determine the device type to be programmed. The ispLSI 1000 family, ispLSI 2000 family, ispLSI 3000 family, and ispLSI 6000 family devices have an eight-bit device ID read during the Idle/ID State. The ispLSI 2000LV family devices have a 32-bit device ID code which can read by clocking in the

Table 3a. ispLSI Device ID Codes (8-Bit)

Device	MSB LSB
ispLSI 1016	00000001
ispLSI 1016E	00001011
ispLSI 1024	00000010
ispLSI 1024E	00001100
ispLSI 1032	00000011
ispLSI 1032E	00001101
ispLSI 1048	00000100
ispLSI 1048C	00000101
ispLSI 1048E	00001110
ispLSI 2032	00010101
ispLSI 2064	00010010
ispLSI 2096	00010011
ispLSI 2128	00010100
ispLSI 3160	00100100
ispLSI 3192	00100001
ispLSI 3256/A	00100010
ispLSI 3256E	00100011
ispLSI 6192	00110010

Device ID Codes Table

read-ID code instruction and clocking out the device ID code through the TAP controller.

Tables 3a and 3b list the device IDs for the ispLSI devices. The 32-bit device ID codes are read within the boundary-scan TAP controller while the 8-bit device IDs are read within the LSC ISP state machine. Notice that the 3000 family (except the 3256) and 6192 family may be programmed through either the LSC ISP state machine or the boundary-scan TAP controller but only support the LSC ISP 8-bit device ID.

The LSC ISP state machine instruction set is listed in Table 4. Instructions are loaded in the LSC ISP state machine Command Shift State and then executed in the Execute State. Notice that the device identification is read during the Idle/ID State, and this operation does not require an instruction.

The TAP controller programming instructions for Lattice's boundary-scan devices are listed in Table 5. Instructions are loaded in the Shift-IR state and executed in the Run-Test/Idle state by setting TMS low before a low-to-high transition on TCK and waiting the associated pulse width time before exiting the Run-Test/Idle state as shown in Figures 15 and 16.

While it is possible to erase the individual arrays of the device, it is recommended that the entire device be erased (UBE) and programmed in one operation. This Bulk Erase operation should precede every programming cycle as an initialization.

When a device is secured by programming the security cell (PRGMSC), the on-chip verify and load circuitry is disabled. The device should be secured as the last procedure, after all the device verifications have been completed. The only way to erase the security cell is to perform a bulk erase (UBE) on the device.

Table 3b. IEEE 1149.1 (JTAG) Device IDCODE (32-Bit)

Device	MSB LSB
ispLSI 2032V	00301043 (hex)
ispLSI 2064V	00306043 (hex)
ispLSI 2096V	00303043 (hex)
ispLSI 2128V	00308043 (hex)

Device ID Codes Table 2

Table 4. LSC ISP Programming State Machine Instruction Set

Instruction	Operation	Description
00000	NOP	No operation performed.
00001	ADDSHFT	Address Register Shift: Shifts address into the Address Shift Register from SDIN.
00010	DATASHFT	Data Register Shift: Shifts data into or out of the Data Serial Shift Register.
00011	UBE	User Bulk Erase: Erase the entire device.
10000	ERALL	Erase the entire device, including the UES (User Electronic Signature).
00100	GRPBE	Global Routing Pool Bulk Erase: Bulk erases the GRP array only.
00101	GLBBE	Generic Logic Block Bulk Erase: Bulk erases the GLB array only.
00110	ARCHBE	Architecture Bulk Erase: Bulk erases the architecture array and I/O configuration.
00111	PRGMH	Program High Order Bits: The data in the Data Shift Register is programmed into the addressed row's high order bits.
01000	PRGML	Program Low Order Bits: The data in the Data Shift Register is programmed into the addressed row"s low order bits.
01001	PRGMSC	Program Security Cell: Programs the security cell of the device.
01010	VER/LDH	Verify/Load High Order Bits: Load the data from the selected row's high order bits into the Data Shift Register for verification.
01011	VER/LDL	Verify/Load Low Order Bits: Load the data from the selected row's low order bits into the Data Shift Register for verification.
01110	FLOWTHRU	Flow Through: Bypasses all the internal shift registers and SDOUT becomes the same as SDIN.
10010	VE/LDH	Verify Erase/Load High Order Bits: Load the data from the selected row's high order bits into the Data Shift Register for erased verification.
10011	VE/LDL	Verify Erase/Load Low Order Bits: Load the data from the selected row's low order bits into the Data Shift Register for erased verification.
01111	PROGUES	Program UES.
10001	VERUES	Verify UES.

Table 5. TAP Controller Instruction Set

Inst.	Operation	Description
00001	ADDSHIFT	**Address Register Shift:** Shift Data into the Address Shift Register of the device.
00010	DATASHIFT	**Data Register Shift:** Clocks data into or out of the Data Shift Register.
00011	UBE	**User Bulk Erase:** Erases the entire device, excluding the USERCODE.
00111	PRGMHIGH	**Program High Order Data:** Data in the Shift Register is programmed into the addressed row, high order data only.
01000	PRGLOW	**Program Low Order Data:** Data in the Shift Register is programmed into the addressed row, low order data only.
01001	PRGMSC	**Program Security Cell:** Programs the Security Cell of the device.
01010	VER/LDHIGHP	**Verify/Load High Order Data:** Load data from the the selected row (high order data only) into the Serial Shift Register for Programmed Cell verification. Also used for the Load algorithm.
01011	VER/LDLOWP	**Verify/Load Low Order Data:** Load data from the selected row (low order data only) into the Serial Shift Register for Programmed Cell verification.
10000	ERALL	**Erase All:** Erase the entire device, including USERCODE.
10010	VER/LDHIGHE	**Verify/Load High Order Data:** Load data from the selected row (high order data only) into the Serial Shift Register for Erased Cell verification.
10011	VER/LDLOWE	**Verify/Load Low Order Data:** Load data from the selected row (low order data only) into the Serial Shift Register for Erased Cell verification.
10101	Prgm EN & PrgmDIS	**Program Enable and Program Disable:** The Program Enable and Program Disable instructions are used to enter and exit the programming mode. To enter the programming mode, three consecutive PrgmEN instructions must be loaded. To exit the programming mode, load the PrgmDIS instruction, followed by loading the BYPASS (11001) instruction and the device will exit the program mode.
10110	IDCODE	**Read ID Code:** This instruction loads the IDCODE into the 32-bit Data Shift Register. The entire 32-bit IDCODE should be verified before the device is programmed, verified or loaded. The IDCODE is loaded into the Data Shift Register during the Capture-DR state if the IDCODE instruction is in the Instruction Register. In addition, the IDCODE instruction is automatically in effect following power-up or a visit to the Test-Logic-Reset state. It stays in effect until the Capture-IR state is entered.
10111	USERCODE	**Read USERCODE or Verify USERCODE:** This instruction accesses the 32-bit USERCODE which is different from the 32-bit IDCODE used to recognize the device. The USERCODE provides 32 bits of data storage for the user and is available in both ispLSI and pLSI versions. In the normal mode, the 32-bit USERCODE is loaded into the 32-bit Shift Register on the rising edge of TCK in the Update-IR state as required in the IEEE 1149.1 specification. In the programming mode, the USERCODE is verified using this instruction. With the instruction loaded into the IR in the Run-Test/Idle state, the 32-bit USERCODE is loaded in the 32-bit Shift Register for cell verification.
11001	BYPASS	**Register BYPASS (default instruction):** The device is placed into bypass mode, where TDI is connected to TDO with one register. Following power-up or a visit to the Test-Logic-Reset state, this instruction is loaded into the Instruction Register during the Capture-IR state. IEEE 1149.1 requires that an instruction ending in 01 has to be in the Instruction Register following Test-Logic-Reset in order to detect Instruction Register length.
11010	Prgm USERCODE	**Program USERCODE:** Used to program the user 32-bit USERCODE.
11111	BYPASS	**Register BYPASS:** The device is placed into bypass mode where TDI is connected to TDO with one register.

Programming Voltage/Timing Specifications and Waveforms for the LSC ISP State Machine

For programming through the LSC ISP state machine, several timing specifications must be met. Table 6 describes a few of the critical timing parameters as they apply to programming sequences. Table 7 and Figures 12 and 13 show the voltage/timing specifictions for these parameters and others for the ispLSI 1000/E, 2000, 3000 and 6000 families of Lattice ISP devices.

Table 6. Timing Parameters for Programming Through the LSC ISP State Machine

Timing Parameter	Description
t_{ispEN}, t_{ispDIS}	Specifies the time it takes to get into the ISP mode after $\overline{\text{ispEN}}$ is activated. Or, the time it takes to come out from the ISP mode after $\overline{\text{ispEN}}$ becomes active.
t_{su}	Set-up time of the control signals before SCLK. Or, the set up time of input signals against other control signals (if applicable).
t_{h}	Hold time of the control signal after SCLK. It also applies to the same input signals from the set-up time.
t_{clkl}	Minimum clock pulse width, low.
t_{clkh}	Minimum clock pulse width, high.
t_{pwv}	Verify or read pulse width. The minimum time requirement from the rising clock edge of a verify/load instruction execution to the next rising clock edge (Figure 13).
t_{pwp}	Programming pulse width. The minimum time requirement from the rising clock edge of a programming instruction execution to the next rising clock edge (Figure 13).
t_{bew}	Bulk erase pulse width. The minimum time requrement from the rising clock edge of a bulk erase instruction execution to the next rising clock edge (Figure 13).
t_{rst}	Power on reset timing requirement. trst must elapse after power-up and before any operations are performed on the device.

Table 7. ISP Programming Voltage/Timing Specifications (ispLSI 1000/E, 2000, 3000 and 6000 Families)

SYMBOL	PARAMETER	CONDITION		MIN.	TYP.	MAX.	UNITS
V_{CCP}	Programming Voltage			4.75	5.0	5.25	V
I_{CCP}	Programming Supply Current			–	50	100	mA
V_{IHP}	Input Voltage High	$\overline{\text{ispEN}}$ = Low		2.0	–	V_{CCP}	V
V_{ILP}	Input Voltage Low			0.0	–	0.8	V
I_{IP}	Input Current			–	100	200	µA
V_{OHP}	Output Voltage High	I_{OH} = -3.2 mA		2.4	–	V_{CCP}	V
V_{OLP}	Output Voltage Low	I_{OL} = 5 mA		0.0	–	0.5	V
t_r, t_f	Input Rise and Fall			–	–	0.1	µs
t_{ispen}	$\overline{\text{ispEN}}$ to Output 3-State Enabled			–	–	10	µs
t_{ispdis}	$\overline{\text{ispEN}}$ to Output 3-State Disabled			–	–	10	µs
t_{su1}	Setup Time, isp State Machine			0.1	–	–	µs
t_{su2}	Setup Time, Program and Erase Cycle*			200	–	–	µs
t_{co}	Clock to Output			–	–	0.1	µs
t_h	Hold Time			0.1	–	–	µs
t_{clkh}, t_{clkl}	Clock Pulse Width, High and Low			0.5	–	–	µs
t_{pwv}	Verify Pulse Width			20	–	–	µs
t_{pwp}	Programming Pulse Width		1000	40	–	100	ms
			others	80	–	160	ms
t_{bew}	Bulk Erase Pulse Width			200	–	–	ms
t_{rst}	Reset Time from Valid V_{CCP}	Rise Time < 50 µs	1000/E, 2000	45	–	–	µs
			3000, 6000	100	–	–	µs

VT Specs/Table 2

Figure 12. Timing Waveforms for In-System Programming (ispLSI 1000/E, 2000, 3000 and 6000 Families)

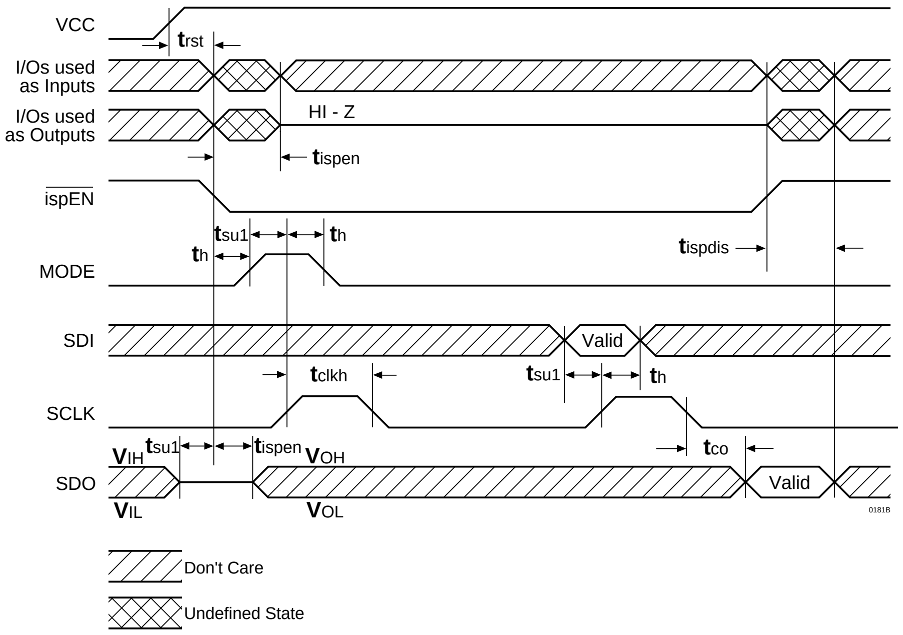
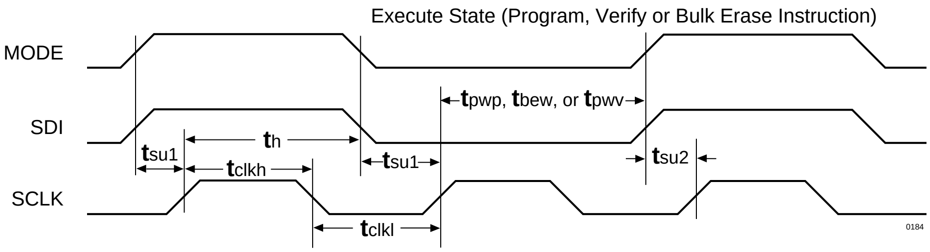


Figure 13. Program, Verify and Bulk Erase Waveforms (ispLSI 1000/E, 2000, 3000 and 6000 Families)



Programming Voltage/Timing Specifications and Waveforms for the Boundary Scan TAP Controller

A few of the critical timing parameters for programming through the boundary-scan TAP controller are described in Table 8 and the voltage/timing specifications for these parameters as well as others are shown in Table 9. Figures 14 and 17 show the timing waveforms for entering and exiting the programming mode in the TAP controller. Figures 15 and 16 show the timing waveforms for applying the programming pulse.

