(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 May 2004 (27.05.2004)

PCT

(10) International Publication Number
WO 2004/044685 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number:
PCT/US2003/035309

(22) International Filing Date:
6 November 2003 (06.11.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/424,333	7 November 2002 (07.11.2002)	US
10/299,007	19 November 2002 (19.11.2002)	US

(71) Applicant *(for all designated States except US)*: **IN-RANGE TECHNOLOGIES CORPORATION** [US/US]; 100 Mount Holly By-Pass, Lumberton, NJ 08048 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **HORANZY, Joseph** [US/US]; 575 South Mount Vernon Circle, Bensalem, PA 19020 (US).

(74) Agents: **TYSVER, Daniel, A.** et al.; Beck & Tysver, P.L.L.C., 2900 Thomas Avenue S., Suite 100, Minneapolis, MN 55416 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2004/044685 A2

(54) Title: A METHOD AND APPARATUS FOR CONFIGURING PROGRAMMABLE LOGIC DEVICES

(57) Abstract: A method and apparatus for configuring and determining the status of a first programmable logic devices includes a microprocessor, a second programmable logic device containing the interface logic for the first programmable device and the microprocessor, a status check that is linked to the second programmable logic device, the status check determines the condition of the first programmable logic device and a storage device linked to the microprocessor. A configuration code for the first programmable logic device is located on the storage device.

A METHOD AND APPARATUS FOR CONFIGURING PROGRAMMABLE LOGIC DEVICES

PRIORITY

[0001] This application claims priority to the provisional U.S. patent application entitled, A Method and Apparatus for Configuring Programmable Logic Devices, filed November 7, 2002, having a serial number 60/424,333, the disclosure of which is hereby incorporated by reference.

FIELD OF THE INVENTION

The present invention relates generally to programmable logic devices. More particularly, the present invention relates to using, configuring and identifying logic devices with a minimum number of memory devices.

BACKGROUND OF THE INVENTION

[0002] The FPGA is an integrated circuit (IC) that can be programmed in the field after manufacture. FPGAs are similar in principle to, but have vastly wider potential application than, programmable read-only memory (PROM) chips. FPGAs are used by engineers in the design of specialized ICs that can later be produced hard-wired in large quantities for distribution to computer manufacturers and end users. Ultimately, FPGAs might allow computer users to

tailor microprocessors to meet their own individual needs. However, there are limitations in FPGAs that can affect a designers decision to use them.

[0003] FPGAs are not only used in the designing of specialized ICS, FPGAs can also be incorporated into actual devices. One of the advantages in using FPGAs is that it enables the device to be multi-functional without having to double the number of internal circuit components. Another advantage in using FPGAs is that logic configured therein is upgradable. Instead of having to change the integrated chip, the existing FPGA can be reprogrammed with a new configuration. Therefore, FPGAs can be upgraded with logic that enables the device to be compatible with latest devices on the market. In the computer industry, this is a distinct advantage because the pace of the technology runs in about a five year cycle.

[0004] One of the disadvantages of using FPGAs is that currently each FPGA requires its own memory device. The memory device serves as the storage area for the configuration logic. The logic is placed into the FPGA and activated once the FPGA has been initialized. One disadvantage to having multiple memory devices for the FPGA is the overall cost. Furthermore, memory prices have a tendency to fluctuate. This cost factor has an affect on cost effective design choices. The other disadvantage with the individual memory devices is the amount of space required on the circuit board. Circuit board space is a never ending battle especially in the view of multi-function devices within an enclosure that is satisfactory with the consumer. The addition of multiple FPGAs has the opposite effect by requiring each FPGA to have its own memory device.

Furthermore, there is a redundancy in the system with memory devices storing the identical configuration logic for each FPGA.

[0005] Finally, integrated chips such as an FPGA can malfunction and disrupt the operation of a system or device. With FPGAs, it is not possible to check the status or condition of the FPGA without pulling the FPGA from its slot. Further complicating the problem is the ability to detect whether the FPGA is the cause of the malfunction.

[0006] Accordingly, it is desirable to provide a system and method to reduce the number of storage devices used to configure an FPGA. The resulting benefit is the reduction of component costs and the increase in availability of real estate on the circuit board. There is an additional need to determine the status or functionality of the FPGAs.

