Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2354248 C 2006/09/19

(11)(21) 2 354 248

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(22) Date de dépôt/Filing Date: 2001/07/27

(41) Mise à la disp. pub./Open to Public Insp.: 2002/02/02

(45) Date de délivrance/Issue Date: 2006/09/19

(30) Priorité/Priority: 2000/08/02 (US09/631130)

(51) Cl.Int./Int.Cl. *G06F 11/30* (2006.01), *G01R 31/3185* (2006.01), *G01R 31/317* (2006.01), *G06F 9/44* (2006.01)

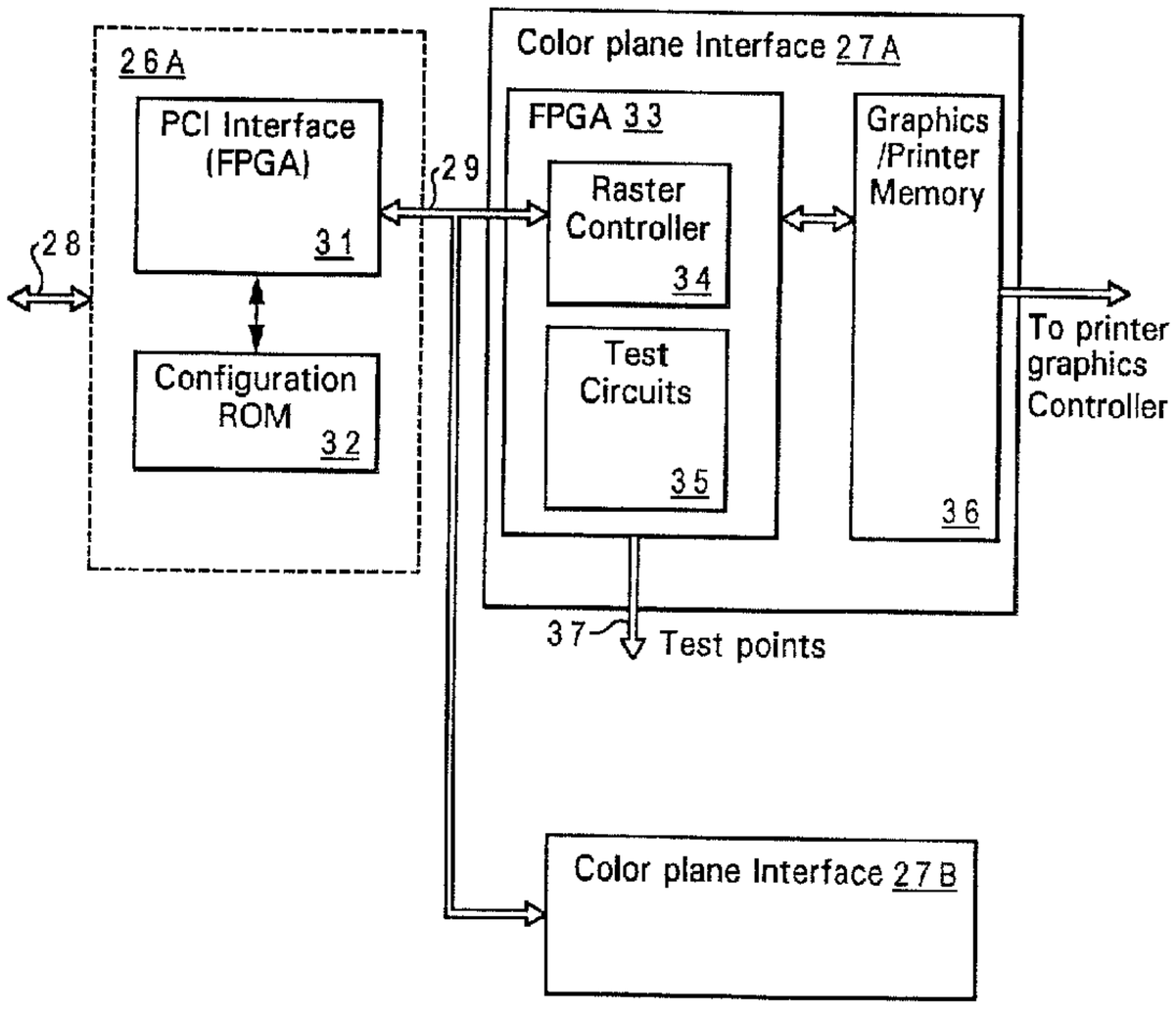
(72) Inventeur/Inventor:
HANNA, STEPHEN DALE, US

(73) Propriétaire/Owner:
INTERNATIONAL BUSINESS MACHINES CORPORATION, US

(74) Agent: HOICKA, LEONORA

(54) Titre : METHODE ET APPAREIL D'ANALYSE DE L'ETAT DU MATERIEL AU MOYEN DE CIRCUITS D'ESSAI DYNAMIQUEMENT RECONFIGURABLES

(54) Title: METHOD AND APPARATUS FOR TRACING HARDWARE STATES USING DYNAMICALLY RECONFIGURABLE TEST CIRCUITS



(57) Abrégé/Abstract:
A method and apparatus for tracing hardware states using dynamically reconfigurable test circuits provides improved debug and troubleshooting capability for functional logic implemented within field programmable logic arrays (FPGAs). Special test logic configurations may be loaded to enhance the debugging of a system using FPGAs. Registers are used to capture snapshots of internal signals for access by a trace program and a test multiplexer is used to provide real-time output to test pins for use with external test equipment. By retrieving the hardware snapshot information with a trace program running on a system in which the FPGA is used, software and hardware debugging are coordinated, providing a sophisticated model of overall system behavior. Special test circuits are implemented within the test logic configurations to enable detection of various events and errors. Counters are used to capture count values when system processor execution reaches a hardware trace point or when events occur. Comparators are used to detect specific data or address values and event detectors are used to detect particular logic value combinations that occur within the functional logic.