Table 8. Timing Parameters for Programming Through the Boundary-Scan TAP Controller

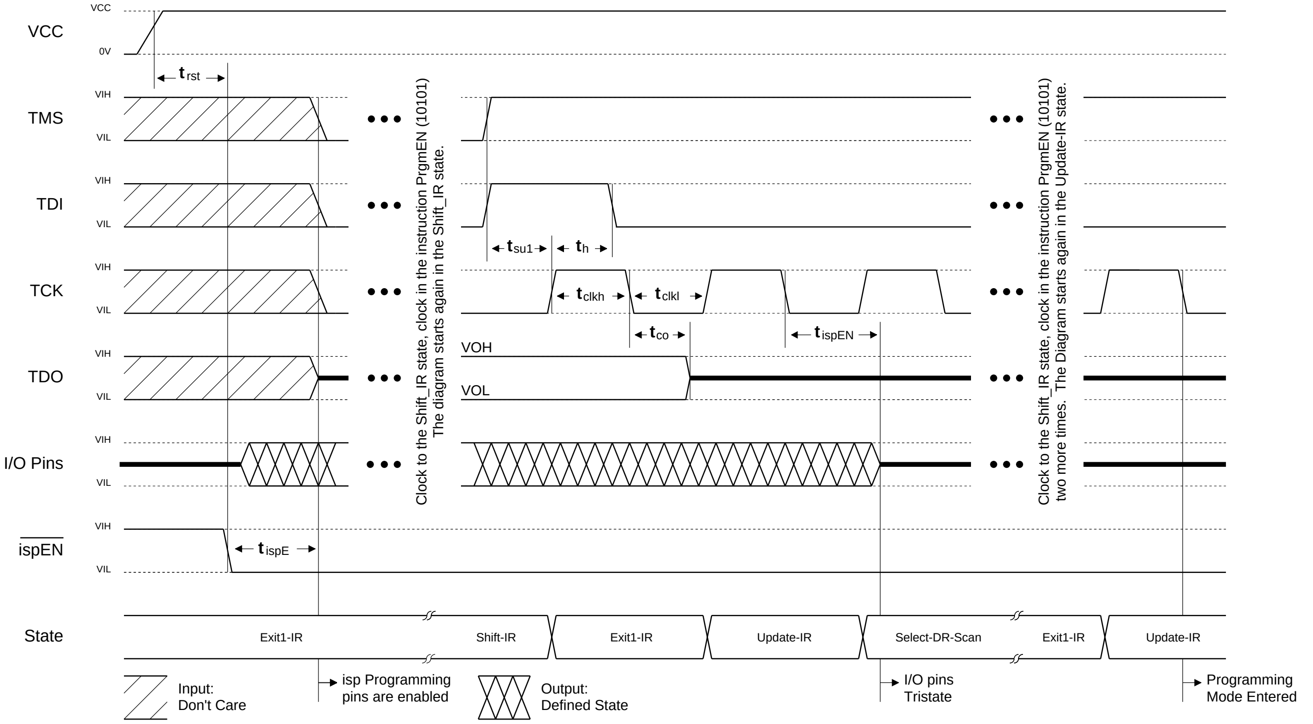
Timing Parameter	Description
t_{ispE}, t_{ispD}	Specifies the time it takes for programming pins to become active after the $\overline{\text{ispEN}}$ pin is asserted. Or, the time it takes for programming pins to deactivate after $\overline{\text{ispEN}}$ becomes inactive.
t_{ispEN}, t_{ispDIS}	Specifies the time it takes for the I/O pins to be put into the high-impedance state after the ProgEN instruction is loaded. Or, the time it takes for I/O pins to return to normal operation from the high-impedance state after exiting the programming mode.
t_{su}	Set-up time of the control signals before TCK.
t_h	Hold time of control signals after TCK.
t_{clkh}	Minimum clock pulse width, high.
t_{clkl}	Minimum clock pulse width, low.
t_{pwv}	Verify or read pulse width. The minimum time requirement from the rising edge of TCK in the Run-Test/Idle state, while executing a verify or read instruction to clocking to the next state.
t_{rst}	Power on reset timing requirement. trst must elapse after power-up and before any operations are performed on the device.
t_{pwp}	Programming pulse width. The minimum time requirement from the rising edge of TCK in the Run-Test/Idle state, while executing a programming instruction to clocking to the next state.
t_{bew}	Bulk erase pulse width. The minimum time requirement from the rising edge of TCK in the Run-Test/Idle state, while executing a bulk erase instruction to clocking to the next state.

Table 9. ISP Programming Voltage/Timing Specifications (ispLSI 2032LV)

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNITS
V_{CCP}	Programming Voltage		3.0	3.3	3.6	V
I_{CCP}	Programming Supply Current		–	50	100	mA
V_{IHP}	Input Voltage High	$\overline{\text{ispEN}}$ = Low	2.0	–	V_{CCP}	V
V_{ILP}	Input Voltage Low		0.0	–	0.8	V
I_{IP}	Input Current		–	100	200	µA
V_{OHP}	Output Voltage High	I_{OH} = -3.2 mA	2.4	–	V_{CCP}	V
V_{OLP}	Output Voltage Low	I_{OL} = 5 mA	0.0	–	0.5	V
t_r, t_f	Input Rise and Fall		–	–	0.1	µs
t_{dft}	TDI to TDO Delay with Flowthru Command		–	–	100	ns
t_{ispE}	$\overline{\text{ispEN}}$ to Programming Pins Enabled		–	–	1.0	µs
t_{ispD}	$\overline{\text{ispEN}}$ to Programming Pins Disabled		–	–	1.0	µs
t_{ispEN}	Program Enable Command to I/O 3-State Enabled		–	–	10	µs
t_{ispDIS}	Program Enable Command to I/O 3-State Disabled		–	–	10	µs
t_{su_1}	Clock Setup Time		100	–	–	ns
t_{su_2}	Program Setup Time		200	–	–	µs
t_{co}	Clock to Output		–	–	80	ns
t_h	Hold Time		10	–	–	ns
t_{clkh}, t_{clkl}	Clock Pulse Width, High and Low		100	–	–	ns
t_{pwv}	Verify Pulse Width		30	–	–	µs
t_{pwp}	Programming Pulse Width		80	–	–	ms
t_{bew}	Bulk Erase Pulse Width $_{CCP}$		200	–	–	ms
t_{rst}	Reset Time from Valid V		1	–	–	µs

Table 2 - 0029isp-2032

Figure 14. Timing Waveforms for Entering the Programming Mode (2000V Family)


VCC
VCC
0V
trst
TMS
TDI
TCK
TDO
I/O Pins
ispEN
VIH
VIL
State
Clock to the Shift_IR state, clock in the instruction PrgmEN (10101). The diagram starts again in the Shift_IR state.
tsu1
th
tclkh
tclkl
tco
tispEN
VOH
VOL
tispE
Clock to the Shift_IR state, clock in the instruction PrgmEN (10101) two more times. The Diagram starts again in the Update-IR state.
Exit1-IR
Shift-IR
Exit1-IR
Update-IR
Select-DR-Scan
Exit1-IR
Update-IR
Input: Don't Care
isp Programming pins are enabled
Output: Defined State
I/O pins Tristate
Programming Mode Entered

ENT_TIM.AI

Figure 15. Program and Bulk Erase Waveform (2000V Family)

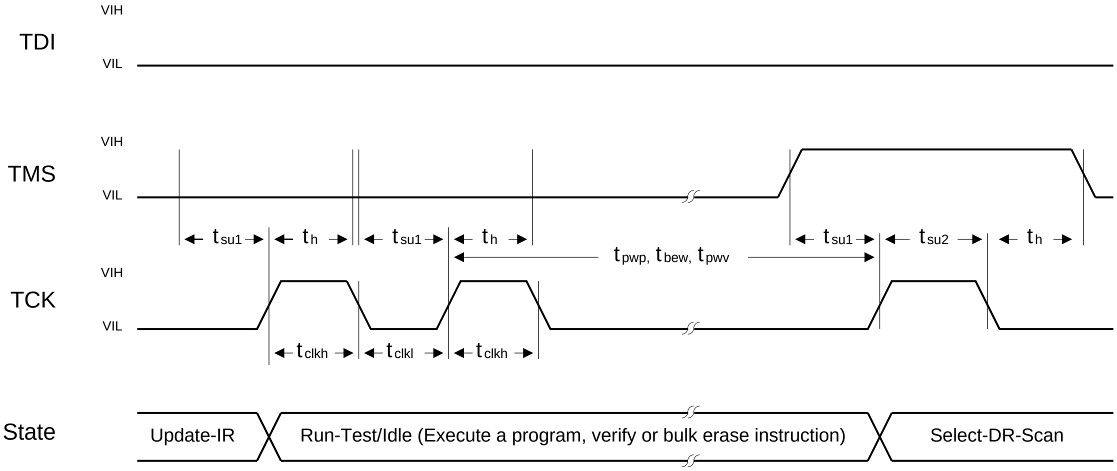


Figure 16. Verify Waveform (2000V Family)