SUMMARY OF THE INVENTION

[0007] The present invention provides a method and apparatus that enables a user to program multiple programmable logic devices with a minimum number of memory devices.

[0008] The present invention to also provides a method and apparatus to detect the condition or status of the programmable logic device.

[0009] The above can be achieved through the use of a novel combination of a single memory device and a processor to program or configure multiple programmable logic devices. The memory serves as the location where the configuration code for the FPGAs is stored. This arrangement enables the user to save and potentially open up valuable real estate on a circuit board within a system.

In accordance with one embodiment of the present invention, an apparatus for configuring a first programmable device includes a microprocessor, a second programmable logic with an interface logic to link the microprocessor and the first programmable logic device and a status check that is linked to the second programmable logic device. The status check determines the condition of the first programmable logic device. The final element is a memory device that is linked to the microprocessor. The memory device provides a configuration code for the first programmable logic device. In this embodiment, the first programmable logic device is a computer programmable logic device (CPLD) while the second programmable device is a field programmable gate array (FPGA).

[0010] In this embodiment, the microprocessor upon reset or boot-up sets a command in the configuration register of the second programmable logic device. In the instance of multiple FPGAs, the command instructs the second programmable device which, if any of the first programmable devices to program.

[0011] The present embodiment can also include a second memory device linked to the microprocessor. The second memory device contains the switching logic to program the first programmable logic device in a different configuration. The second memory device can be an individual memory device for each of the FPGAs. In this configuration, the microprocessor can selectively choose among the differing configuration codes that are currently present in any of the memory devices.

[0012] In an alternate embodiment of the present invention, a method for configuring a first programmable device contains the steps of deasserting a reset line of the first programmable logic device, toggling the program reset line of the first programmable logic device, transmitting an initialization signal from the first

programmable device to a microprocessor, determining the status of the first programmable logic device and downloading a configuration from the second programmable device to the first programmable device.

[0013] In another aspect of the present invention, a further step includes transmitting a completion status signal from the first programmable device to the microprocessor after the step of downloading is completed. The completion status signal indicates to the microprocessor that a successful download has been accomplished. The completion status signal can also indicate whether a fault condition is present or has occurred. If a fault condition is present, the alternate embodiment includes the step of re-downloading a configuration from the second programmable device to the first programmable device.

[0014] In the presence of a fault condition, the alternate embodiment includes the step of transmitting an alarm status to a user interface. The user interface can be an alphanumeric display, a light emitting diode or a serial port.

[0015] In this alternate embodiment, once a successful configuration is downloaded, the invention includes the steps of activating the first programmable logic device. One such method to activate the FPGA is to de-assert the logic-reset line and set a command in the second programmable logic device. The command can be an instruction to determine which first programmable logic device, if any, to download the configuration.

[0016] In another alternate embodiment, an apparatus for configuring a first programmable device includes means for deasserting a reset line of a first programmable logic device, means for toggling the program line of the first programmable logic device, means for transmitting an initialization signal from the

first programmable device to a microprocessor, means for determining the status of the first programmable logic device and means for downloading a configuration from the second programmable device to the first programmable. This alternate embodiment can also include means for transmitting a completion status signal from the first programmable device to the microprocessor after the step of downloading is completed. The completion status signal can indicate a successful download of the configuration or a fault condition. If a fault condition is detected, then this alternate embodiment can include means for re-downloading a configuration from the second programmable device to the first programmable device.

[0017] Upon detecting the fault condition, the apparatus can contain means for transmitting an alarm status to a user interface. The user interface can be an alphanumeric display, a light emitting diode or a serial port.

[0018] Upon reset or boot-up of the microprocessor, the alternate embodiment can include means for activating the first programmable logic device, which in this alternate embodiment, is de-asserting the logic reset line.

[0019] The microprocessor also contains means for setting a command in the second programmable logic device. The command can be an instruction to the second programmable logic device which of the first programmable logic devices to download the configuration.

[0020] There has thus been outlined, rather broadly, the more important features of the invention in order that the detailed description thereof that follows may be better understood, and in order that the present contribution to the art may be better appreciated. There are, of course, additional features of the invention that will

be described below and which will form the subject matter of the claims appended hereto.

[0021] In this respect, before explaining at least one embodiment of the invention in detail, it is to be understood that the invention is not limited in its application to the details of construction and to the arrangements of the components set forth in the following description or illustrated in the drawings. The invention is capable of other embodiments and of being practiced and carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein, as well as the abstract, are for the purpose of description and should not be regarded as limiting.