METHOD AND APPARATUS FOR TRACING HARDWARE STATES USING DYNAMICALLY RECONFIGURABLE TEST CIRCUITS

ABSTRACT OF THE DISCLOSURE

A method and apparatus for tracing hardware states using dynamically reconfigurable test circuits provides improved debug and troubleshooting capability for functional logic implemented within field programmable logic arrays (FPGAs). Special test logic configurations may be loaded to enhance the debugging of a system using FPGAs. Registers are used to capture snapshots of internal signals for access by a trace program and a test multiplexer is used to provide real-time output to test pins for use with external test equipment. By retrieving the hardware snapshot information with a trace program running on a system in which the FPGA is used, software and hardware debugging are coordinated, providing a sophisticated model of overall system behavior. Special test circuits are implemented within the test logic configurations to enable detection of various events and errors. Counters are used to capture count values when system processor execution reaches a hardware trace point or when events occur. Comparators are used to detect specific data or address values and event detectors are used to detect particular logic value combinations that occur within the functional logic.

METHOD AND APPARATUS FOR TRACING HARDWARE STATES USING DYNAMICALLY RECONFIGURABLE TEST CIRCUITS

BACKGROUND OF THE INVENTION

Technical Field

The present invention relates generally to built-in test circuits implemented in field programmable gate arrays (FPGAs), and more specifically to a method and apparatus for tracing hardware states within an FPGA.

Description of the Related Art

Field Programmable Gate Arrays (FPGAs) provide flexibility in implementing logic designs by allowing reconfiguration of logical circuits via download of binary information. Recent developments in FPGA technology have led to the availability of FPGAs with over 100,000 gates or more within a single Integrated Circuit (IC) package.

FPGAs are frequently used in dedicated peripherals attached to computer systems, particularly in graphics applications, where their high speed and reconfigurability yield an advantage. For example, graphics display electronics and printer graphics electronics that convert bit-plane information into serial data streams have been implemented using the FPGA technology. Because microprocessors and microcontrollers are not efficient for the high-speed serialization/de-serialization of bitstreams, dedicated very-large-scale integrated (VLSI) circuits are used for this purpose. The VLSI circuits are typically gate arrays, having a mask that is designed once and never modified until another version of the VLSI circuit is designed and verified. VLSI circuits have a high non-recurring engineering (NRE) cost for mask design and production tooling. FPGAs provide an alternative solution having advantages including quick design and modification turn-around and reconfigurability.

Since the FPGA designs can be quickly modified during the design process and for version upgrades, and since alternate logic configurations may be supported, a method that matches the short design turn cycle (in many cases less than one hour) to the verification and debugging process would be desirable. It would also be desirable to allow for field debugging in cases where there may be a quick solution to a field site problem by modifying the logic, but there is no ready way to verify the low-level behavior of a new logic design in the field.

Due to the complexity of circuits that can be implemented in a present-day FPGA, there is a need to provide measurement of intermediate signals within the FPGA, but without using a significant number of an FPGA's Input/Output (I/O) pins from the FPGA.

Therefore, it would be desirable to provide a method and apparatus for tracing logic states within an FPGA, and further provide for field testing and design debugging of computer peripherals using FPGAs in their implementations.

SUMMARY OF THE INVENTION

The above-mentioned objectives are achieved in a method and apparatus for tracing hardware states within functional logic using dynamically reconfigurable test circuits. One or more sets of reconfigurable test circuits are used to make measurements for debugging and troubleshooting. A particular test circuit can be selected by a software tracing program and test information may be read from the test circuit by a microcontroller or microprocessor. The information can be synchronized with software trace information, providing a unified trace history of software and hardware. The test circuits may incorporate counters, event detectors, comparators and other miscellaneous test circuits that may enable design engineers or field service personnel to determine more readily the cause of problems within the functional logic or software.

In one aspect, the present invention is directed to a method for tracing hardware states within a functional logic within a field programmable logic circuit using dynamically reconfigurable test circuits. The method comprises the steps of first selecting, by a software trace program, a test mode from a plurality of test modes for tracing the operation of said functional logic, then, after said step of selecting a test mode, second selecting one of a plurality of configurations for said dynamically reconfigurable test circuits in conformity with said selected test mode, then configuring said field programmable logic circuit using said selected configuration, then tracing execution of program code to create a software trace history and periodically reading a set of registers within said dynamically reconfigurable test circuits to create a hardware trace log in response to said tracing step reaching a hardware trace request, and recording contents of said registers in synchronization with said software trace history to produce a full trace history.

In a second aspect, the present invention is directed to a system for controlling an electronic device. The system comprises a field programmable logic circuit in which functional logic and a plurality of dynamically reconfigurable test circuits are implemented wherein the plurality of test circuits includes a register set, a processor for executing program instructions to control said electronic device only for instantiating the plurality of test circuits by loading a selected one of a plurality of configurations into said field programmable logic circuit, and a storage for storing the plurality of configurations for configuring said field programmable logic circuit and for storing a test program. An interconnect couples the field programmable logic circuit, the processor and the storage. The test program includes instructions for tracing execution of program code to create a software trace history, instructions for periodically reading a set of registers within the dynamically reconfigurable test circuits to create a hardware trace log in response to receipt of a hardware trace request, and instructions for recording content of said registers in synchronization with said software trace history to produce a full trace history.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives, and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein like reference numerals indicate like components, and:

Figure 1 is a block diagram of a computer network having a network compatible printer, in which a preferred embodiment of the present invention can be implemented;

Figure 2 is a block diagram of the network compatible printer from **Figure 1**;

Figure 3 is a block diagram of a PCI interface and color plane memory module within the network compatible printer of **Figure 2**;

Figure 4 is a block diagram of an FPGA implementing control circuits within the color plane memory module of **Figure 3**, in accordance with a preferred embodiment of the present invention; and

Figure 5 is a flow diagram depicting a method for tracing hardware states using dynamically configurable test circuits, in accordance with a preferred embodiment of the invention.

DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

With reference now to the figures, and in particular with reference to **Figure 1**, there is depicted a block diagram of a computer network having a network compatible printer, in which a preferred embodiment of the present invention can be implemented. An application server **12** is coupled to a network **15** having a workstation **13** and a printer **14**. Application server **12** may perform services requested by workstation **13**, such as spooling a print request targeted at printer **14**.