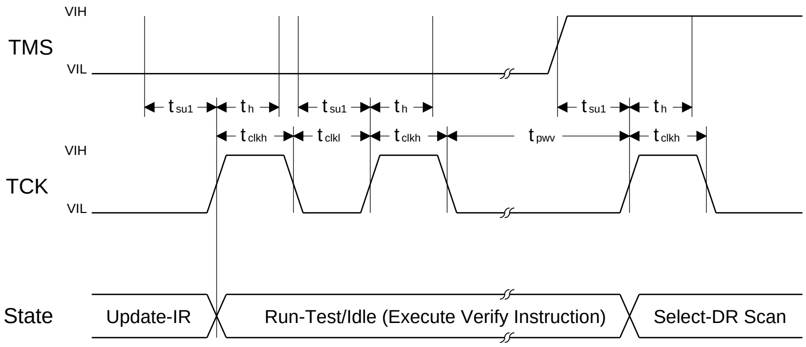
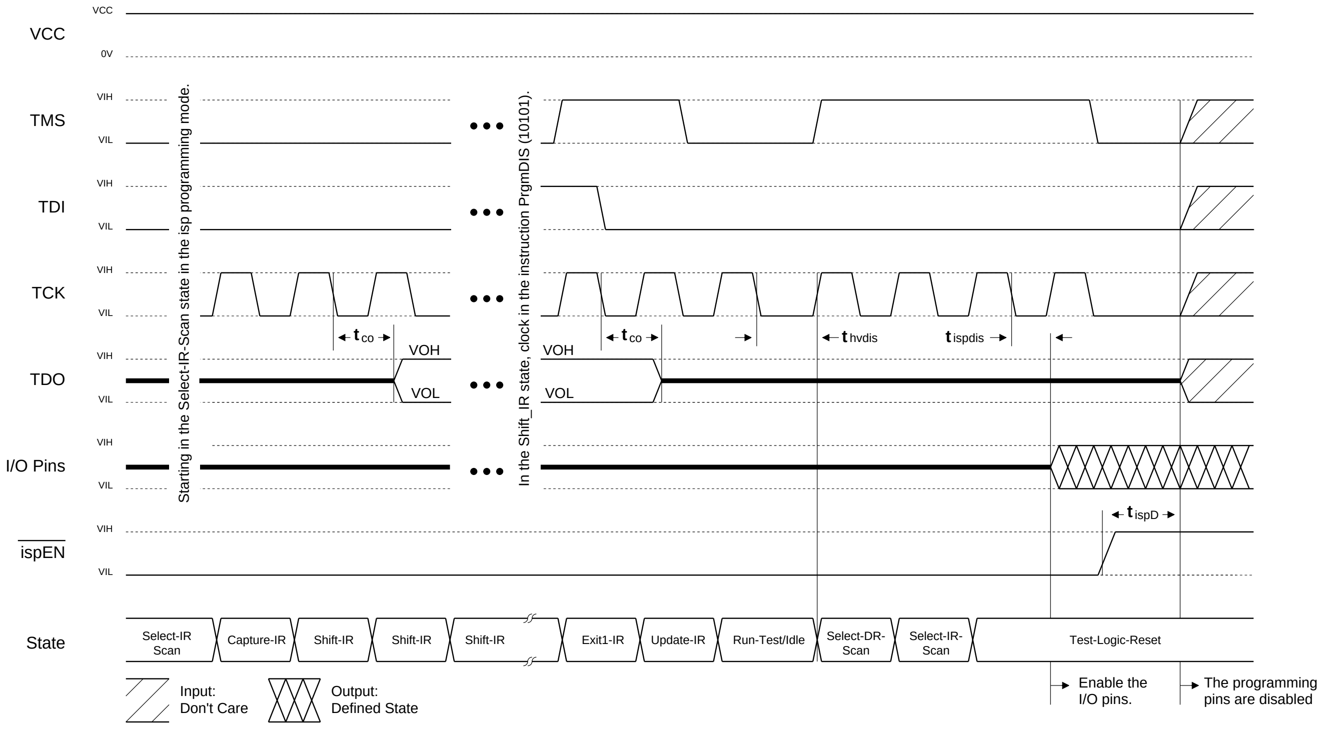


Figure 17. Timing Waveforms for Exiting the Programming Mode (2000V Family)



Device Layout

To translate the JEDEC format programming file into the serial data stream format for programming ispLSI devices, it is necessary to know the physical device layout and programming architecture. Two main factors determine how the translation must be implemented: the length of the address shift register and the length of the data shift register. The length of the address shift register indicates how many rows of data are to be programmed into the device. The length of the data shift register indicates how many bits are to be programmed in each row. Both registers operate on a First In First Out (FIFO) basis, where the Least Significant Bit (LSB) of the data or address is shifted in first and the Most Significant Bit (MSB) of the data or address is shifted in last. For the data shift register, the low order bits and the high order bits are separately shifted in.

Each ispLSI device has a predefined number of address rows and data bits needed to access its E^2CMOS cells during programming. The data bits span the columns of the E^2 array. From this information, the number of programming cells (or fuses) are determined. Table 10 highlights the address and data shift register (SR) sizes for currently available ispLSI devices. The JEDEC file for these ispLSI devices will reflect the number of cells (fuses) seen in Table 10. The total number of cells becomes critical if the programming patterns are to be stored in an on-board memory storage of limited capacity such as EPROM or PROM.

The L-fields in the JEDEC programming file indicate the link or fuse numbers of the device. The first cell of the device is indicated by cell number L00000. L-fields of subsequent lines are optional. From this reference cell location, all other cell locations are determined by relative position. A zero (0) in the cell location indicates that the E^2 cell in that particular location is programmed (or has a logic connection intact). A one (1) in the cell location indicates that the cell is erased (equivalent to an open connection). The logic compiler software automatically generates this JEDEC standard programming file after the design has been fit into the device.

Timing

When programming ispLSI devices, there are several critical timing parameters that must be met to ensure proper programming. The two most critical parameters are the programming pulse width (t_{pwp}) and the bulk erase pulse width (t_{bew}). These pulse widths determine the programming and erasing times of the E^2 cells. The preceding section detailed these critical program and erase timing specifications.

Fuse Map to Device Conversion

While the ISP Daisy Chain Download or ispCODE software takes care of this detail, it is important to understand how the JEDEC fuse map is mapped onto the physical ispLSI device during programming. The physical layout of the fuse pattern begins with Address Row 0 and ends

Table 10. ispLSI Address, Data Shift Register and Total Cell Summary

Device	Address SR Length	Data SR Length/Address	Total Number of Cells
ispLSI 1016/E	96/110	160/160	15,360/17,600
ispLSI 1024/E	102/122	240/240	24,480/29,280
ispLSI 1032/E	108/134	320/320	34,560/42,880
ispLSI 1048/C/E	120/155/158	480/480/480	57,600/74,400/75,840
ispLSI 2032/V	102	80	8,160
ispLSI 2064/V (84/100-Pin)	118	160	18,880
ispLSI 2064V (44-Pin)	110	80	17,600
ispLSI 2096/V	134	240	32,160
ispLSI 2128/V (160/176-Pin)	150	320	48,000
ispLSI 2128V (84/100-Pin)	134	160	42,880
ispLSI 3160	200	400	80,000
ispLSI 3192	216	480	103,680
ispLSI 3256/A	180	676	121,680
ispLSI 3256E	248	640	158,720
ispLSI 6192	180	600	108,000

ispLSI Address Table

Table 11. Summary of ispLSI Data Shift Register Bits

Device	High Order Data SR LSB	High Order Data SR MSB	Low Order Data SR LSB	Low Order Data SR MSB	Data SR Size (Bits)
ispLSI 1016/E	0	79	80	159	160
ispLSI 1024/E	0	119	120	239	240
ispLSI 1032/E	0	159	160	319	320
ispLSI 1048/C/E	0	239	240	479	480
ispLSI 2032/V	0	39	40	79	80
ispLSI 2064/V	0	79	80	159	160
ispLSI 2096/V	0	119	120	239	240
ispLSI 2128/V	0	159	160	319	320
ispLSI 3160	0	399	–	–	400
ispLSI 3192	0	239	240	479	480
ispLSI 3256/A	0	337	338	675	676
ispLSI 3256E	0	639	–	–	640
ispLSI 6192	0	299	300	599	600

ispLSI DSR Bits Table

with the maximum Address Row N and is determined by the length of the Address SR as described in Table 10. Spanning the Address Rows are the outputs of the High-Order Data SR and Low-Order Data SR, as described in Table 11. Programming fuses on a given row are enabled by a “1” within the Address Shift Register for the appropriate row and the use of state machine instructions that selectively operate on the High-Order Data SR or the Low-Order Data SR. For example, the PRGMH instruction in the LSC ISP state machine instruction set programs the High-Order data bits within the device for the selected Address Row and the PRGML instruction programs the Low-Order data bits (Table 4 lists the LSC ISP state machine instructions). The starting cell (L00000) of the JEDEC fuse map shifts into the device at the physical location corresponding to Address Row 0, High-Order Data SR bit 0 (Figure 30). The “n” and “m” in the figure refer to the Address SR length and the Data SR length respectively, of the device. Table 10 lists the size of the Address and Data shift registers for all ispLSI devices. A series of sequential shifts eventually results in the last cell location (Total # of Cells - 1) of the JEDEC fuse map shifting into Address Row (n-1), Low-Order Data SR bit (m-1) on the actual device.

The ispCODE Software routines make use of a bit packed data format, called ispSTREAM™, to transfer data between the JEDEC fuse map and the physical device locations. The binary ispSTREAM format uses one bit to represent the state of each of the programmable cells, instead of the byte value used in an ASCII JEDEC file. Considering the additional characters present in a JEDEC file, this adds up to a space savings of more than a factor of eight. In addition, the ispSTREAM does not require any parsing; the bits are simply read from the file and shifted into the device. As only 1922 bytes are required to store the pattern for an ispLSI 1016 device, multiple patterns can be stored in a small amount of memory. The JEDEC fuse map can be translated into ispSTREAM format using the dld2isp.exe program.

Algorithms

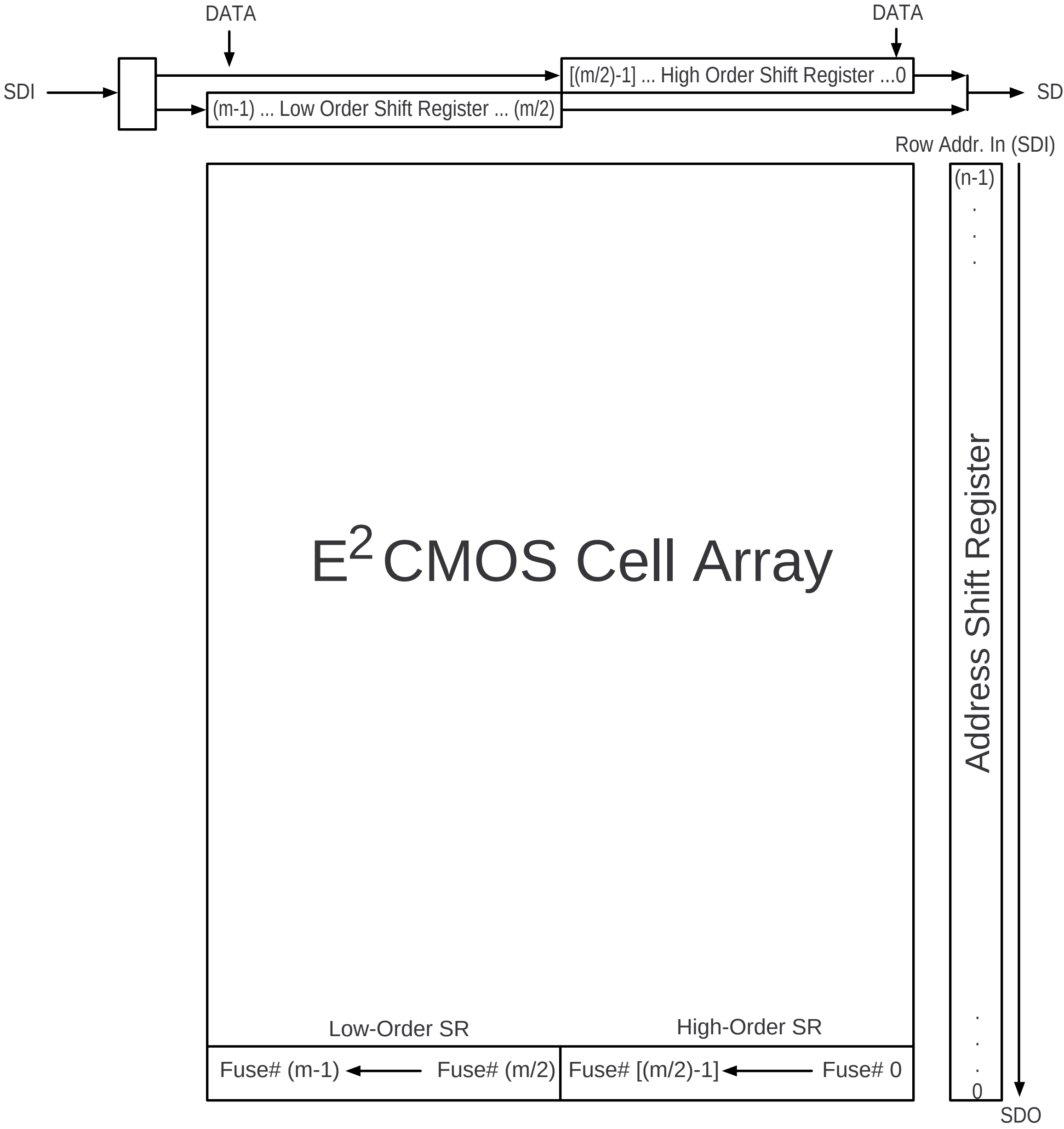
Command Stream

The first step in programming is to determine the device type to be programmed. At this point it is important to be aware of the hardware configuration that is being used. Serial daisy chain and parallel programming configurations are listed earlier in this section.

For simple serial daisy chains with all devices programmable with the LSC ISP state machine the device IDs can be read once the ISP programming mode is enabled ($\overline{\text{ispEN}}$ low) by keeping SDI to a known level (either high or low), the ID shift can be terminated when a sequence of eight ones or eight zeros is read. From the device IDs received the serial bit stream for programming can be arranged.