[0022] As such, those skilled in the art will appreciate that the conception upon which this disclosure is based may readily be utilized as a basis for the designing of other structures, methods and systems for carrying out the several purposes of the present invention. It is important, therefore, that the claims be regarded as including such equivalent constructions insofar as they do not depart from the spirit and scope of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] FIG. 1 is a block diagram of the present invention illustrating a central storage device for the programmable logic devices

[0024] FIG. 2 is a block diagram of an alternate embodiment of the present invention illustrating a central storage device as well as individual storage devices for programmable logic devices.

[0025] FIG. 3 is a flow diagram of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

[0026] A preferred embodiment of the present invention provides a method and apparatus that enables a user to check the status of a programmable logic device and program or configure this device in a manner that is intended or that the user desires. A preferred embodiment of the present inventive apparatus and method is illustrated in FIG. 1. In FIG. 1, the present invention includes a microprocessor 10, which is linked to storage device 12. The storage device 12 contains the application computer executable code, such as the boot code, and the configuration computer code for the field programmable gate arrays (FPGA) 14,16,18.

[0027] In the preferred embodiment, the storage device is non-volatile memory such as static random access memory (SRAM). SRAM (static RAM) is random access memory (RAM) that retains data bits in its memory as long as power is being supplied. Unlike dynamic RAM (DRAM), which stores bits in cells consisting of a capacitor and a transistor, SRAM does not have to be periodically refreshed. Static RAM provides faster access to data and is more expensive than DRAM. SRAM is used for a computer's cache memory and as part of the random access memory digital-to-analog converter on a video card. One of ordinary skill in the art recognizes that the invention is not limited to a configuration of SRAM storage devices. Any storage device capable of storing the configuration code is within the scope of the invention.

[0028] The preferred embodiment is connected to a communications port 20. In the preferred embodiment, the microprocessor 10 is linked to the

communications port 20. The communications port 20 enables differing FPGA configurations to be downloaded to the storage device 12 via the microprocessor 10. From the storage device 12, the configuration code is downloaded to the various FPGAs 14,16,18.

[0029] In order to allow the FPGAs 14, 16, 18 to communicate with microprocessor 10, a computer programmable logic device (CPLD) 22 is located between the two devices. The CPLD 22 contains the interface logic to enable the two devices to communicate. The CPLD 22 also contains the switching logic to program the multiple FPGAs 14,16,18.

[0030] The microprocessor 10 communicates over a local data bus 24 to the FPGAs 14, 16, 18 and to the CPLD 22. Through the bus line 24, the microprocessor 10 has the ability to update, change and/or reconfigure the FPGAs logical functions.

[0031] After a system is initialized or through a boot-up, the initial condition of the reset lines 26,28,30 of the FPGAs 14,16,18 are held de-asserted. The microprocessor 10 then sets a command in a configuration register of the CPLD 22 to instruct which of the FPGAs 14,16,18 are to be programmed. After the CPLD 22 is instructed as to which of the FPGAs 14, 16, 18 to program, the CPLD 22 directs the appropriate control signals to that FPGA 14, which includes toggling the FPGA's program reset signal. Note that the program-reset signal is different from the FPGA's logic reset.

[0032] In response to the program reset signal, the FPGA 14 transmits an initialization status bit to the microprocessor 10 through the CPLD 22. Once this is received, the microprocessor 10 begins to start downloading the configuration

sequence to the FPGA 14 from the storage device 12. Upon completion of this configuration sequence, the microprocessor 10 receives a completion or done bit from the FPGA 14 via the CPLD 22. The completion bit informs the microprocessor 10 that the FPGA 14 was properly programmed. If the completion bit shows a fault condition, the microprocessor 10, in the preferred embodiment, performs another download of the sequence configuration from the storage device.

[0033] At some point in the downloading process, the microprocessor 10 is instructed to report the error. In the preferred embodiment, the microprocessor 10 reports the error after two unsuccessful downloads that result in a fault condition that cannot be corrected. The report sends an alarm status to a user interface such as an alphanumeric display, LED or a serial port or a terminal.

