US00RE46021E

(19) **United States**

(12) **Reissued Patent**

Mayer

(10) **Patent Number: US RE46,021 E**

(45) **Date of Reissued Patent: *May 31, 2016**

(54) **SYSTEM-ON-CHIP WITH MASTER/SLAVE DEBUG INTERFACE**

(71) Applicant: **INFINEON TECHNOLOGIES AG**, Neubiberg (DE)

(72) Inventor: **Albrecht Mayer**, Deisenhofen (DE)

(73) Assignee: **Infineon Technologies AG**, Neubiberg (DE)

(*) Notice: This patent is subject to a terminal disclaimer.

(21) Appl. No.: **13/775,962**

(22) Filed: **Feb. 25, 2013**

Related U.S. Patent Documents

Reissue of:

(64) Patent No.: **8,347,158**
Issued: **Jan. 1, 2013**
Appl. No.: **13/528,140**
Filed: **Jun. 20, 2012**

U.S. Applications:

(63) Continuation of application No. 12/913,236, filed on Oct. 27, 2010, now Pat. No. 8,234,531, which is a continuation of application No. 11/954,362, filed on Dec. 12, 2007, now Pat. No. 7,870,455.

(51) **Int. Cl.**

G01R 31/28	(2006.01)
G06F 11/00	(2006.01)
G06F 11/07	(2006.01)
G06F 13/18	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 11/0724*** (2013.01); ***G06F 13/18*** (2013.01)

(58) **Field of Classification Search**

CPC .. G06F 11/0724; G06F 11/2236; G06F 13/18
USPC 714/734, 733, 724, 741, 744, 25, 714/31–33, 35–37, 30, 38, 39; 709/203, 38, 709/39; 710/110, 33

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,539,522	B1	3/2003	Devins et al.	
6,557,119	B1	4/2003	Edwards et al.	
6,715,042	B1	3/2004	Mirza et al.	
6,744,274	B1	6/2004	Arnold et al.	
6,779,145	B1	8/2004	Edwards et al.	
6,857,029	B2	2/2005	Ganasan et al.	
6,895,530	B2	5/2005	Moyer et al.	
7,068,757	B1 *	6/2006	Burnett	379/29.01
7,080,283	B1	7/2006	Songer et al.	
7,107,494	B1	9/2006	Tischler	
7,197,680	B2	3/2007	Kimelman et al.	
7,263,566	B2	8/2007	Ganasan et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

DE 10 2006 016 303 A1 10/2007

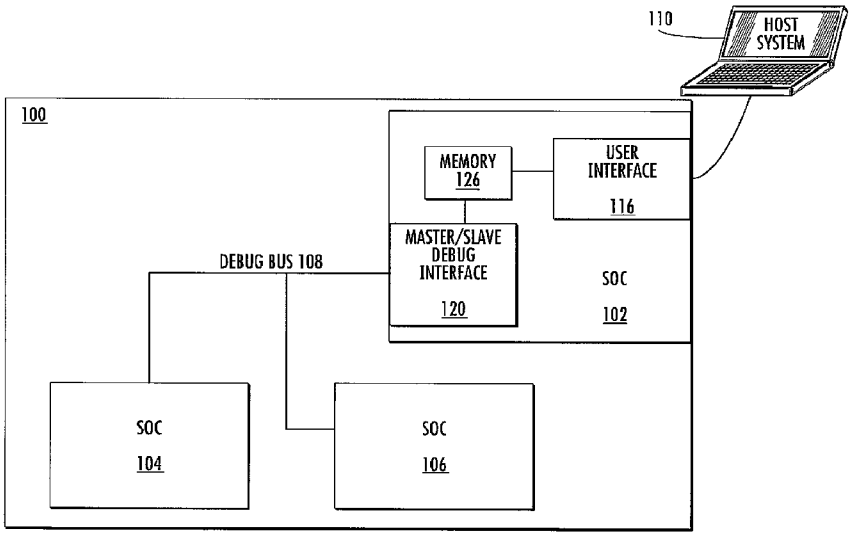
Primary Examiner — Christine Tu

(74) *Attorney, Agent, or Firm* — Schiff Hardin LLP

(57) **ABSTRACT**

A System-on-Chip (SOC) debugging system comprising a plurality of SOCs connected to a shared bus, at least one of the plurality of SOCs being a master SOC and comprising a master/slave debug interface, wherein the master/slave debug interface is a bidirectional debug interface configured to initiate transactions on the shared bus and operable to send and receive debug data between the SOCs, wherein the debug data comprises trace data.