For software that must recognize both LSC ISP state machine programmable and TAP controller programmable ispLSI devices, the algorithm must start by determining which type of devices are present. After enabling the programming mode ($\overline{\text{ispEN}}$ low) the software checks for the presence of LSC ISP devices by setting MODE/TMS and SDI/TDI low. If LSC ISP devices are present the SDO/TDO signal will eventually show eight 0’s in a row and the intervening bits are read as the

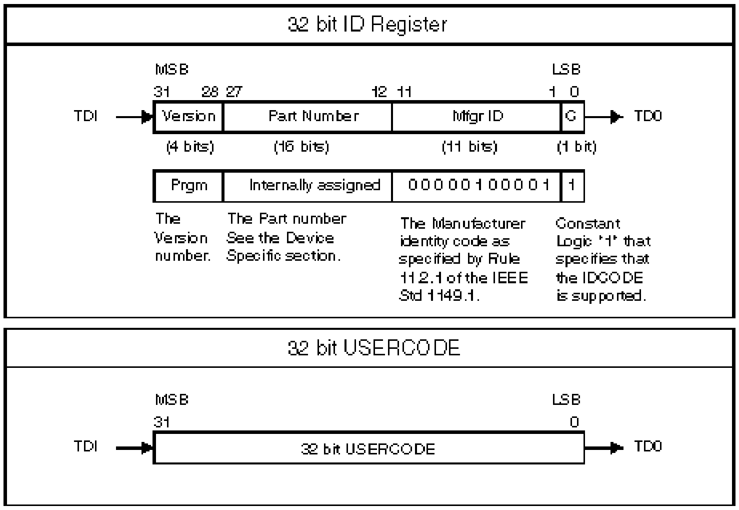
Figure 18. ispLSI Device to Fuse Map Translation



device IDs. To check for the presence of boundary-scan TAP ISP devices the $\overline{\text{ispEN}}$ signal is driven high and the devices are clocked to the Shift-IR state and the IDCODE instruction loaded. After moving to the Shift-DR state subsequent clocking will shift out the 32-bit device IDs for the boundary-scan devices. Figure 19 shows the 32-bit ID register present in all TAP controller programmable ispLSI devices.

The devices present are identified by the device IDs read by the programming software. The correct programming sequence can then be followed for the device. Below are typical sequences for programming LSC ISP state machine ispLSI devices and ispLSI devices programmable through the TAP programming interface.

Figure 19. BSCAN 32-Bit ID Registers and 32-Bit USERCODE



Typical LSC ISP Programming Sequence

1) Power up
2) Read ID code
3) ADDSHFT command shift
4) Execute ADDSHFT command
5) Shift address
6) DATASHFT command shift
7) Execute DATASHFT command
8) Shift high order data
9) PRGMH command shift
10) Execute PRGMH
11) DATASHFT command shift
12) Execute DATASHFT command
13) Shift low order data
14) PRGML command shift
15) Execute PRGML
16) Repeat from 1) until all rows are programmed
17) Program UES
18) Optional: Program security bit
19) Power down

Typical TAP programming sequence

1) **Power Up the device**

2) **Verify the 32 bit IDCODE:** Clock-in the instruction Read IDCODE (10110), Clock-out and verify the 32 bit IDCODE, then Clock back to the Shift-IR state.

3) **Enter the Programming Mode:** Load and execute the Program Enable (PrgmEN) (10101) instruction three times. Upon loading the PrgmEN instruction and clocking to the Update-IR state, the third time the device enters the programming mode. Then Clock to the Shift-IR state.

4) **Initialize the Address (Load all Zeros):** Clock-in the instruction Address Reg. Shift (00001); use the correct setup and hold times. Then Clock back to the Shift-IR state.

5) **Bulk Erase the device:** Clock-in the Bulk Erase All (10000) instruction. Execute the Erase Command, using T_{BEW} and T_{SU2}. Then Clock back to the Shift-IR state.

6) **Initialize the Address (select the last row):** Clock-in the instruction Address Shift (00001) and then Clock back to the Shift-IR state.

7) **PROGRAM LOOP:** Program High Order and Low Order data, Increment Address and Check Address.

8) **VERIFY LOOP:** Initialize the Address, Verify High and Low Order data, using Lvt & Hvt instructions.

9) **PROGRAM USERCODE data:** Using instruction Program USERCODE (11010). If JEDEC file contains USERCODE data (U-field binary data only) clock-in USERCODE data. If there is no U-Field data clock in all 0s.

10) **Verify USERCODE data:** Using the instruction Verify USERCODE (10111).

11) **OPTIONAL: Program the security cell:** Using the instruction PRGMSC (01001). Use T_{PWP}, and T_{SU2}, then finish clocking back to the Shift-IR state.

12) **Power Down.**

Note: The number of clock pulses required to read data is one less than the length of the shift register. This is because the first bit is always available on TDO. Correct setup and hold times must be implemented.

Boundary Scan (ispLSI 3000 & 6000 Families)

The Lattice Semiconductor ispLSI 3000 and 6000 families of devices support the IEEE 1149.1 boundary-scan specifications. The following sections explain in detail how to interface to the devices through the Test Access Port (TAP), how the boundary scan registers are implemented within the devices, and the boundary-scan instructions that are supported by the ispLSI and pLSI 3000 and 6000 families.

Test Access Port (TAP)

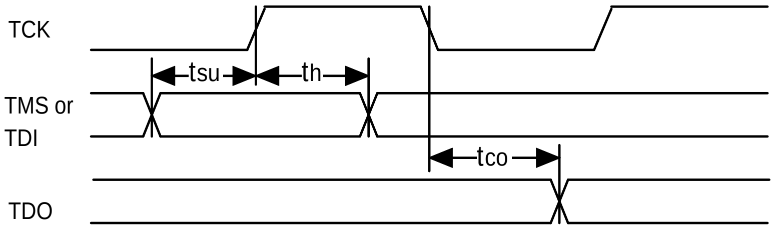
The boundary-scan test access port is accessed through six interface signals: TDI, TDO, TCK, BSCAN, TMS, $\overline{\text{TRST}}$. These interface signals have two functions in the case of the ispLSI 3000 and 6000 families; they serve as both the boundary-scan interface and in-system programming interface signals. For the pLSI 3000 and 6000 families, the six interface signals are only used for the boundary scan TAP interface. Table 12 describes the interface signals.

The abovementioned six signals are dedicated for boundary-scan use for the pLSI devices. As ISP programming is accomplished through the same pins, five of the six signals have both boundary-scan interface and ISP functions on the ispLSI devices. $\overline{\text{TRST}}$ is the only signal that does not have a dual function. It is used only to reset the TAP controller state machine. The sequencing of test routines are governed by the TAP controller state machine. The state machine uses the TMS and TCK signals as its inputs to sequence the states. Figure 5 is the

Table 12. ispLSI and pLSI 3000 and 6000 Family Boundary-Scan Interface Signals

pLSI	ispLSI	Pin Function Description
BSCAN	BSCAN/$\overline{\text{ispEN}}$	Active high signal on this pin selects the Boundary-Scan function while active low signal selects the ISP function on the ispLSI devices. Internal pullup on this pin drives the signal high when the external pin is not driven.
TCK	TCK/SCLK	Test Clock function for Boundary-Scan and Serial Clock for the the ISP function.
TMS	TMS/MODE	Test Mode Select for Boundary-Scan and MODE control for the ISP function.
TDI	TDI/SDI	Test Data Input for Boundary-Scan and Serial Data Input for the ISP function. Functions as a serial data input pin for both interfaces.
$\overline{\text{TRST}}$	$\overline{\text{TRST}}$	Test Reset Input is an asynchronous signal to initialize the TAP controller to the Test-Logic-Reset state.
TDO	TDO/SDO	Test Data Output for Boundary-Scan and Serial Data Output for the ISP function. Functions as a serial data output pin for both interfaces.

Figure 20. TAP Controller Timing Diagram



IEEE1149.1 specified state machine. The condition for the state transition is the state of the TMS input condition before TCK within a given state. The timing diagram is shown in Figure 20.

The TAP controller state machine includes the Test-Logic-Reset state to reset the controller and the Run-Test states. Two main components of the TAP controller are Data Register (DR) control states and Instruction Register (IR) control states. Both of these register control states are organized in a similar manner. The user can capture the registers, shift the register string, or update the registers. Capturing the DRs simply loads the DR with the data from the corresponding functional input, output, or I/O pins. The IR capture, on the other hand, loads the IRs with the previously executed instruction bits. Shift register states serially shift the DR and IR. In the case of DR shift, the data is shifted according to the order of the inputs, outputs, and I/Os defined in the boundary-scan section of each device data sheet. The IRs are shifted out from the least significant bit first. During update register states, the DRs update the latches to drive the external pins and the IRs update the instruction bits with the instruction that is to be executed.

Boundary-Scan Registers

In order to support boundary-scan test, three types of data registers are defined for the ispLSI and pLSI devices — I/O cell registers, input cell registers and output cell registers (6000 family only). The main purpose of these registers is to capture test data from the appropriate signals and shift data to either drive the test pins or examine captured test data.

Figure 21. Boundary Scan I/O Cell

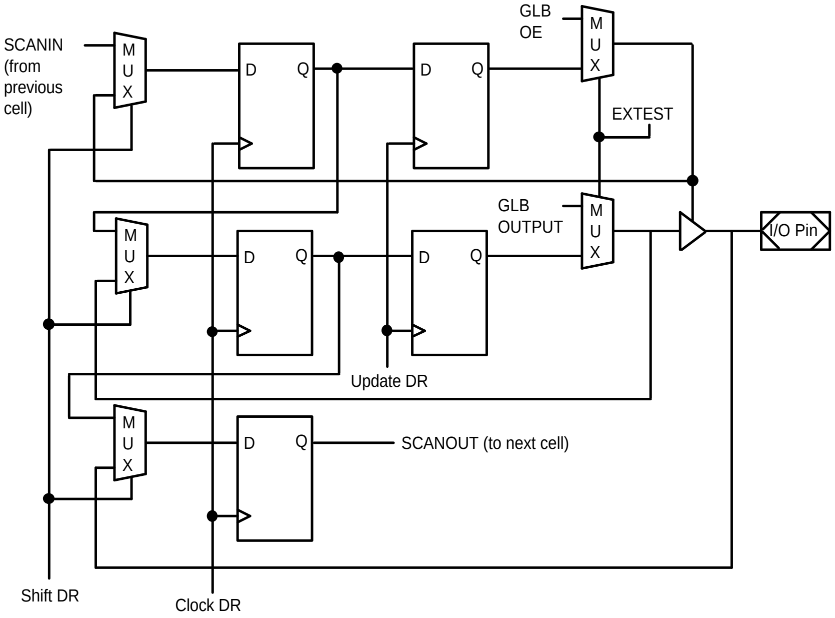


Figure 22. Boundary Scan Input Cell

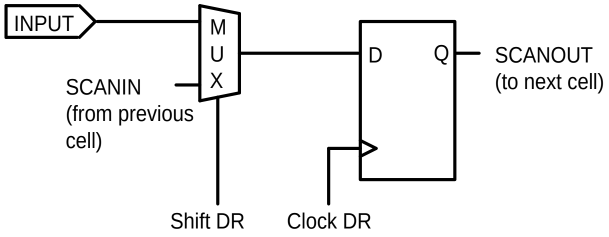


Figure 21 describes the register for the I/O cell. The I/O cell, by definition, must have three components: one register component drives the output enable (OE) signal, the second component drives the output data, and the third captures the input data. These components make up the three registers that are part of the shift register string for each of the I/O pins. Only parts of the I/O cell registers will have valid data when I/O pins are configured as input-only or output-only, thus the test routines must be able to monitor the appropriate register bits. The update registers are used mainly to store data that is to be driven onto the I/O pins. The multiplexer controls are driven by the signal from the TAP controller at appropriate states.

The function of an input cell register is simpler than that of an I/O cell. Figure 22 illustrates the single input register cell. The purpose of the input cell is to capture the input test data and shift the data out TDO for verification.

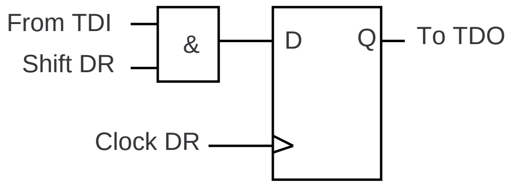
Boundary-Scan Instructions

Lattice ispLSI and pLSI devices support the three mandatory instructions defined by the boundary-scan definition. The following paragraphs describe each of the instructions and its instruction code. A shift register of five bits for the ispLSI and pLSI 3192 and ispLSI and pLSI 6192 and two bits for the ispLSI and pLSI 3256 is defined within the devices to implement the instruction shift register.

The SAMPLE/PRELOAD (Instruction Code - 10*/11100) instruction is used to sample the pins that are to be tested. During the Capture-DR state, while executing this instruction, the DRs are loaded with the state of the pins which can then be examined after shifting the data through TDO. The PRELOAD part of this instruction is simply loading the DRs during Shift-DR state with the desired condition for each of the pins.

The EXTEST (Instruction Code - 00*/00000) instruction drives the external pins with the previously updated values from the DR during the Update-DR state.