[0034] After a successful download configuration to the FPGA 14, the FPGA 14 is activated. In the preferred embodiment, the microprocessor 10 activates the FPGA 14 with the new logic configuration by de-asserting the logic re-set line 26. Once activated, the microprocessor 10 tests the FPGA 14 and, if applicable, its surrounding circuitry by sending command sequences to the FPGAs interface register. The FPGAs 14,16,18 are assessed as to their status by using handshake/status signals 32,34,36. The microprocessor 10 requests status back through the predetermined registers in the FPGA 14. If the status is not what is expected, the microprocessor 10 transmits an alarm status indicating what section of the circuitry failed via the user interface.

[0035] The above configuration is also beneficial in a system where the power source is not large enough to program large multiple FPGAs concurrently.

The microprocessor 10 can selectively program the FPGAs in a sequence, which is beneficial in reducing overall drain on a power supply during configuration. In other words, the present invention can program the FPGAs 14, 16, 18 either simultaneously or one at a time.

[0036] FIG. 2 is a block diagram of an alternate embodiment of the present invention illustrating a central storage device and individual storage device for each programmable logic device. In this alternate embodiment as with the preferred embodiment in FIG.1, the microprocessor 10 is linked to the FPGAs 14, 16, 18 through the CPLD 22. Attached to the microprocessor 10 is a storage device 10 and a communications port 20. The difference in this alternate embodiment is the inclusion of FPGA storage devices 38, 40, 42 with the FPGAs 14,16,18. In this alternate embodiment, the microprocessor 10 is able to control where the FPGAs 14,16,18 obtain its configuration code. The configuration code can be pulled from the FPGAs 14,16,18 own storage device 38,40,42 or from a central location as such the storage device 12 that is attached to the microprocessor 10.

[0037] This alternate embodiment enables the FPGAs 14,16,18 to have the ability to have a primary configuration stored in it's storage device 38,40,42 and then be able to switch to a secondary configuration logic by downloading the configuration data stored in the storage device 12, which is attached to the microprocessor 10. The alternate embodiment can also be configured so that the FPGAs 14,16,18 are initialized with the configuration data stored in its own storage device 38,40,42. The alternate embodiment can also be programmed to initialize the FPGAs 14,16,18 with the configuration data stored in the storage

device 12 attached to the microprocessor 10. The benefit of such a configuration enables a device to be multifunctional without the need for a different component set.

[0038] FIG. 3 is a flow diagram of the present invention. The present invention begins the process by the step 44 of deasserting a reset line of a FPGA. This steps ensures that the FPGA is not initialized with any configuration data already within it. In other words, the FPGA is instructed to withhold initializing any configuration logic until another instruction is given. This enables the present invention to begin the process of programming the FPGA is a manner desired. To accomplish this task, the step 46 of toggling the program-reset line of the programmable logic device is undertaken. From this point, the step 48 of transmitting an initialization signal is received by the microprocessor 10 from the FPGA. The initialization signal enables the present invention to perform the step 50 of determining the status of the first programmable logic device from the initialization signal. After a determination is made and no fault is detected, the present invention proceeds to the step 52 of downloading a configuration from a CPLD or second programmable device to a FPGA or a first programmable logic device. Once the download has been completed, the FPGA completes the step 54 of transmitting a completion status signal to the microprocessor. If the status signal indicates a successful download, then the present invention proceeds to the step 56 of activating the first programmable logic device. If the status signal does not indicate a successful download, then the present invention proceeds to the step 58 of re-downloading a configuration data from the CPLD. The present invention further includes the step 60 of alerting a user of the unsuccessful download. The alert can

be through a user interface such as alphanumeric display, and LED or a serial port.

[0039] The many features and advantages of the invention are apparent from the detailed specification, and thus, it is intended by the appended claims to cover all such features and advantages of the invention which fall within the true spirits and scope of the invention. Further, since numerous modifications and variations will readily occur to those skilled in the art; it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly, all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.

What is claimed is:

1. An apparatus for configuring a first programmable device comprising:

a microprocessor;

a second programmable logic comprising an interface logic, which links the microprocessor and the first programmable logic device;

a status check that is linked to the second programmable logic device, the status check determines the condition of the first programmable logic device; and

a memory linked to the microprocessor, the memory comprises a configuration code for the first programmable logic device.

2. The apparatus as in claim 1, wherein the second programmable logic device is a computer programmable logic device (CPLD).