Within printer **14**, a local control console **17** provides user interface capability for programming and otherwise, controlling operation of printer **14**, including selection and display of special test modes in accordance with the present invention. Workstation **13** may also control test modes and trace formats via a network and interface **16** may also be used to display the results of the trace of the present invention. Network interface **16** provides the printer connection to network **15** and receives printer commands and graphics data, network interface **16** also returns printer status and query responses over network **15**, including trace results in accordance with an embodiment of the present invention. A print server **18** acts as a server to network **15** and may perform all spooling functions, and other print support operations. Printer engines **19** embody the portions of printer **14** that control printing operations, convert graphics input data, control printhead/laser position and print state, etc.

Referring now to **Figure 2** there is depicted a detailed block diagram of printer **14**. Network interface **16** couples network **15** to processors **23A** and **23B** within print server **18**. Processors **23A** and **23B** are coupled to a memory **24** that stores data and program instructions for use by processors **23A** and **23B**. PCI bus **28** couples processors **23A** and **23B**, memory **24** a local control console **17**. Within local control console **17** are a touch screen **21** and a keypad **22** for controlling printer **14** operations.

Printer engines **19** contain color plane interfaces **27A-D** in accordance with the preferred embodiment of the present invention. Color plane interfaces **27A-D** are coupled to PCI bus **28** by

PCI interfaces **26A** and **26B**. Printer engines **19** control printer graphics controller **25** to provide graphical output for driving a printhead of a laser printer, inkjet printer, etc.

Referring now to **Figure 3**, details of PCI interface **26A** and color plane interface **27A** of **Figure 2** are depicted. PCI interface FPGA **31** is configured at startup by Configuration ROM **32.** The power-up configuration of PCI interface FPGA is pre-configured to read FPGA configuration information severally from configuration ROM **32**. Once the configuration contained within configuration ROM **32** is loaded, PCI interface FPGA implements a PCI bus interface and may be further configured via this interface.

Color plane interface **27A**, also contains an FPGA **33**. FPGA **33** is configured by configuration image data sent to PCI interface **31** by processor **23A** or **23B**. When programmed with configuration data, FPGA **33** contains Test Circuits **35** as well as functional circuits such as Raster controller **34**. Raster controller **34** interfaces graphics/printer memory **36** to PCI bus **23** via PCI interface **31**, allowing Raster controller **34** to receive graphics data and convert it to data within graphics/printer-memory **36** for use by printer graphics controller **25**. Test circuits, which have designs that vary by desired testing abilities, can be coupled to virtually any signal within Raster controller **34**, allowing advanced testing, debug and monitoring of Raster controller **34.**

Test points **37** are provided for real time external measurements of signals within FPGA **33**, and results may also be returned to processors **23A** and **23B** via PCI interface **26A** over PCI bus **28** for display of debug information to a user.

Referring now to **Figure 4**, configuration of FPGA **33** in accordance with an embodiment of the present invention is depicted. Interface **29** is coupled to interface circuits **41**, which provide means for PCI Interface **31** to communicate with Raster Controller **34**. Internal signals **50, 51** and **52** from Raster Controller are coupled to test circuits **35.** These test circuits may be varied as needed. Designs are produced via FPGA design software and binary image files are produced that may be stored within persistent storage coupled to network **15**, embedded within program code within

memory **24** or selected from a configuration ram that may be optionally coupled to FPGA **33**. Depending on a desired particular test or debugging sequence or a level of trace output required (sparse vs. verbose), different configurations may be loaded into FPGA **33**.

In the configuration of FPGA **33** depicted in **Figure 4**, exemplary test circuits **35** are shown. Some elements may be common to all configuration images, while others are used for specific tests and debug operations. It is helpful during the debug/design phase of a peripheral device to build a library of these test circuits that can be later used for maintenance or field troubleshooting. Particular points can be selected from the functional logic and particular circuitry can be implemented withing test circuits **35** to allow troubleshooting to be a progressive process, with new configuration images developed for download to FPGA **33** until the information needed to solve a particular problem is available. A configuration providing less test detail about a particular portion of the functional logic, but general information about the entire operation of the functional logic is useful in the production cycle of a product containing test circuits in accordance with the present invention.

The information that is held in register array **42** and presented to interface **29** may also be varied as a function of error detection. For example, raster controller **34** may contain circuitry for detecting parity errors, graphics object sequence errors and graphics object type errors. Depending on the type of error detected, different subsets of internal signals **52** may be latched in register array **42**. The selection of the particular signals is based upon the type of error, in order to present the most useful information for that type of error, or all of internal signals **52** can be latched and the output format of the trace program varied to display the information in a form that assists in determining the cause of an error. The trace program output can also be transmitted over network **15**, allowing remote debugging. An Internet connection might be used to provide a graphical or text interface accessible from an Internet or intra-net connection.

Data selector **44** interfaces direct signals **51** from the functional logic (Raster Controller **34**) as well as data from register array **42**, which can provide static versions of signals **52** from Raster

Controller **34**. Data selector **44** also selects data from specialized test circuits such as comparators **45**, counter **48**, as well as other miscellaneous test circuits **49**. Comparators **45**, provide a means for determining when specific values occur, for example producing an output when a particular data value or address is processed by Raster Controller **34.** Event Detector **47** may detect a particular combination of logic signals **53** from Raster Controller **34**. Counter **48** can be configured to count how many events detectable by event detector **47** have occurred, or may count transitions of some other clock signal, such as a data strobe between events, such as a parity error. Counters are also useful for determining the source of errors that are data dependant. For example, counters may be implemented within test circuits **35** to determine which graphics object is being processed by raster controller **34** and precisely which values within the graphics object are being processed when the error occurs.