Figure 23. Bypass Register



The BYPASS (Instruction Code - 11*/11111) instruction is used to bypass any device that is not accessed during any part of the test. The definition of the BYPASS instruction allows TDI not to be driven during the Shift-IR state. In order to shift in the correct instruction code, the TDI pin has an internal pull-up to drive logic high. A bypassed boundary-scan device has a single bypass register as shown in Figure 23.

*3256 only.

ispGDS Programming Details

The following sections describe the state machine instruction set, timing parameters, device layout, and programming algorithms as they apply to ispGDS devices in general. Figure 24 shows the ispGDS22 28-pin device pinout.

Shift Registers

The ispGDS devices have three shift registers, the device ID shift register, the Instruction shift register and the Data shift register. All shift registers operate on a First In First Out (FIFO) basis, and are chosen by which state the programming state machine is in.

Figure 24. ispGDS22 28-Pin PLCC Pinout Diagram

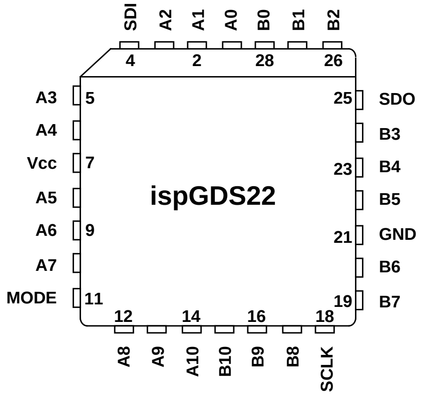


Table 13. ispGDS Device Codes

Device	Pins	Device ID
ispGDS22	28	0111 0010 (72 hex)
ispGDS18	24	0111 0001 (71 hex)
ispGDS14	20	0111 0000 (70 hex)

ispGDS ID Codes

The device ID shift register is only accessible in the IDLE state. It is eight bits long, and is only used to shift out the device ID. The ispGDS device IDs are 70-72 (hex) (Table 13). The Instruction shift register is only accessible in the SHIFT state. It is five bits long, and is only used to shift the Instruction Codes into the device. The Device ID and Instruction shift registers expect the LSB to be shifted in first. The Data shift register is 24 bits long, and is used to shift all addresses and data into or out of the device. The Data shift register is only accessible in the EXECUTE state when executing a SHIFT_DATA instruction (Table 14).

To program an ispGDS device, data is read from a serial bit stream and shifted into the shift registers. Twenty-four bits are read at a time, shifted into the device, and then a programming operation is performed. The exact sequence, and the methods for converting a JEDEC map into a serial bit stream are explained in the ispGDS Internal Architecture section.

Timing

Programming the ispGDS devices properly requires that a number of timing specifications be met. The specifications relating to programming and erasing the E^2CMOS cells are the most critical. In addition to a minimum pulse width, there is also a maximum timing specification. Refer to the ispGDS programming mode timing specifications in Table 15 for the timing requirements. Timing diagrams for the programming mode specifications are shown in Figures 25, 26 and 27.

ispGDS Internal Architecture

This section covers the details of constructing the ispSTREAM format. Only 49 bytes are required to store the pattern for an ispGDS device. If you are using the supplied software tools, a conversion utility (complete with source code) is included to convert an industry-standard JEDEC file to ispSTREAM format. All of the Lattice software routines read and write this ispSTREAM.

The ispGDS devices are composed of two basic architectural components (Figure 28). The first component consists of three rows of architectural information, which contain the three bits that control the function of each I/O cell. The rows are 24 bits long, providing one bit for each I/O cell (the ispGDS18 and ispGDS14 do not use all of the bits). The second component contains the cell data for the switch matrix area of the device and the User Electronic Signature (UES) data area. There are two UES rows of 24 bits each, and 11 switch matrix rows of 24 bits each.

Although the shift register lengths are 24 bits long, it is not composed entirely of data area. In the architectural section, two bits are used for addressing. In the matrix/UES area, six bits are used for addressing. In the switch matrix area, there are only 11 bits of actual data, and seven dummy bits which exist only to make the shift

Table 14. ispGDS Programming State Machine Instruction Set

Instruction	Operation	Description
00000	NOP	No operation performed.
00010	SHIFT_DATA	Clocks data into, or out of, the Data Shift Register.
00011	BULK_ERASE	Erases the entire device.
00101	ERASE_ARRAY	Erases everything except the Architecture rows.
00110	ERASE_ARCH	Erases the Architecture rows only.
00111	PROGRAM	Programs the Shift Register data into the addressed row.
01010	VERIFY	Load data from the selected row into the Serial Shift Register.
01110	FLOWTHRU	Disables the Shift Register (SDI=SDO).

Table 15. Programming Mode Timing Specifications (ispGAL22V10 and ispGDS Families)

Param.	Description		Min.	Max.	Unit
t_{rst}	Time from power-up of device to any progamming operation.		1	—	µs
t_{isp}	Time from leaving IDLE state to I/O pins tri-state, or entering IDLE state to I/O pins active.		—	10	µs
t_{su}	Setup time, from either MODE or SDI to rising edge of SCLK.		100	—	ns
t_{h}	Hold time, from rising edge of SCLK to MODE or SDI changing level.		100	—	ns
t_{co}	Time from falling edge of SCLK to data out on SDO.	ispGAL22V10	—	210	ns
		ispGDS	—	150	ns
t_{clkh}	Clock pulse width of SCLK while high.		0.5	—	µs
t_{clkl}	Clock pulse width of SCLK while low.		0.5	—	µs
t_{pwp}	Time for a programming operation.		40	100	ms
t_{pwe}	Time for an erase operation.		200	—	ms
t_{pwv}	Time for a verify operation.		5	—	µs

registers the same length. These seven bits are read as a one, or a logic High on SDO. For the UES, there are 16 bits of actual data in each row and two dummy bits.

ispGDS ispSTREAM Format

To convert the information in a standard JEDEC file into the ispSTREAM format, add all of the addressing information and the placeholding bits (dummy bits). The objective is to include every bit needed for programming. For the three architecture rows, simply add the two address bits.

For the UES and Switch Matrix rows, there are eight bits to add. The first two bits are always 00, which distinguishes this area from the Architectural row. In addition, there are four bits needed to address the specific row, and two bits needed as placeholders. In the Switch Matrix rows, there are also five bits needed for placeholding at the end of the rows. The various placeholding bits are built into the device so that all rows appear to be the same length, thus simplifying programming operations.

The ispSTREAM uses one bit for each programmable cell. This means that each row includes 24 bits, or three *bytes* of storage. With three bytes of storage per row, and 16 rows per device, the ispSTREAM uses only 48 bytes of storage area. However, there is one extra byte used at the front of the file to store the device ID code, and a 32-bit checksum. The ID code is identical to the one that is hardwired into the device. This ID code ensures that the ispSTREAM type is the same as the device to be programmed. For example, if an ispSTREAM is stored in EPROM, it is stacked end to end. The ID code determines not only which device type the ispSTREAM belongs with, but its length, and thus, where the next pattern starts. All

Figure 25. Programming Mode Timing (ispGDS and ispGAL22V10 Families)

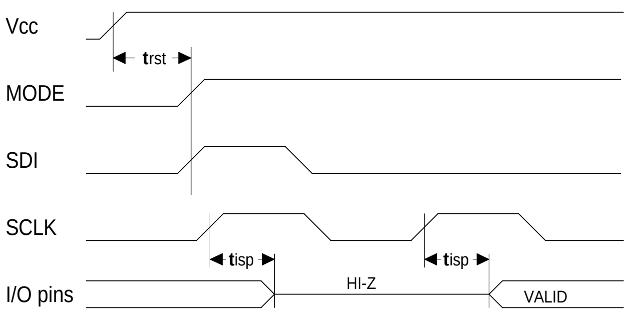


Figure 26. Shift Register Timing (ispGDS and ispGAL22V10 Families)

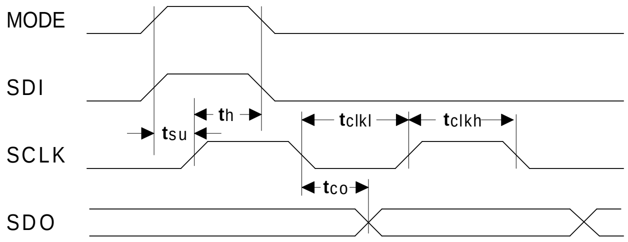
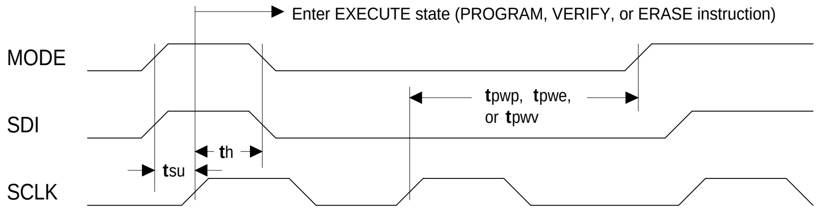


Figure 27. Program, Verify, and Erase Timing (ispGDS and ispGAL22V10 Families)

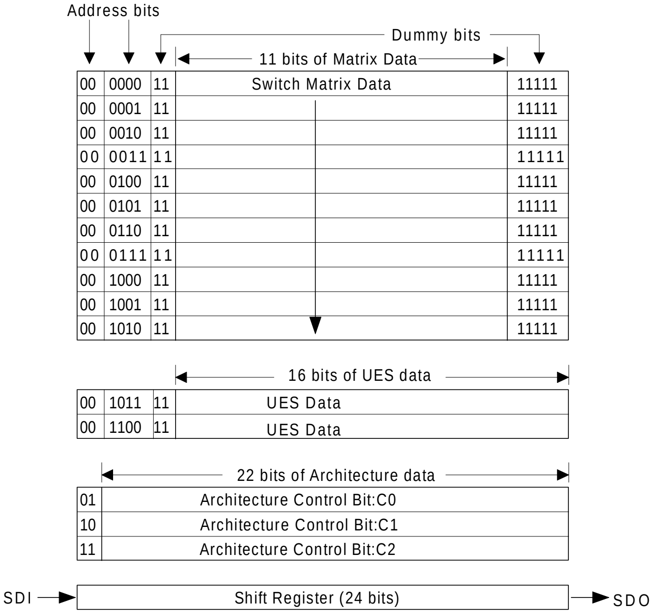

Enter EXECUTE state (PROGRAM, VERIFY, or ERASE instruction)
MODE
SDI
SCLK
t_{pwp}, t_{pwe}, or t_{pwv}
t_h
t_{su}

ispSTREAM formats, regardless of which Lattice In-System Programmable device they are intended for, contain this ID code in the first byte. See Figure 29 for details of the ispSTREAM format, and Figure 30 for the JEDEC map.

Algorithms

ispGDS device programming is described as a hierarchical set of algorithms and functions. This section contains high-level algorithms for erasing, programming, verifying, and loading ispGDS devices. A universal set of functions is used to make up the algorithms and enable them to be written in a modular format. The individual functions are explained in the next section. Note that most procedures leave the device in the SHIFT state. These algorithms and functions closely follow the ispCODE source code library provided by Lattice.

Figure 28. ispGDS Architecture



00	0000	11	Switch Matrix Data	11111
00	0001	11		11111
00	0010	11		11111
00	0011	11		11111
00	0100	11		11111
00	0101	11		11111
00	0110	11		11111
00	0111	11		11111
00	1000	11		11111
00	1001	11		11111
00	1010	11		11111

00	1011	11	UES Data
00	1100	11	UES Data

01	Architecture Control Bit:C0
10	Architecture Control Bit:C1
11	Architecture Control Bit:C2

Shift Register (24 bits)

To simplify the algorithms, all operations use an ispSTREAM format as the data structure from which to read and write. The ispSTREAM contains the address information and simplifies the operations considerably. Working from the ispSTREAM, the device appears as an array of 16 rows, each 24 bits long.

Program Algorithm

Before programming a device, it must be erased. Cells can be programmed (set to a JEDEC zero) using the programming command, but only an Erase procedure erases a cell (set a cell back to a JEDEC “1” (one)). In the algorithm in Listing 1, the entire device is erased (Bulk Erased), and then the entire device is programmed.