3. The apparatus as in claim 1, wherein the first programmable device is a field programmable gate array (FPGA).

4. The apparatus as in claim 1, wherein the microprocessor sets a command in a configuration register of the second programmable logic device.

5. The apparatus as in claim 4, wherein the command instructs the second programmable device which, if any, first programmable device to program.

6. The apparatus as in claim 1, further comprising a second memory device linked to the microprocessor.

7. The apparatus as in claim 6, wherein the second memory device comprises an alternate configuration code for the first programmable logic device.

8. The apparatus as in claim 1, wherein the status check directs control signals to the first programmable logic device.

9. The apparatus as in claim 8, wherein the control signal is an initialization status.

10. The apparatus as in claim 9, wherein the initialization status is transmitted to the microprocessor.

11. A method for configuring a first programmable device comprising the steps of:

deasserting a reset line of a first programmable logic device;

toggling the program reset line of the first programmable logic device;

transmitting an initialization signal from the first programmable device to a microprocessor;

determining the status of the first programmable logic device from the initialization signal; and

downloading a configuration from a second programmable device to the second programmable device if there is no fault.

12. The method as in claim 11, further comprising the step of transmitting a completion status signal from the first programmable device to the microprocessor after the step of downloading is completed.

13. The method as in claim 12, wherein the completion status signal indicates a successful download of the configuration.

14. The method as in claim 12, wherein the completion status signal indicates a fault condition.

15. The method as in claim 14, further comprising re-downloading a configuration from the second programmable device to the first programmable.

16. The method as in claim 14, further comprising the step of transmitting an alarm status to a user interface.

17. The method as in claim 16, wherein the user interface is an alphanumeric display.

18. The method as in claim 16, wherein the user interface is a light emitting diode.

19. The method as in claim 16, wherein the user interface is a serial port.

20. The method as in claim 11, further comprising the step of activating the first programmable logic device.

21. The method as in claim 20, wherein the step of activating is accomplished by the step of de-asserting the logic reset line.

22. The method as in claim 11, further comprising the step of setting a command in the second programmable logic device.

23. The method as in claim 22, wherein the command is an instruction which first programmable logic device to download the configuration.

24. An apparatus for configuring a first programmable device comprising:

means for deasserting a reset line of the first programmable logic device;

means for toggling the program resent line of the first programmable logic device;

means for transmitting an initialization signal from the first programmable device to a microprocessor;

means for determining the status of the first programmable device from the initialization signal; and

means for downloading a configuration code from a second programmable device to the first programmable.

25. The apparatus as in claim 22, further comprising means for transmitting a completion status signal from the first programmable device to the microprocessor after the step of downloading is completed.

26. The apparatus as in claim 23, wherein the completion status signal indicates a successful download of the configuration.

27. The apparatus as in claim 23, wherein the completion status signal indicates a fault condition.

28. The apparatus as in claim 25, further comprising means for re-downloading a configuration from the second programmable device to the first programmable.

29. The apparatus as in claim 25, further comprising means for transmitting an alarm status to a user interface.

30. The apparatus as in claim 26, wherein the user interface is an alphanumeric display.

31. The apparatus as in claim 27, wherein the user interface is a light emitting diode.

32. The apparatus as in claim 27, wherein the user interface is a serial port.

33. The apparatus as in claim 22, further comprising means for activating the first programmable logic device.

34. The apparatus as in claim 31, wherein means for activating de-asserts the logic reset line.

35. The apparatus as in claim 22, further comprising means for setting a command in the second programmable logic device.

36. The apparatus as in claim 33, wherein the command is an instruction which first programmable logic device to download the configuration.