The output of data selector **44** is provided through interface circuits **41** to interface **29**. The values selectable by data selector **44** can thus be accessed over PCI bus **28** via PCI interface **26A** and thus used by processors **23A** and **23B.**

Latch **46,** is provided to select test point Signals **50** for output to physical pins on FPGA **33** via Multiplexer **43**. This allows selection of real time output of signals from Raster Controller **34** that can be used with a logic analyzer, oscilloscope or other device. Multiplexer **33** can also be controlled by an external pin connection to control the selection of internal nodes output to test point signals **50**, the external pins can supply the multiplexer select signals, instead of or in addition to the select signals supplied from latch **46**.

Referring now to **Figure 5**, there is illustrated a flow diagram of a method for tracing hardware states using dynamically reconfigurable test circuits in accordance with a preferred embodiment of the invention. This method is generally embodied in a software trace program running within printer **14**.

A user or software program selects a trace format (debug level) for output (step **60**). A logic

configuration for FPGA **33** is selected in conformity with the trace format (step **61**) and the logic configuration is loaded into FPGA **33** (step **62**). Multiplexer outputs for external test points **37** are selected (step **63**) to provide any external signals desired for debugging. When the trace program reaches a dump request (decision **64**) the register array contents are read (step **65**) and the trace program outputs them to a display or as a file, or network data. If during execution an error is detected by the trace program for which a dump request is enabled (decision **66**), selected or all) register array contents are read (step **67**) and output to a display or as a file or network data. The steps of multiplexer selection **53**, and hardware tracing **54** and **56** are repeated until the trace program terminates or the printer is shut down (step **68**).

Although the invention has been described with reference to specific embodiments, this description is not meant to be construed in a limiting sense. Various modifications of the disclosed embodiments, as well as alternative embodiments of the invention, will become apparent to persons skilled in the art upon reference to the description of the invention. It is therefore contemplated that such modifications can be made without departing from the spirit or scope of the present invention as defined in the appended claims.

The invention in which an exclusive property or privilege is claimed is defined as follows:

1. A method for tracing hardware states within a functional logic within a field programmable logic circuit using dynamically reconfigurable test circuits, said method comprising the steps of:

first selecting, by a software trace program, a test mode from a plurality of test modes for tracing the operation of said functional logic;

after said step of selecting a test mode, second selecting one of a plurality of configurations for said dynamically reconfigurable test circuits in conformity with said selected test mode;

configuring said field programmable logic circuit using said selected configuration;

tracing execution of program code to create a software trace history;

periodically reading a set of registers within said dynamically reconfigurable test circuits to create a hardware trace log in response to said tracing step reaching a hardware trace request; and

recording contents of said registers in synchronization with said software trace history to produce a full trace history.

2. A method as claimed in claim 1, wherein said step of first selecting a test mode for tracing the operation of said functional logic further selects a format for output of said hardware trace log.

3. A method as claimed in claim 1, wherein said method further comprises the step of detecting by the trace program, an error associated with said functional logic, and wherein said hardware trace request is made in response to said detected error.

4. A method as claimed in any one of claims 1 to 3, wherein said dynamically reconfigurable test circuits further comprise counters for counting transitions on internal signals within said functional logic, wherein said method further comprises the step of counting said transitions on internal signals, and wherein said step of periodically reading said set of registers further reads values of said counters.

5. A method as claimed in claim 4, wherein said dynamically reconfigurable test circuits further comprise event detectors for detecting events occurring within said functional logic, wherein said method further comprises the step of detecting events occurring within said functional logic, and wherein said step of periodically reading said set of registers further reads values of said counters to determine the number of counts that have occurred between said events.

6. A method as claimed in claim 5, wherein said event detectors are error detectors for detecting errors occurring within said functional logic and wherein said step of detecting events detects said errors.

7. A method as claimed in any one of claims 1 to 6, wherein said method further comprises the step of transmitting said selected configuration into said field programmable logic circuit to configure said dynamically reconfigurable test circuits.

8. A method as claimed in claim 7, wherein said step of transmitting said selected configuration is performed by writing data to said field programmable logic circuit over a peripheral bus.

9. A method as claimed in any one of claims 1 to 8, wherein said method further comprises the step of selecting a plurality of nodes within said functional logic for output to external test pins on said field programmable logic circuit.

10. A system for controlling an electronic device, the system comprising:

a field programmable logic circuit in which functional logic and a plurality of dynamically reconfigurable test circuits are implemented wherein the plurality of test circuits includes a register set;

a processor for executing program instructions to control said electronic device only for instantiating the plurality of test circuits by loading a selected one of a plurality of configurations into said field programmable logic circuit;

a storage for storing the plurality of configurations for configuring said field programmable logic circuit and for storing a test program; and

an interconnect coupling said field programmable logic circuit, said processor and said storage;

wherein the test program includes:

instructions for tracing execution of program code to create a software trace history;

instructions for periodically reading a set of registers within said dynamically reconfigurable test circuits to create a hardware trace log in response to receipt of a hardware trace request; and

instructions for recording content of said registers in synchronization with said software trace history to produce a full trace history.

11. A system as claimed in claim 10, wherein said processor configures said field programmable logic circuit with a test multiplexer having connections to external pins of said field programmable logic circuit for coupling signals within said functional logic to said external pins for measurement by external test equipment.