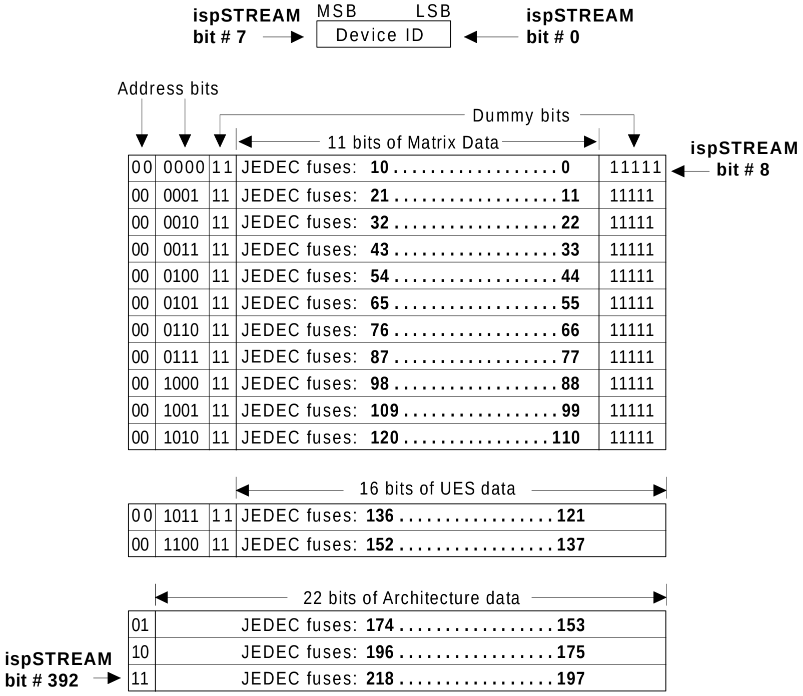
Load Algorithm

The load algorithm in Listing 2 is the same for all ispGDS devices. First, the 13 rows of array data (11 rows for the array matrix, and two for the UES) are read, and then the three rows of architectural information are read. After each row is read, it is stored in an ispSTREAM format.

In order to load each row’s data into the shift register, it is necessary to load the address of the row into the appropriate area of the shift register. Because of the unique way the different areas of the device are addressed, the simplest way to get the addresses into the device in the proper order is to use an existing ispSTREAM to supply those addresses. In other words, the full data for each row is loaded from the ispSTREAM into the device. When a VERIFY command is executed, the device’s data for the same row is then loaded into the shift register to be shifted out. This method will be used in this algorithm.

When using an existing ispSTREAM to supply the addresses, the data should not be the same as the expected data, or a failure to verify may not be detected. To avoid this possibility, a pattern that contains all “1s” (ones) for data can be used (and is supplied with the software tools provided by Lattice Semiconductor). This ispSTREAM still has the addresses intact, but all programmable cell data is set to a “1” (one) (erased state).

Figure 29. ispGDS ispSTREAM Format

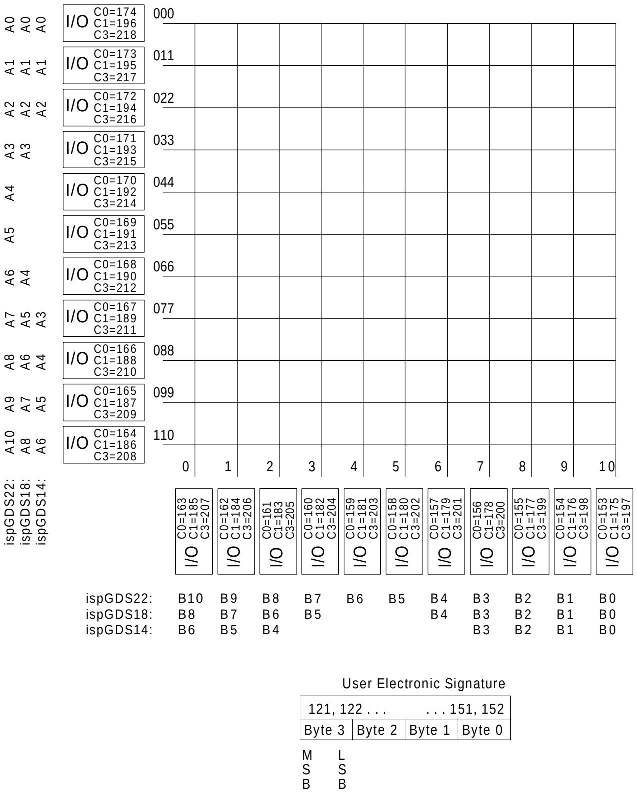


00	0000	11	JEDEC fuses: **10** **0**	11111
00	0001	11	JEDEC fuses: **21** **11**	11111
00	0010	11	JEDEC fuses: **32** **22**	11111
00	0011	11	JEDEC fuses: **43** **33**	11111
00	0100	11	JEDEC fuses: **54** **44**	11111
00	0101	11	JEDEC fuses: **65** **55**	11111
00	0110	11	JEDEC fuses: **76** **66**	11111
00	0111	11	JEDEC fuses: **87** **77**	11111
00	1000	11	JEDEC fuses: **98** **88**	11111
00	1001	11	JEDEC fuses: **109** **99**	11111
00	1010	11	JEDEC fuses: **120** **110**	11111

00	1011	11	JEDEC fuses: **136** **121**
00	1100	11	JEDEC fuses: **152** **137**

01	JEDEC fuses: **174** **153**
10	JEDEC fuses: **196** **175**
11	JEDEC fuses: **218** **197**

Figure 30. ispGDS JEDEC Fuse Map



Listing 1. ispGDS Programming Algorithm

To program a device:

```
Call procedure: Get_ID ( to check device type)
Call procedure: Change_State ( from IDLE to SHIFT state)
(Erase entire device)
      Call procedure: Shift_Command, with command: ERASE
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Execute_Command (starts operation)
      Call procedure: Wait (Erase_Time)
      Call procedure: Change_State ( to SHIFT state)
Set row_count =0
Loop until row_count = 15
(Program one row on each loop)
      Call procedure: Shift_Command, with command: SHIFT_DATA
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Shift_Data_In, with data location in ispSTREAM at (row_count
      x 24)
      Call procedure: Change_State ( to SHIFT state)
      Call procedure: Shift_Command, with command: PROGRAM
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Execute_Command (starts operation)
      Call procedure: Wait (Program_Time)
      Call procedure: Change_State ( to SHIFT state)
End Loop
```

Verify Algorithm

A row by row verification procedure is used to verify the ispGDS device. This procedure is basically the same as the Load algorithm, except that each row is compared with (instead of stored in) an ispSTREAM as the data is shifted out of the device. Note that the special pattern used for verifying is used to load the addresses, as in the Load algorithm.

ispGDS Procedures

This section describes the procedures that make up the program, verify, and load algorithms for the ispGDS family of devices. The procedures are written so that each algorithm may be written in a high-level modular format, calling one of the following procedures to actually change pin levels and handle timing.

Important: Notice that most of the procedures are written so that the state machine is left in the Shift State, ready to perform the next operation. This point is important in keeping all the routines compatible.

Goto_IDLE Procedure

The Goto_IDLE procedure resets the programming state machine to the IDLE state, regardless of which state it is in. Procedure steps:

set MODE pin High, and SDI pin Low

wait Tsu

bring SCLK pin High

wait Tclkh

bring SCLK pin Low

(END Procedure)

Get_ID Procedure

The 8-bit device ID codes identify the three different ispGDS devices (Table 13). The ID is read in the IDLE state by first loading the ID into the shift register and then clocking the data out. The ID is loaded by holding MODE high and SDI low and clocking the device. The ID is clocked out of the device by holding MODE low and clocking SCLK. Only seven clock cycles are required, since the first bit is available at SDO after the ID is loaded.

Listing 2. Load Algorithm

To load a device:

```
Call procedure: Get_ID ( to check device)
Call procedure: Change_State ( from IDLE to SHIFT state)
Set row_count =0
Loop until row_count = 15
      Call procedure: Shift_Command, with command: SHIFT_DATA
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Shift_Data_In, with data location in Source ispSTREAM at
(row_count x 24)
      Call procedure: Change_State ( to SHIFT state)
      Call procedure: Shift_Command, with command: PROGRAM
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Execute_Command (starts operation)
      Call procedure: Change_State ( to SHIFT state)
      Call procedure: Shift_Command, with command: SHIFT_DATA
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Shift_Data_Out, with data location in Target ispSTREAM at
      (row_count x 24)
      Call procedure: Change_State ( to SHIFT state)
End Loop
```

Listing 3. Verify Algorithm

To verify a device:

```
Call procedure: Get_ID (to check device type)
Call procedure: Change_State ( from IDLE to SHIFT state)
Set row_count =0
Loop until row_count = 15
      Call procedure: Shift_Command, with command: SHIFT_DATA
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Shift_Data_In, with data location in Source ispSTREAM at
(row_count x 24)
      Call procedure: Change_State ( to SHIFT state)
      Call procedure: Shift_Command, with command: VERIFY
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Execute_Command (starts operation)
      Call procedure: Wait (Verify_Time)
      Call procedure: Change_State ( to SHIFT state)
      Call procedure: Shift_Command, with command: SHIFT_DATA
      Call procedure: Change_State (to EXECUTE State)
      Call procedure: Shift_Data_Out, with data location a 24 bit temporary buffer
      Compare temp row buffer with data location in ispSTREAM to be verified
      against, at (row_count x 24) Verify Error if the 24 bits don’t match
      Call procedure: Change_State ( to SHIFT state)
End Loop
```

Procedure steps:

```
set MODE pin High, and SDI pin Low
wait Tsu
Set SCLK pin High
wait Tclkh
Set SCLK pin Low
set count =0
get value from SDO and store in temp_buffer[0]
set count = 1
loop until count == 7
    bring SCLK pin High
    wait Twh
    bring SCLK pin Low
    wait Twl
    get value from SDO and store in
    temp_buffer[count]
End loop
( Device ID code is now stored in the temp_buffer array)
(END procedure)
```

Change_State Procedure

The Change_State procedure changes the programming state machine to the next state, according to the state diagram. Procedure steps:

```
set MODE pin High, and SDI pin High
wait Tsu
bring SCLK pin High
wait  Th
set MODE pin Low, and SDI pin Low
wait Tclkh
bring SCLK pin Low
(END Procedure)
```

Shift_ Command Procedure

The Shift_Command procedure shifts a five-bit command into the device's shift register. The various commands should be coded so the procedure can use a mnemonic (such as PROGRAM), and the controlling software can use the appropriate five-bit sequence for that command. Procedure steps:

```
set MODE pin Low
set count =0
loop until count == 4
    get next bit of command code (count = bit number)
    set SDI pin to bit value
    wait Tsu
    bring SCLK pin High
    wait Tclkh
    bring SCLK pin Low
    count = count +1
End loop
(END Procedure)
```

Shift_ Data_In Procedure

The Shift_Data_In procedure explains the steps to clock a row of data into the device, reading the data from an ispSTREAM. This procedure shifts in 22 bits of data, and is used for all 16 rows. Procedure steps:

```
set MODE pin Low
set count =0
loop until count == 23
    get next bit from ispSTREAM ( bit number = count x
    row_number)
    set SDI pin to bit value
    wait Tsu
    bring SCLK pin High
    wait Tclkh
    bring SCLK pin Low
End loop
(END Procedure)
```

Shift_ Data_Out Procedure

The Shift_Data_In procedure explains the steps to clock a row of data out of the device and store it in an ispSTREAM. This procedure shifts out 22 bits of data, and is used for all 16 rows. Procedure steps:

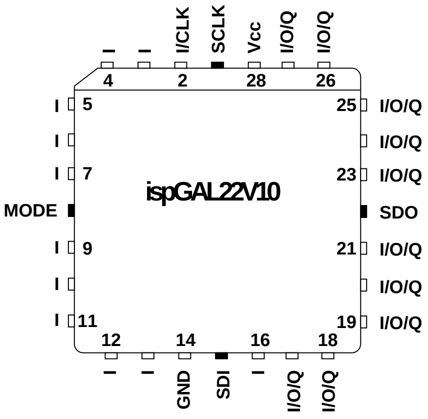
```
set MODE pin Low
wait Tsu
set count =0
loop until count == 23
        bring SCLK pin High
        wait Tclkh
        bring SCLK pin Low
        get value of SDO pin and store as next bit in
        ispSTREAM ( bit number = count x row_number)
End loop
(END Procedure)
```

Execute_Command Procedure

The Execute_Command procedure causes many of the commands to begin executing after the state machine is in the EXECUTE state. Procedure steps:

```
set MODE pin Low, and SDI pin Low
wait Tsu
bring SCLK pin High
wait Twh
bring SCLK pin Low
(END Procedure)
```

Figure 31. ispGAL22V10 28-Pin PLCC Pinout Diagram



Wait Procedure

The Wait procedure waits the indicated time to ensure that various timing parameters are met. This procedure is likely to be used when executing the PROGRAM and ERASE procedures, which need a long delay (tens of milliseconds). The other timing parameters may be guaranteed by the system timing. Various timing parameters should be coded so that a mnemonic may be passed to the procedure. Procedure steps:

```
wait the indicated time
(END Procedure)
```

ispGAL Programming Details

The following sections describe the state machine instruction set, timing parameters, and device layout as they apply to ispGAL devices in general. Figure 31 shows the ispGAL22V10 28-pin device pinout.

Shift Registers

The ispGAL device has four shift registers: Device ID, Instruction, Data, and Architecture. All shift registers operate on a First In-First Out (FIFO) basis, and are enabled by the programming state machine.