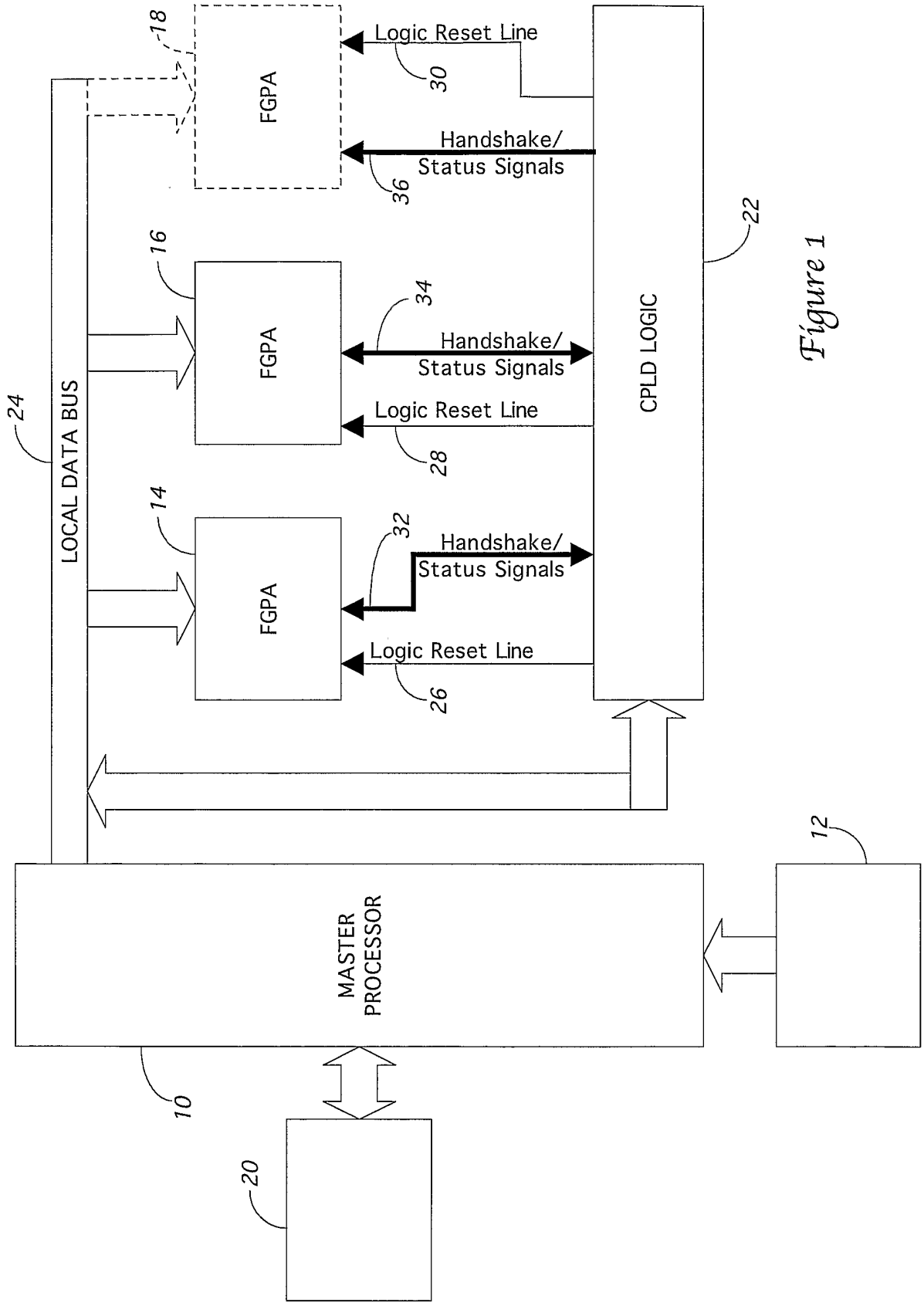

18
FGPA
Logic Reset Line
30
Handshake/
Status Signals
36
22
CPLD LOGIC
16
FGPA
34
Handshake/
Status Signals
Logic Reset Line
28
24
LOCAL DATA BUS
14
FGPA
32
Handshake/
Status Signals
Logic Reset Line
26
12
MASTER
PROCESSOR
10
20