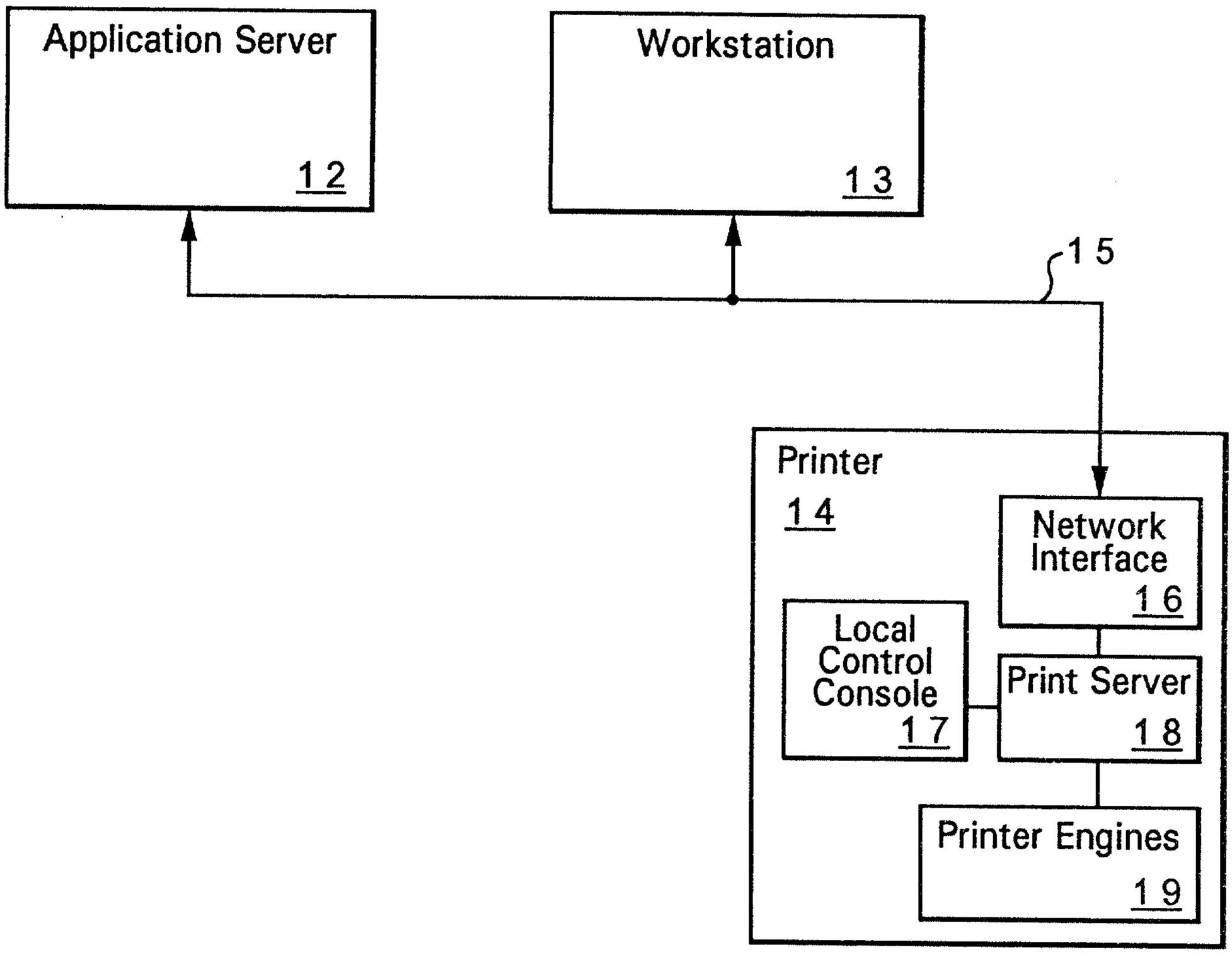

Application Server
12
Workstation
13
15
Printer
14
Network Interface
16
Local Control Console
17
Print Server
18
Printer Engines
19