The Device ID shift register is only accessible in the IDLE state. It is eight bits long, and is only used to shift out the device ID. For the ispGAL22V10, the ID is defined to be 08 (hex). The Instruction shift register is only accessible in the SHIFT state. It is five bits long, and is only used to shift the Instruction Codes into the device. The Data and Instruction shift registers expect the LSB to be shifted in first. The Data shift register is 138 bits long, and is used to shift all addresses and data into or out of the device. The Data shift register is only accessible in the EXECUTE state when executing a SHIFT_DATA instruction. The Architecture shift register is 20 bits long and the Output Logic Macro Cell (OLMC) 1's S1 architecture bit is shifted in first and OLMC 10's S0 architecture bit is shifted in last. The Architecture shift register is accessed during the EXECUTE state, when the ARCH_SHIFT instruction is executed.

To program an ispGAL device, data is read from a serial bit stream and shifted into the shift registers. The data is read 138 bits at a time, shifted into the device, and then programmed into the device through a programming operation. Table 16 describes the instructions for the ispGAL state machine. The exact sequence and methods for converting a JEDEC map into a serial bit stream are explained in the Internal Architecture section below.

Table 16. ispGAL Programming State Machine Instruction Set

Instruction	Operation	Description
00000	NOP	No operation performed.
00010	SHIFT_DATA	Clocks data into, or out of, the Data Shift Register.
00011	BULK_ERASE	Erases the entire device.
00101	ERASE_ARRAY	Erases everything except the Architecture rows.
00110	ERASE_ARCH	Erases the Architecture rows only.
00111	PROGRAM	Programs the Shift Register data into the addressed row.
01010	VERIFY	Load data from the selected row into the Serial Shift Register.
01101	IOPRLD	Preload the I/O register with given data.
01110	FLOWTHRU	Disables the Shift Register (SDI=SDO).
10100	ARCH SHIFT	Enables the Architecture Shift Register for shifting data into, or out of, the register.

Timing

Programming the ispGAL devices properly requires that a number of timing specifications be met. Most critical are the specifications relating to programming and erasing the E^2CMOS cells. In addition to a minimum pulse width, there is also a maximum specification for these parameters. Refer to the ispGAL programming mode timing specifications for the timing requirements (Table 15), which are identical to the ispGDS specifications. Diagrams for the programming mode specifications are shown in Figures 25, 26, and 27 of the ispGDS timing section.

Securing an ispGAL Device

The ispGAL devices are not secured by an instruction. To secure ispGAL devices, row 61 must be programmed in the same manner that other data rows are programmed. When programming this security row, the data bits are “don't care.”

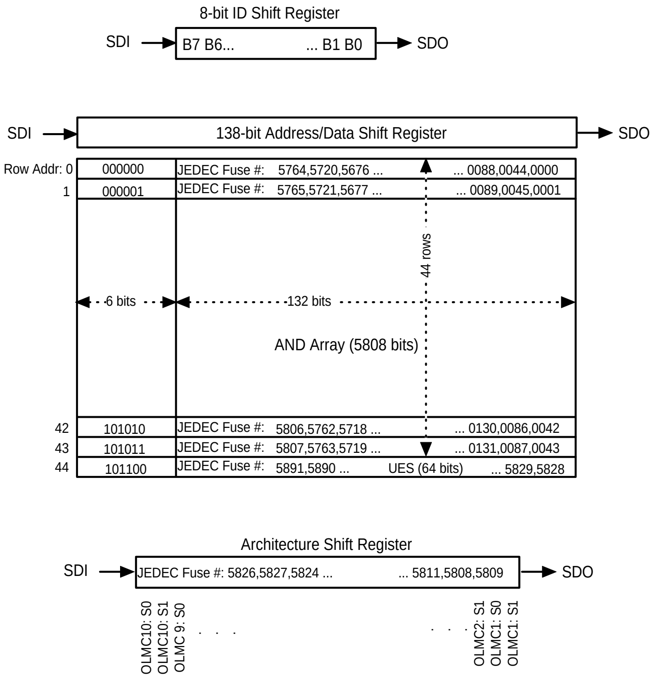
Internal Architecture

This section describes the internal architecture of the device as it relates to programming and covers constructing the ispSTREAM format. If you are using the supplied software tools, a conversion utility (complete with source code) is included to convert an industry-standard JEDEC file to ispSTREAM format. All of the Lattice Semiconductor software routines read and write the ispSTREAM format.

Three components comprise the ispGAL device programming architecture (Figure 32): 44, 132-bit rows of AND array, one 64-bit row of User Electronic Signature (UES), and one 20-bit row of architecture information.

The AND array section of the physical layout is organized so that each column of JEDEC fuse numbers shown in

Figure 32. ispGAL Device Shift Register Layout



the logic diagram of the ispGAL22V10 corresponds to one row of shift register for the device layout. Each physical row is 132 bits long. With each row of AND array data, there is a 6-bit row address associated with it, which including the row address bits, makes the shift registers 138 bits long. The row address bits must be shifted into the shift register along with the AND array data. Executing a PROGRAM command following the combination of data and row address shift programs the row that is specified by the shift instruction.

The I/O preload (IOPRLD) is performed in the same order as the Architecture shift register shown in Figure 32. Once in I/O Preload, the length of the shift register is determined by the number of I/Os that are configured as registered output. The length of the shift register and the order must be determined before IOPRLD can be executed.

The UES row is unique in that it is only 64 bits long. When the row address bits are added to the row itself, the total shift register length required to fully specify the UES row is 70 bits long. In other words, only 70 bits out of the 132-bit shift register are used for the UES. The 20-bit Architecture shift register is selected when the ARCH_SHIFT instruction is executed. The OLMC 0, S1: OLMC 0, S0; OLMC 1, S1: OLMC 1, S0: etc. are shifted in order with the last bit of the shift register being OLMC 10, S0.

Algorithms and Procedures

The ispGAL's programming algorithm and programming procedure are very similar to the ispGDS. For the sake of brevity, please refer to the algorithm and procedures section in the ispGDS section if you are interested in this information. If you have further questions, please call the Lattice Semiconductor Hotline at 1-888-ISP-PLDS.

Table 17. Features of the ISP Device Families

	ispGDS, ispGAL 22V10B/C	ispLSI 1K/E and ispLSI 2K Families	ispLSI 3256	ispLSI 3K (except 3256) and ispLSI 6K Families	ispLSI 2KLV Family
LSC ISP State Machine for Programming	Yes	Yes	Yes	Yes	No
TAP Controller State Machine for Programming	No	No	No	Yes	Yes
Boundary-Scan Test Operations Supported Through the TAP Controller	No	No	Yes	Yes	No
Programming Signals	LSC ISP	LSC ISP	LSC ISP/TAP	LSC ISP/TAP	TAP
Command Shift Register Length	5 bits	5 bits	5 bits/2 bits	5 bits	5 bits
FLOWTHRU Instruction	Yes	Yes	Yes	Yes	Yes
$\overline{\text{ispEN}}$ Signal	No	Yes	Yes	Yes	Yes
Address and Data Shift Registers	Both address and data shifted in one command.	Different shift instructions for address and data.	Different shift instructions for address and data.	Different shift instructions for address and data.	Different shift instructions for address and data.
Device ID	8-bit LSC ISP	8-bit LSC ISP	8-bit LSC ISP	8-bit LSC ISP	32-bit BSCAN TAP ID

ISP Daisy Chain Details

This section provides a detailed look at the issues associated with daisy chain programming. Before examining the details, the reader should understand the differences between ISP devices. This section describes those differences and the unique programming features of each ISP device.

ISP Overview for Daisy Chain

Similarities and Differences Between Devices

For the purpose of cascading, ISP devices can be categorized into five device groups: the ispGDS and ispGAL22V10B/C; the ispLSI 1000/E and ispLSI 2000 families; the ispLSI 3256; the ispLSI 3000 (except ispLSI 3256) and ispLSI 6000 families; and the ispLSI 2000LV family of devices. Table 17 highlights the similarities between these device groups.

The ispGDS and ispGAL22V10B/C devices use only the LSC ISP state machine for programming. The I/O's of ispGDS and ispGAL22V10B/C devices are put into a high impedance state when the programming state machine goes into the Command Shift State. The ispGDS and ispGAL devices do not use a dedicated $\overline{\text{ispEN}}$ pin for this function.

The ispLSI 1000/E and ispLSI 2000 families of devices are programmed exclusively through the LSC ISP state machine but also use a dedicated $\overline{\text{ispEN}}$ pin to enable the programming mode: by driving $\overline{\text{ispEN}}$ low, all of the device I/Os are put into a high-impedance state and the programming functions for SDI, SDO, MODE, and SCLK are enabled.

The ispLSI 3256 is programmed through the LSC ISP state machine but also is boundary-scan compliant for testing. A dedicated $\overline{\text{ispEN}}$ pin selects between the ISP state machine and the TAP controller. By driving $\overline{\text{ispEN}}$ low, all of the device I/Os are put into the high-impedance state and the programming functions for SDI, SDO, MODE, and SCLK are enabled. When $\overline{\text{ispEN}}$ is high the TAP controller is active and the functions for TDI, TDO, TMS and TCK are enabled.

The ispLSI 3000 (except the ispLSI 3256) and ispLSI 6000 families are programmable through either the LSC ISP state machine or the boundary-scan TAP controller. By driving $\overline{\text{ispEN}}$ low, all of the device I/Os 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 $\overline{\text{ispEN}}$ is high, the TAP controller is active and the functions for TDI, TDO, TMS, and TCK are enabled. With the TAP controller active, the device I/Os can also be put into the high-impedance state by loading and executing the Program Enable (ProgEN) instruction. To put the devices into the programming mode, the ProgEN instruction is loaded and executed three times in succession. When the TAP controller is active, boundary-scan test operations are available for the ispLSI 3000 and ispLSI 6000 families of devices.

The ispLSI 2000LV family of devices is programmed exclusively through the boundary-scan TAP controller. A dedicated $\overline{\text{ispEN}}$ pin multiplexes the functionality of the programming pins. When $\overline{\text{ispEN}}$ is held low, the TAP controller is active and the functions for TDI, TDO, TMS, and TCK are enabled. The device enters the programming mode after the Program Enable instruction is loaded and executed three times in succession. Device I/Os go to the high-impedance state after the first ProgEN instruction is loaded.

Figure 33. Configuration for Programming and Boundary-Scan Operations for ispLSI 3000 Family (Except 3256) and ispLSI 6000 Family Devices on the Same Board with ispLSI 2000LV Family Devices.

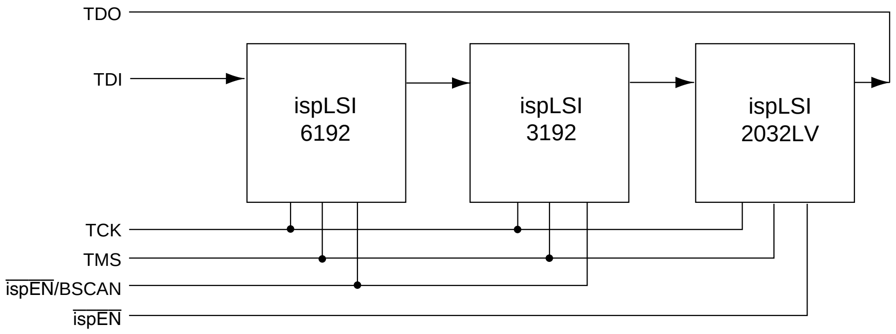
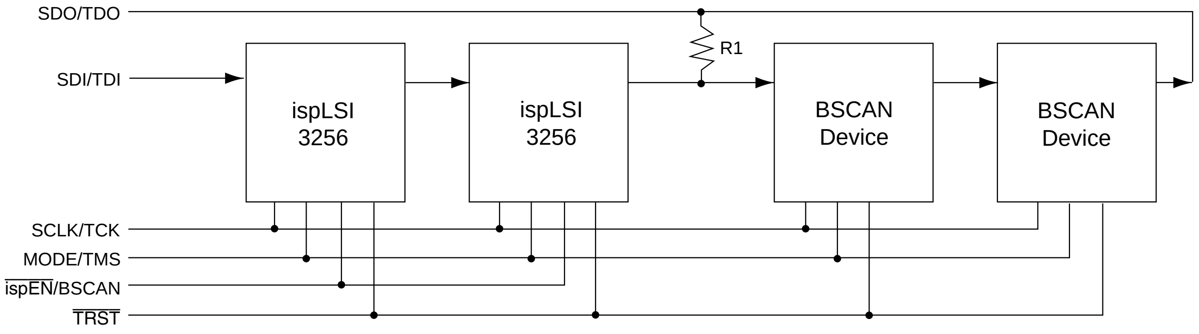


Figure 34. Daisy Chain Configuration Supporting Programming Through the LSC ISP State Machine and Boundary-Scan Operations for ispLSI 3256 Devices with Other Boundary-Scan Devices.