Figure 1

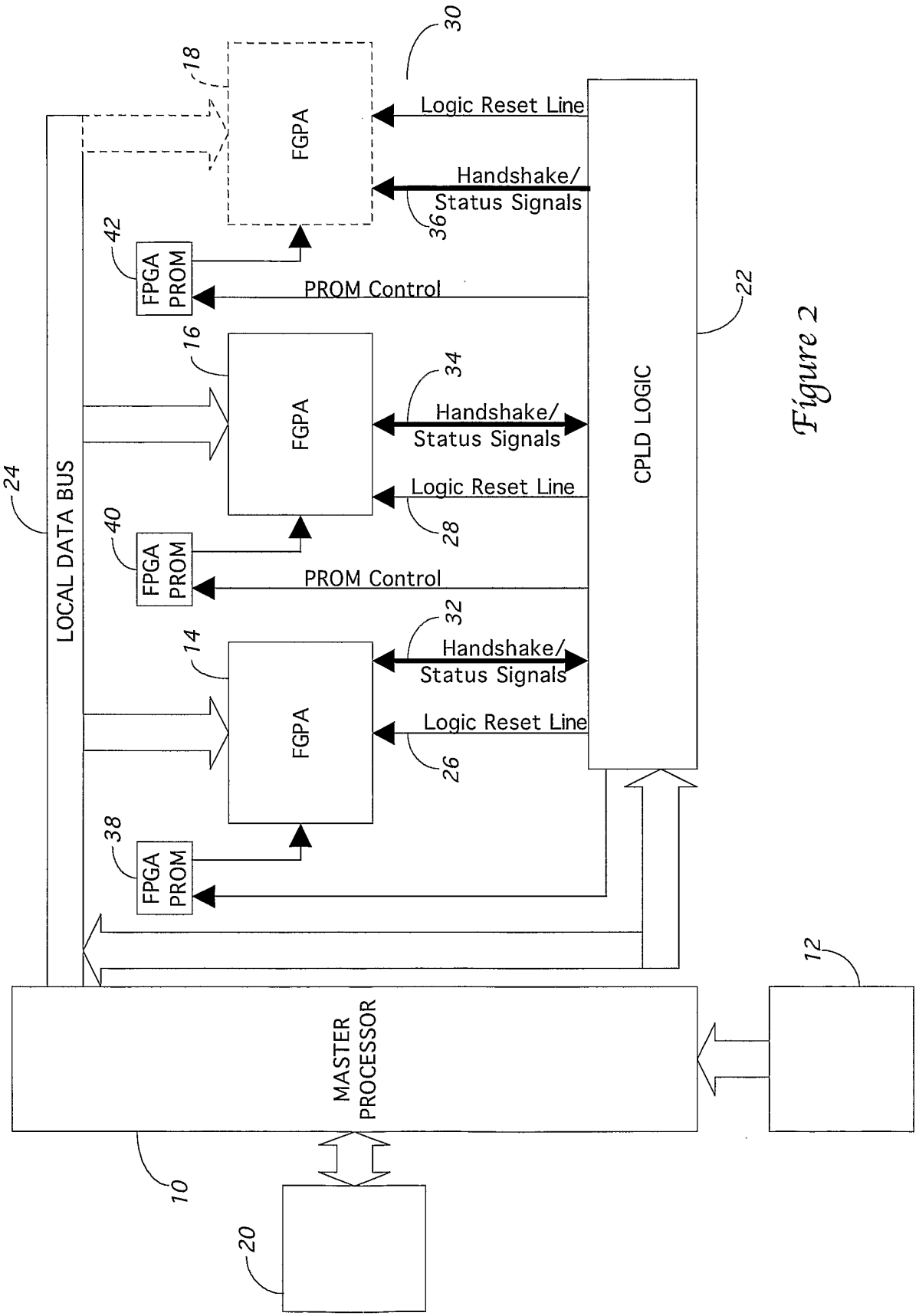
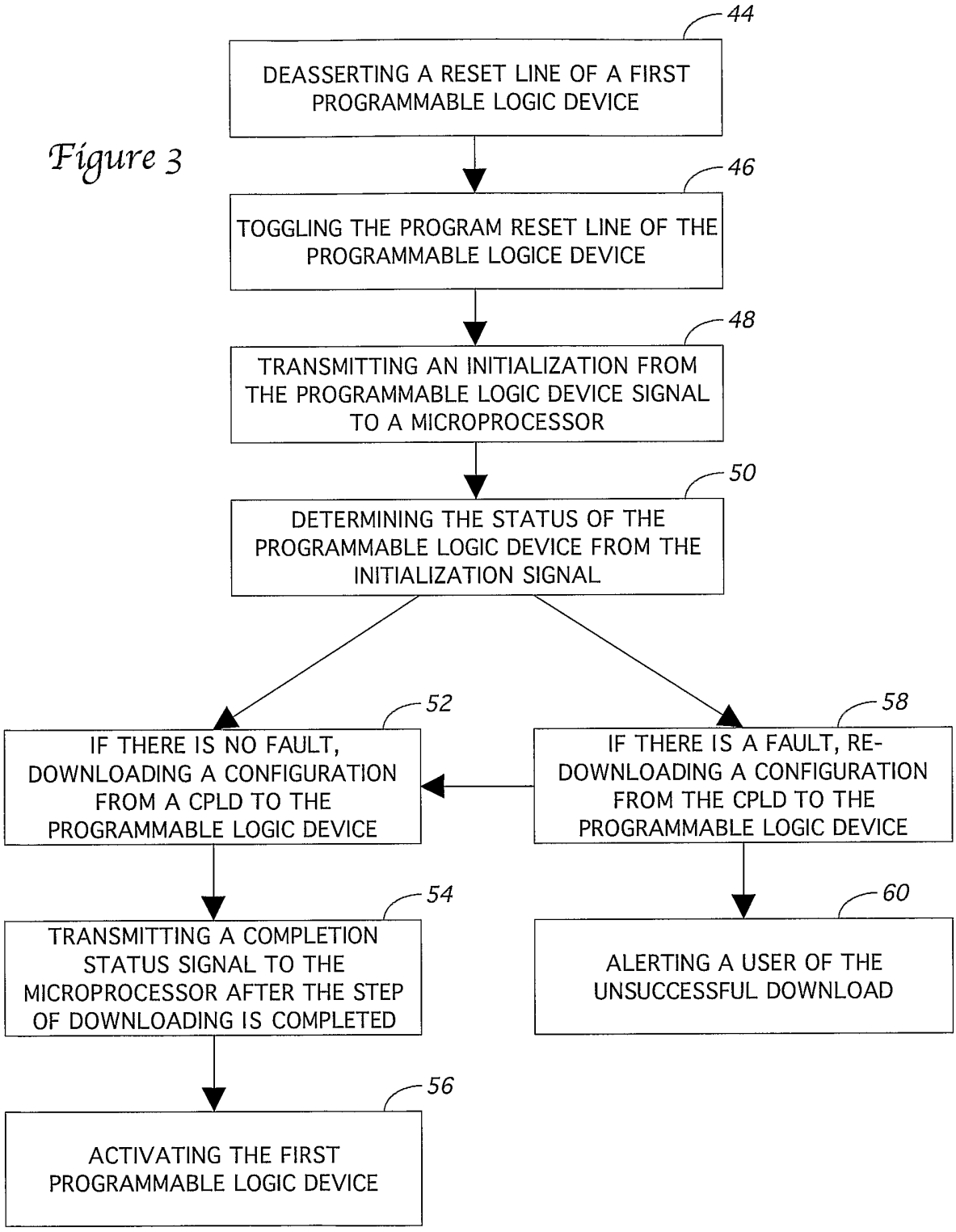