Fig. 1

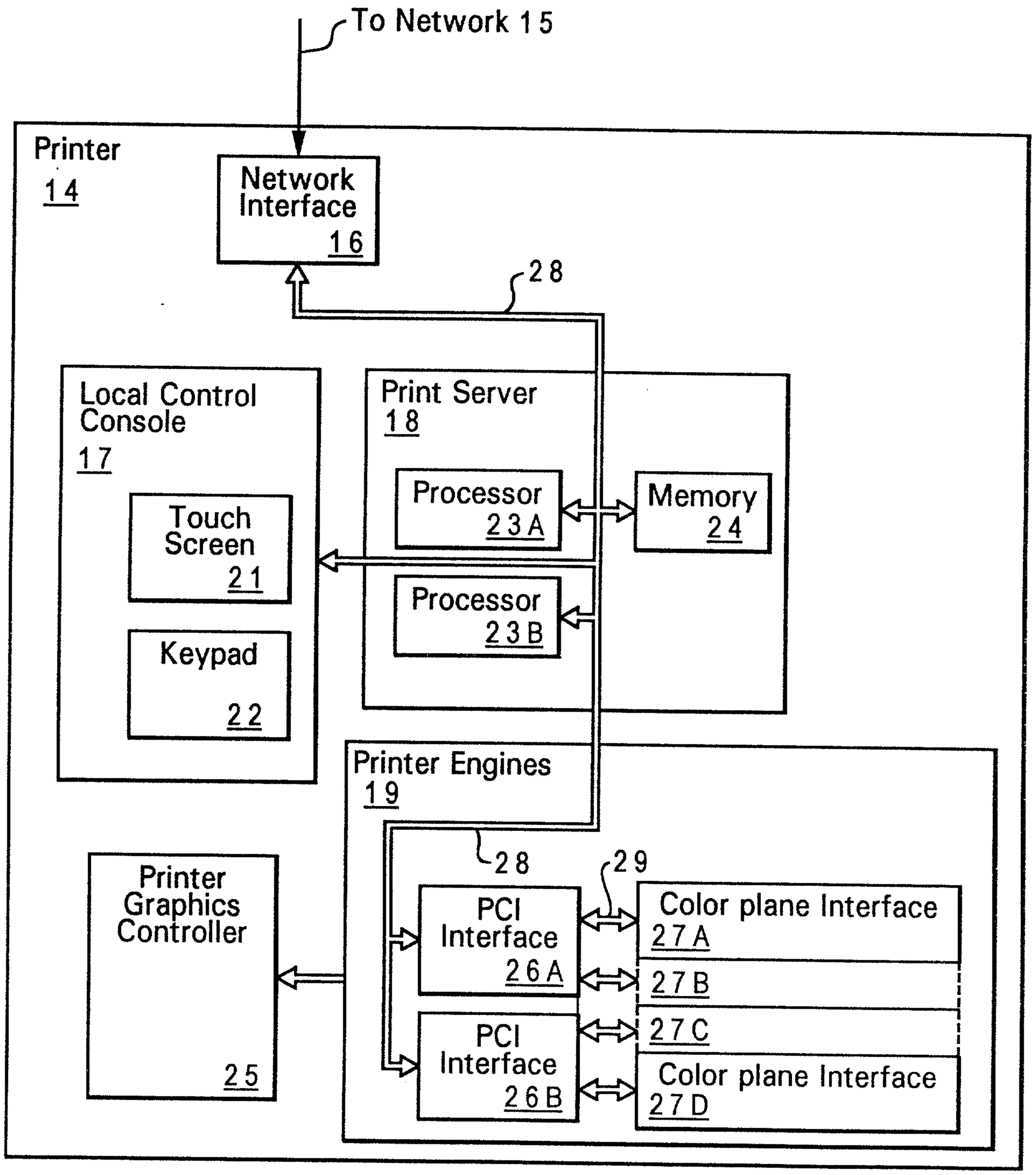

To Network 15
Printer
14
Network
Interface
16
28
Local Control
Console
17
Touch
Screen
21
Keypad
22
Print Server
18
Processor
23A
Memory
24
Processor
23B
Printer Engines
19
Printer
Graphics
Controller
25
28
29
PCI
Interface
26A
PCI
Interface
26B
Color plane Interface
27A
27B
27C
Color plane Interface
27D

Fig. 2

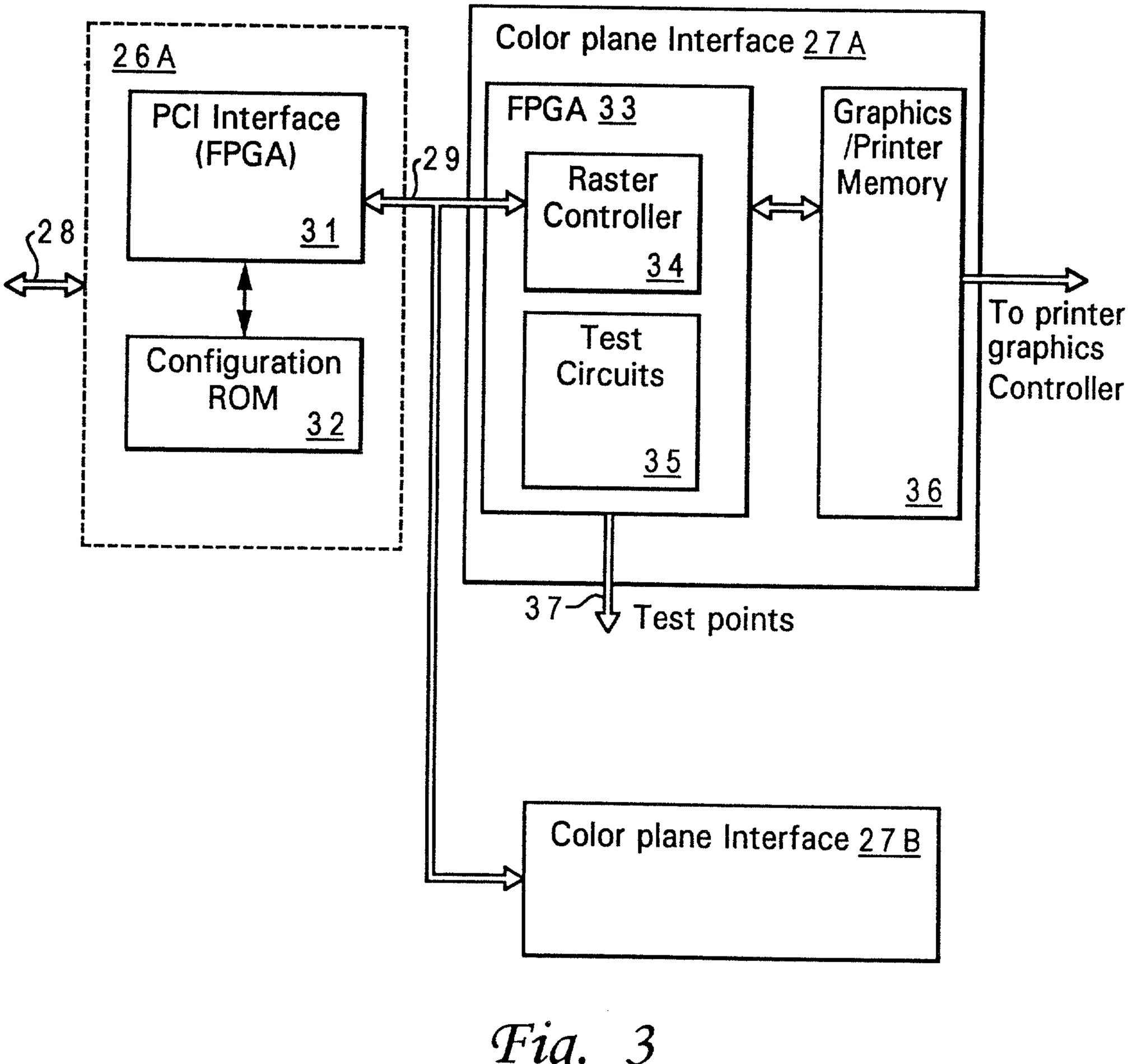

26A
PCI Interface (FPGA)
31
28
Configuration ROM
32
29
Color plane Interface 27A
FPGA 33
Raster Controller
34
Test Circuits
35
Graphics /Printer Memory
36
To printer graphics Controller
37
Test points
Color plane Interface 27B