If devices that use the same state machine for programming are put in a serial daisy chain, it is possible to program multiple ISP devices by operating all the state machines in parallel. This synchronizes all the devices within the daisy chain to a known state. However, having all ISP devices in the same state does not mean that all devices are executing the same instruction. The ability of each device in the daisy chain to execute a different instruction makes it possible to selectively program one or multiple ISP devices at a time.

The internal device layout is the same for all ispLSI devices regardless of state machine interface used for programming. The ispLSI devices have separate address and data shift commands. The row(s) are selected by the address that is shifted-in prior to each programming command. The data can then be shifted with the data shift instruction. With ispGDS and ispGAL devices, both address and data are shifted-in with a single shift command (the address is part of the Data shift register). When executing commands that only require a row address, a dummy data stream or no data can be shifted in place of the data stream.

ISP Programming for Mixed LSC ISP and Boundary-Scan Systems

This section highlights the hardware interface when LSC ISP devices are mixed with boundary-scan testable devices and boundary-scan TAP ISP programmable devices on the same board. Following a few simple procedures below 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 LSC ISP interface can be common with corresponding boundary-scan TAP ISP interface signals. This usually includes some or all of SDI and TDI, MODE and TMS, SDO and TDO, and SCLK and TCK. If a parallel programming configuration is used, where the boundary-scan and non-boundary-scan devices are in two separate chains both fed by SDI/TDI and combining SDO/TDO at the end of the chains, the $\overline{\text{ispEN}}$ and $\overline{\text{TRST}}$ pins can be used to select which chain is active. On ispLSI devices the $\overline{\text{ispEN}}$ pin can be used to select the devices for programming, and when the devices are de-selected the programming pins are in the high impedance state and will not affect the programming of active devices. Many boundary-scan devices include the optional $\overline{\text{TRST}}$ pin which holds the TAP controller in the Test-Logic-Reset state. For boundary-scan devices held in the Test-Logic-Reset state, the TDO pin will be in the high-impedance state and programming of the non-boundary-scan daisy chain will not be affected. This type of configuration implies that the ISP programming pins cannot be used for normal mode functions.

Hardware design considerations for new boards include whether the hardware designer will be using boundary-scan test operations or low voltage (3.3V) devices. In a system using 3.3V ISP devices, the ispDOWNLOAD cable v2.0 should be used. The cable operates with either a 3.3V or 5V VCC source. Lattice's Daisy Chain Download software makes the ISP software interface to 3.3V and mixed-voltage systems transparent to the user.

Boundary-scan devices, such as the ispLSI 3000 and 6000 families, can be put into a boundary-scan serial daisy chain for test and programming purposes. The configuration choice for the devices depends on the boundary-scan test operations needed, programming requirements, and device combinations. For both boundary-scan test operations and programming through the boundary-scan TAP the ispLSI 3000 family (except the ispLSI 3256) and the ispLSI 6000 family may be put in any order in a chain of boundary-scan devices. Boundary-scan test operations are available for the ispLSI 3256 through the boundary-scan TAP but programming is done through the LSC ISP state machine. Programming operations are conducted through the boundary-scan TAP for the ispLSI 2000LV family as well and they may also be put in any order in a boundary-scan chain of devices. A sample boundary-scan chain of devices is shown in Figure 33. In this configuration the $\overline{\text{ispEN}}$/BSCAN pin on the ispLSI 3192 and ispLSI 6192 devices is pulled up to VCC either with the internal active pull-up or externally hardwired to VCC. The $\overline{\text{ispEN}}$ pin on the ispLSI 2032LV is driven high for normal operations and driven low for boundary-scan programming operations. The ispDOWNLOAD cable connection to the $\overline{\text{ispEN}}$/BSCAN pin of the ispLSI 2032LV takes care of this operation.

The ispLSI 3256 may also be put into a serial daisy chain with other boundary-scan devices for test operations. However, since programming of the ispLSI 3256 is done through the LSC ISP state machine, if in-system programming is required, the daisy chain must be altered to allow LSC ISP state machine operation independent of the boundary-scan operations.

Figure 34 shows a possible programming configuration that allows in-system programming of ispLSI 3256 devices in the same daisy chain as other boundary-scan devices that only use the TAP controller state machine. This configuration makes use of the optional boundary-scan pin $\overline{\text{TRST}}$ that holds the TAP controller state machine in the Test-Logic-Reset state to split up the chain and allow LSC ISP state machine operation in the first part of the chain. When $\overline{\text{TRST}}$ is held low the TDO pin for the boundary-scan only devices will be in the high-impedance state and the SDO output of the last ispLSI 3256 device in the chain will drive back to the programmer through resistor R1. This will allow programming through the LSC ISP state machine with $\overline{\text{ispEN}}$/BSCAN held low for the ispLSI 3256 devices in the chain without affecting the boundary-scan only devices. When $\overline{\text{TRST}}$ and $\overline{\text{ispEN}}$/BSCAN are both high boundary-scan test operations will be available for the entire chain of devices. In this configuration other ispLSI 3000 and ispLSI 6000 devices can be put in either part of the chain but must have $\overline{\text{ispEN}}$/BSCAN tied to VCC if they are in the BSCAN part of the chain.

If LSC ISP programmable devices such as the ispLSI 1000/E and ispLSI 2000 families will be put on the same board as boundary-scan devices, parallel daisy chain loops are required for successful in-system programming and boundary-scan operations. Figure 35 shows a sample configuration using parallel programming loops. This programming configuration again makes use of the $\overline{\text{TRST}}$ pin to hold the boundary-scan devices in the Test-Logic-Reset state while programming the LSC ISP loop. For boundary-scan operations the $\overline{\text{ispEN}}$ pin is pulled high to de-select the LSC ISP programming loop. The

Figure 35. Parallel Programming Loops for Programming Boundary-Scan and LSC ISP Devices on the Same Board.

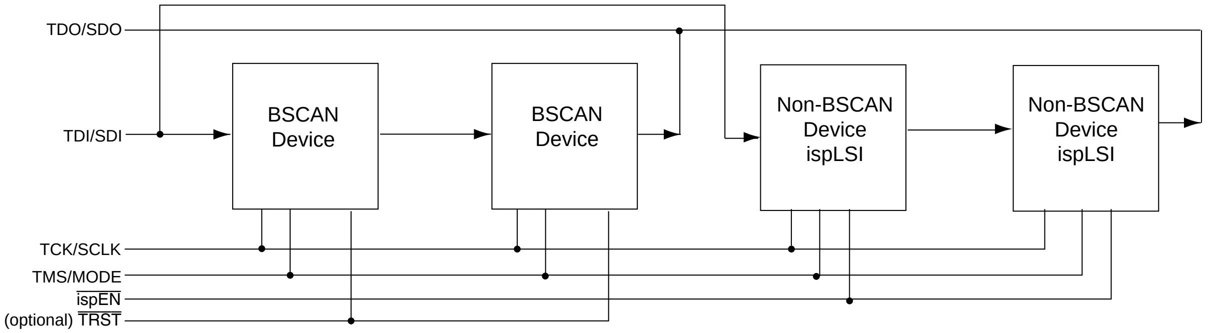


Table 18. ISP Programming Information

Description	ispLSI 1032	ispGAL22V10	ispGDS22	ispLSI 2032
Device ID (8-bits)	0000 0011	0000 1000	0111 0010	0001 0101
Command Register	5 bits	5 bits	5 bits	5 bits
Address Shift Register	108 bits	n/a	n/a	102 bits
Data/Addr. & Data Shift Register	160 bits	(6+132) bits	(6+18) bits	40 bits

advantage of this configuration is being able to use one connector for both boundary-scan testing procedures and LSC ISP programming operations.

The boundary-scan test daisy chain can include any combination of ispLSI 2000LV family, ispLSI 3000 family, and ispLSI 6000 family devices for boundary-scan test operations (Figure 35). If boundary-scan test and programming operations are required with ispLSI 3256 devices, a combination of this configuration with Figure 34 should be used. For both the configuration in Figure 34 and Figure 35 an alternative to controlling the $\overline{\text{TRST}}$ pin is to multiplex the MODE/TMS signal to isolate the two daisy chains.

ISP Daisy Chain Programming

A specific illustration of multiple device programming in a daisy chained environment is shown in Figure 1. This following example shows ISP programming aspects such as identifying the devices in the daisy chain, shifting commands, bypassing devices, and executing commands for a simple serial daisy chain.

All of the programming state machines run in parallel which keeps the devices synchronized. The programming information for the serial daisy chain in Figure 1 is summarized in Table 18. Similar details for any ISP device can be found previously in this section and in the appropriate device data sheet.

The first procedure of the programming sequence identifies the devices in the ISP chain. The following procedure describes one way of reading the device IDs.

```
Load_ID Procedure
set $\overline{\text{ispEN}}$ = L
set MODE, SDI = H, L
clock SCLK  (Load ID)
Continue to Shift_ID Procedure ...
```

At this point, the 8-bit ID registers are loaded with the hardwired device IDs. Figure 36 shows the configuration of the ID shift registers.

After the device ID has been loaded, the following shift ID procedure sequentially shifts the IDs through to the last device's SDO. While the ID is being shifted out, keep SDI at a known logic level so that the end of the ID stream can be identified. This is especially important when there are an unknown number of devices in the ISP daisy chain. By detecting a sequence of eight zeros or eight ones, the ISP controller can detect the end of the ID string.

```
Shift_ID Procedure
... Continued from Load_ID Procedure
set MODE, SDI = L, H
clock SCLK (Shift ID)
if last 8 SDO = H then go to End
else go to Shift_ID
End
```

Now, all of the devices within the ISP daisy chain and their order can be properly identified. The next step is to match the proper JEDEC fuse map file to the appropriate device. There are several programming options at this point. To simplify the programming routines however, this example programs the devices one at a time. In programming time critical applications, the daisy chained devices can be programmed in parallel. The parallel programming routines must keep track of the differences in the fuse map lengths between different ISP devices.

The following procedures illustrate how to shift commands, shift data, and execute commands to program the ispGAL22V10. Since the ispGAL22V10 is the second device in the ISP daisy chain, these procedures also illustrate how to put the other devices into flow-through mode. The following procedure shifts the SHIFT_DATA command into the ispGAL22V10 and the FLOWTHRU command into the rest of the ISP devices.

```
Load_Command Procedure
... Continued from end of Shift_ID Procedure
set MODE, SDI = H, H
clock SCLK (Shift State)
set MODE = L
Loop
    set SDI = command stream (Figure 36b)
    clock SCLK (Shift Command)
End Loop
End Procedure
```

```
Execute_Command Procedure
set MODE, SDI = H, H
clock SCLK (Execute State)
set MODE = L
Loop 138 times
    set SDI = data stream (Figure 36c)
    clock SCLK (Execute SHIFT_DATA Command)
End Loop
set MODE, SDI = H, H
clock SCLK (Shift State)
End Procedure
```

At the end of the Execute_Command procedure, the state machine is returned to the Shift State. This readies the devices for another command shift procedure. For the ispGAL22V10, the DATA_SHIFT instruction of 138 bits includes the row address and the data associated with the row. Similar procedures can be used to complete the programming of the ispGAL22V10.

Figure 36a. ID Shift Register Configuration

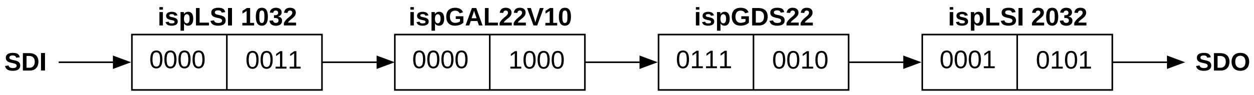


Figure 36b. ISP Command Stream

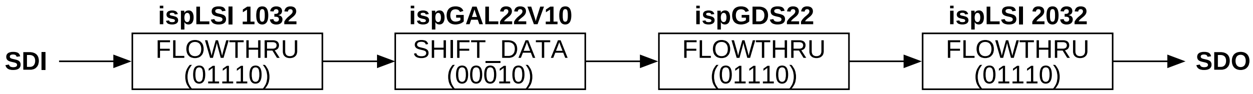
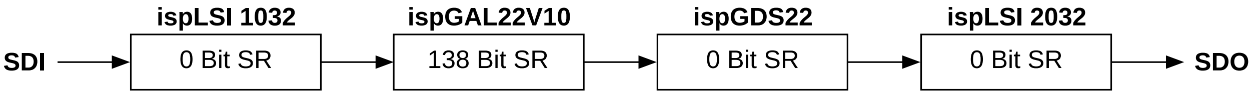


Figure 36c. ISP Data Stream



LATTICE SEMICONDUCTOR CORPORATION
5555 Northeast Moore Court
Hillsboro, Oregon 97124 U.S.A.
Tel.: (503) 681-0118
FAX: (503) 681-3037
http://www.latticesemi.com

November 1996