30
18
FGPA
Logic Reset Line
Handshake/
Status Signals
36
42
FPGA
PROM
PROM Control
22
16
FGPA
34
Handshake/
Status Signals
CPLD LOGIC
24
LOCAL DATA BUS
Logic Reset Line
28
40
FPGA
PROM
PROM Control
32
14
Handshake/
Status Signals
FGPA
Logic Reset Line
26
38
FPGA
PROM
12
MASTER
PROCESSOR
10
20

Figure 2

Figure 3


44
DEASSERTING A RESET LINE OF A FIRST PROGRAMMABLE LOGIC DEVICE
46
TOGGLING THE PROGRAM RESET LINE OF THE PROGRAMMABLE LOGICE DEVICE
48
TRANSMITTING AN INITIALIZATION FROM THE PROGRAMMABLE LOGIC DEVICE SIGNAL TO A MICROPROCESSOR
50
DETERMINING THE STATUS OF THE PROGRAMMABLE LOGIC DEVICE FROM THE INITIALIZATION SIGNAL
52
IF THERE IS NO FAULT, DOWNLOADING A CONFIGURATION FROM A CPLD TO THE PROGRAMMABLE LOGIC DEVICE
58
IF THERE IS A FAULT, RE-DOWNLOADING A CONFIGURATION FROM THE CPLD TO THE PROGRAMMABLE LOGIC DEVICE
54
TRANSMITTING A COMPLETION STATUS SIGNAL TO THE MICROPROCESSOR AFTER THE STEP OF DOWNLOADING IS COMPLETED
60
ALERTING A USER OF THE UNSUCCESSFUL DOWNLOAD
56
ACTIVATING THE FIRST PROGRAMMABLE LOGIC DEVICE