Fig. 3

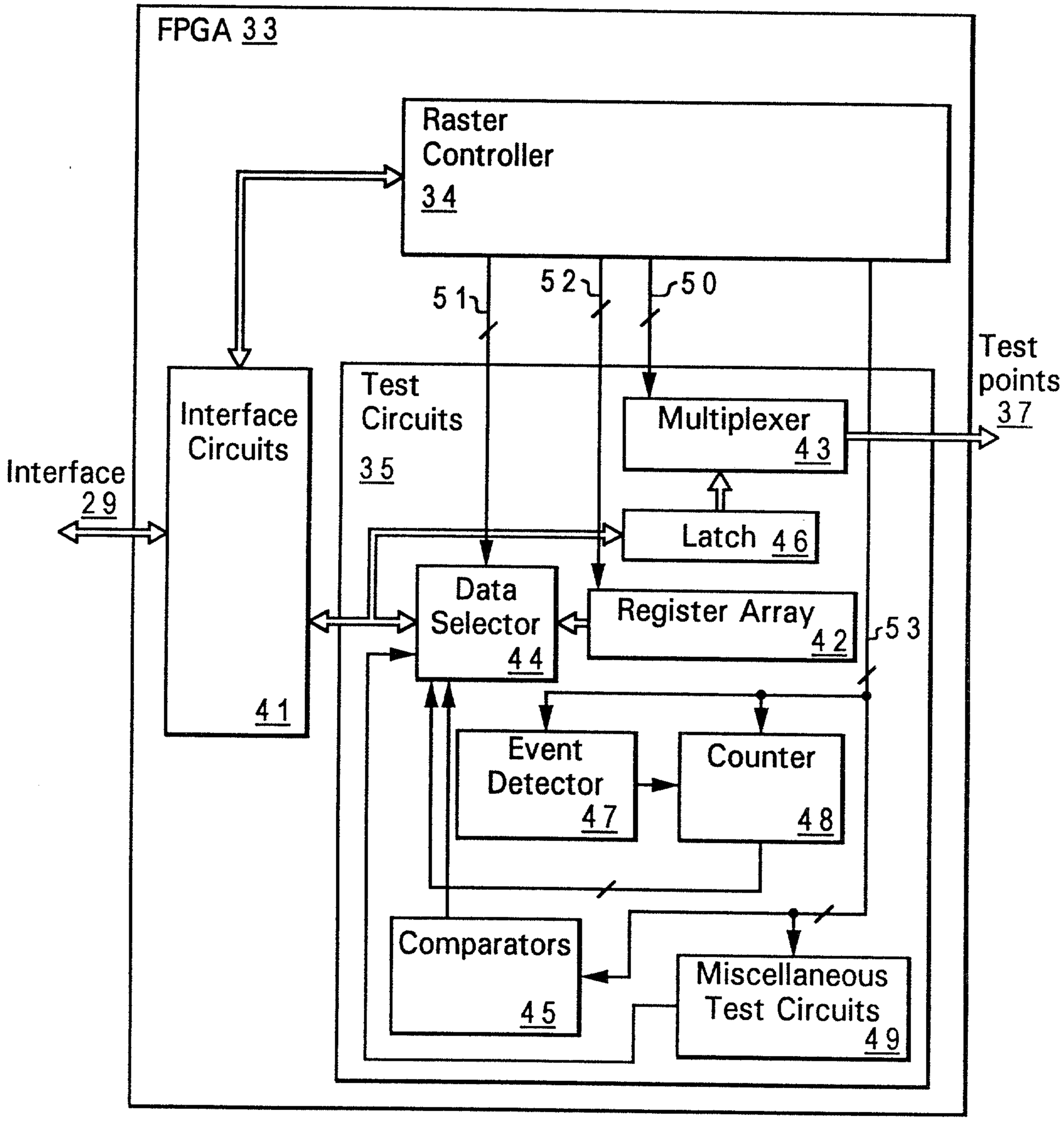

FPGA 33
Raster Controller 34
51
52
50
Test points 37
Interface Circuits 41
Test Circuits 35
Multiplexer 43
Interface 29
Latch 46
Data Selector 44
Register Array 42
53
Event Detector 47
Counter 48
Comparators 45
Miscellaneous Test Circuits 49

Fig. 4

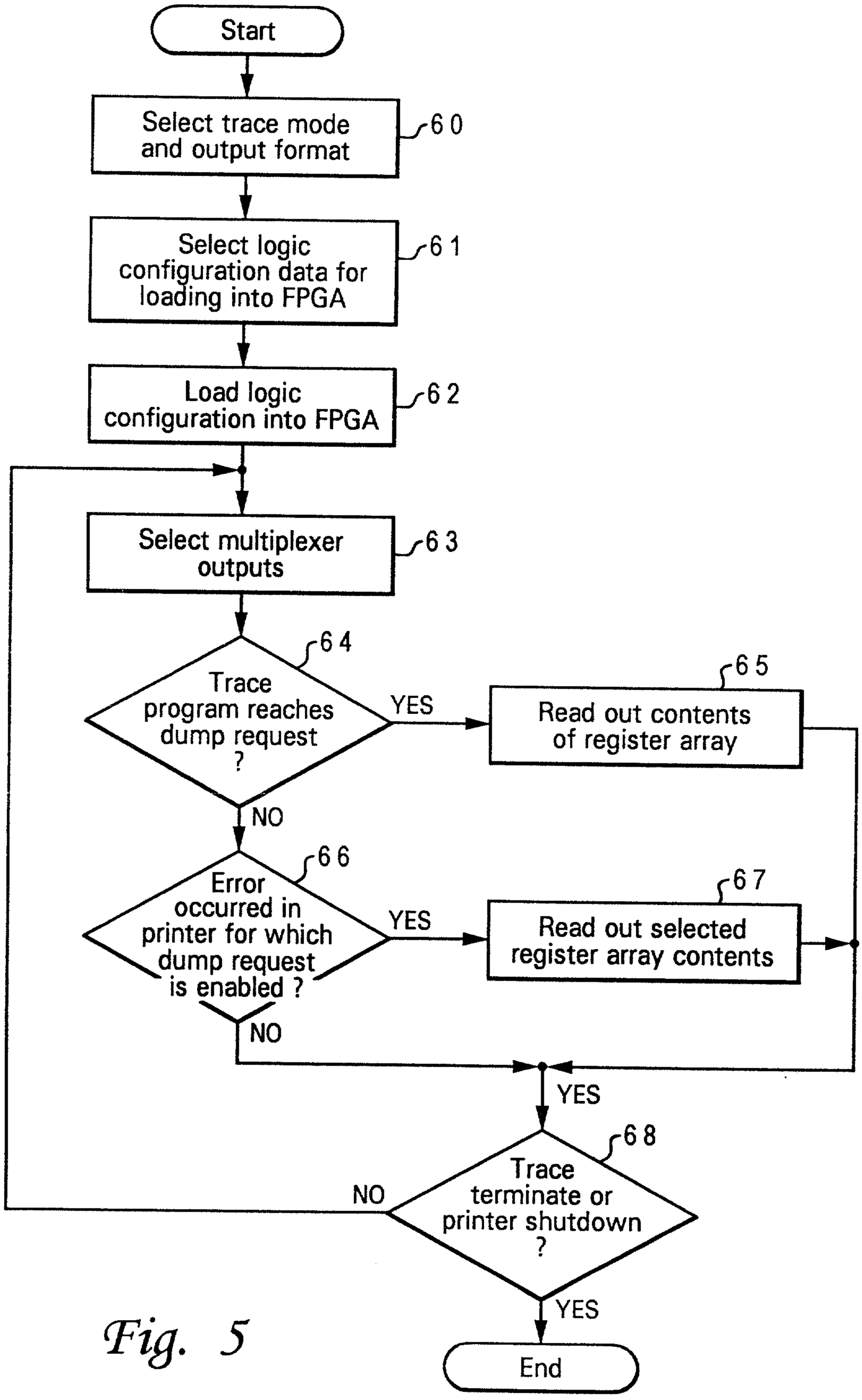
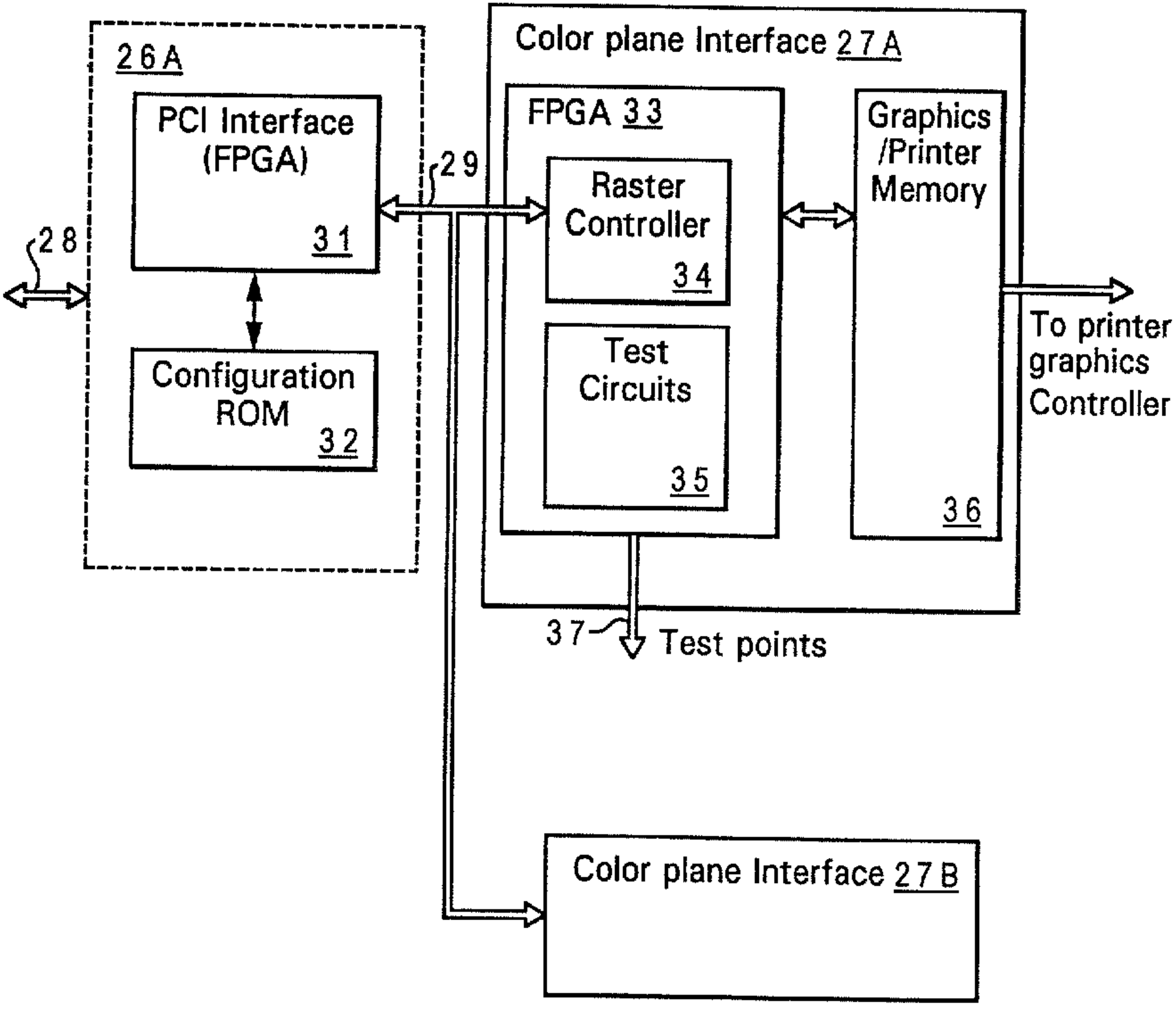

Start
Select trace mode and output format
60
Select logic configuration data for loading into FPGA
61
Load logic configuration into FPGA
62
Select multiplexer outputs
63
64
Trace program reaches dump request ?
YES
65
Read out contents of register array
NO
66
Error occurred in printer for which dump request is enabled ?
YES
67
Read out selected register array contents
NO
YES
68
Trace terminate or printer shutdown ?
NO
YES
End

Fig. 5


26A
PCI Interface (FPGA)
31
28
Configuration ROM
32
29
Color plane Interface 27A
FPGA 33
Raster Controller
34
Test Circuits
35
Graphics /Printer Memory
36
To printer graphics Controller
37
Test points
Color plane Interface 27B