31 Claims, 2 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

7,469,273	B2 *	12/2008	Anderson et al.	709/213
7,475,303	B1	1/2009	Edgar et al.	
7,810,004	B2	10/2010	Mayer et al.	
7,870,455	B2 *	1/2011	Mayer	714/734
7,984,215	B2 *	7/2011	Tsujimoto	710/110
8,234,531	B2 *	7/2012	Mayer	714/734
2003/0217306	A1	11/2003	Harthcock et al.	
2006/0149958	A1	7/2006	Omathuna	
2006/0212768	A1 *	9/2006	Ishida	714/733

* cited by examiner

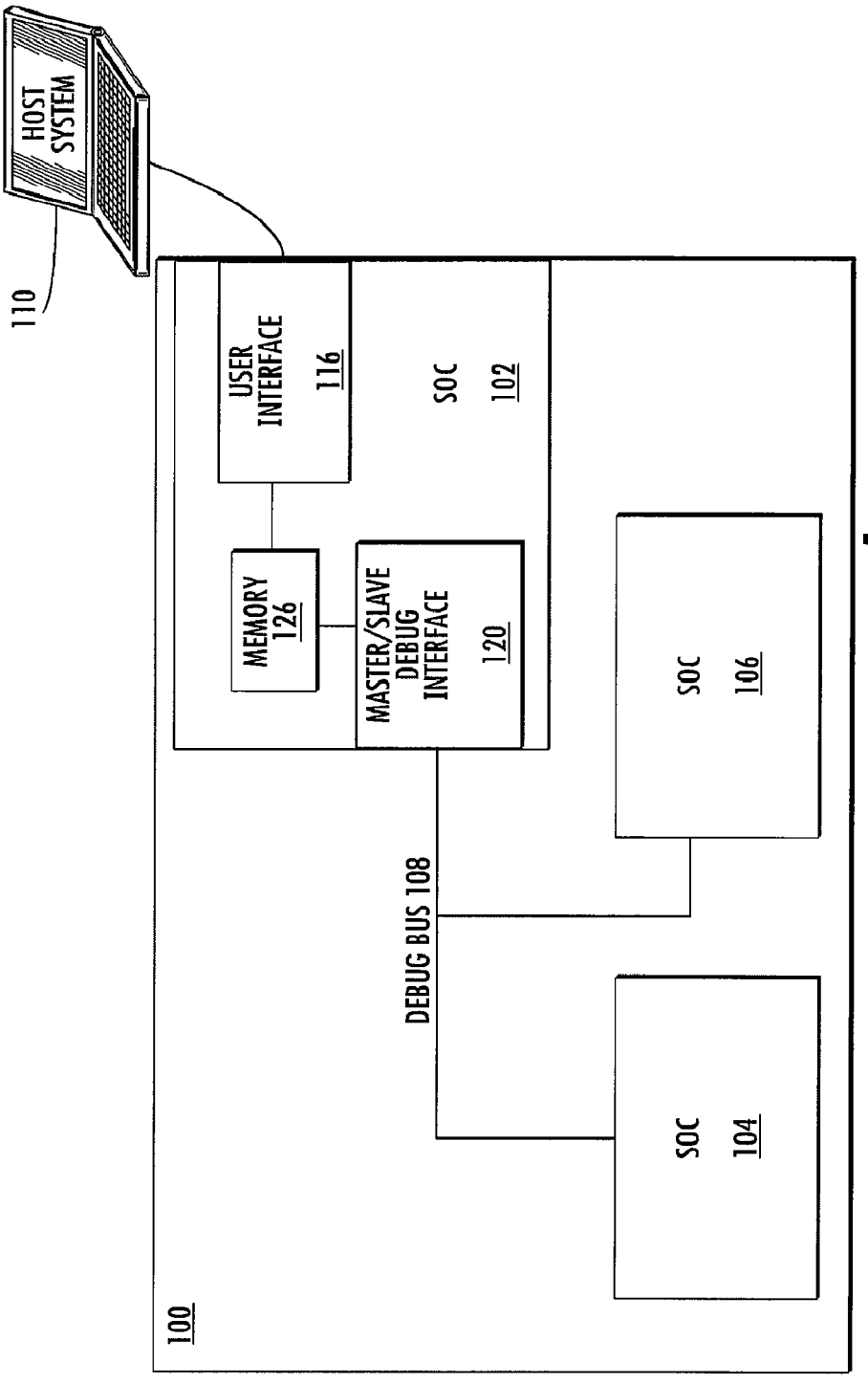

HOST SYSTEM
110
100
USER INTERFACE 116
SOC 102
MEMORY 126
MASTER/SLAVE DEBUG INTERFACE 120
DEBUG BUS 108
SOC 106
SOC 104

FIG. 1

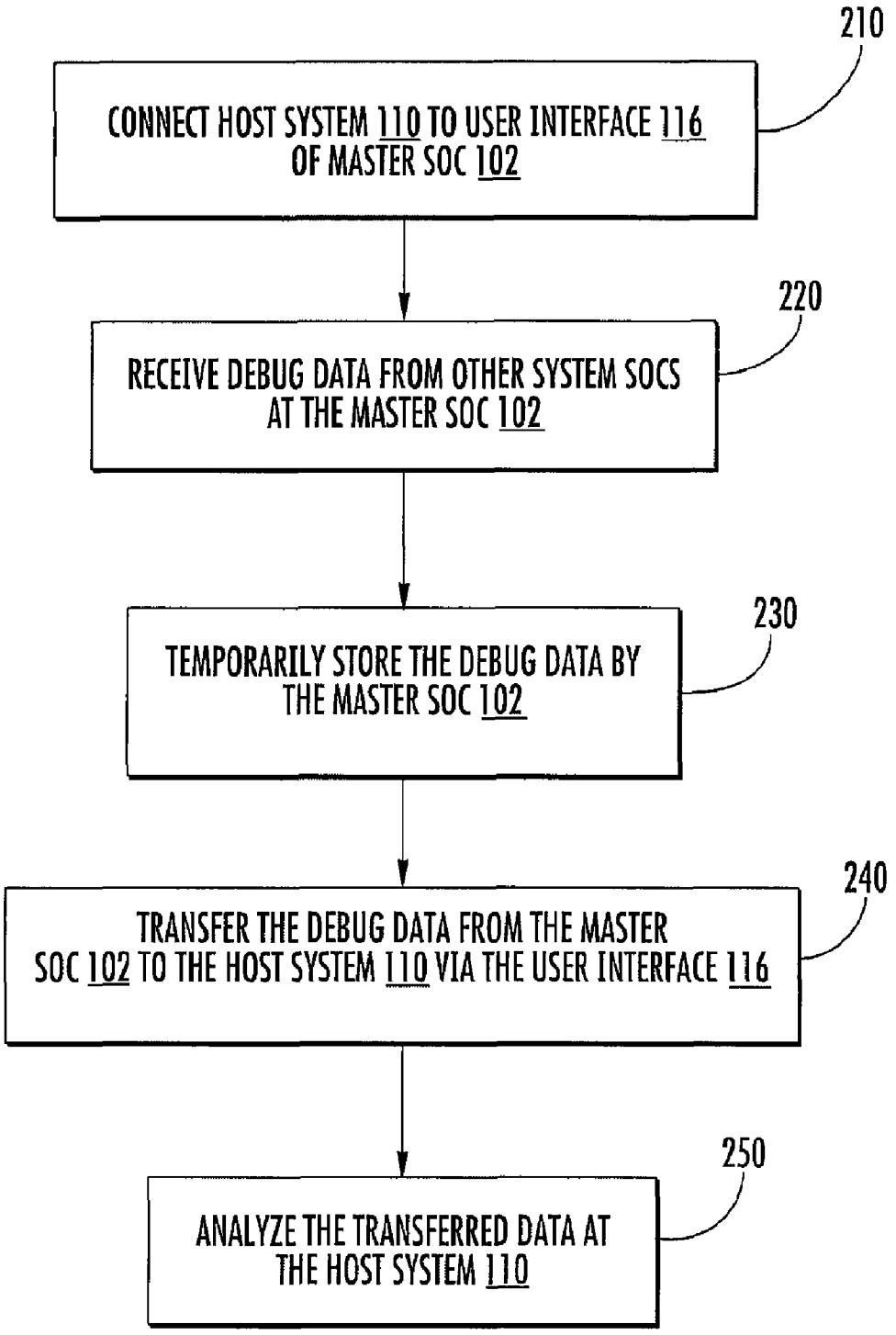

210
CONNECT HOST SYSTEM 110 TO USER INTERFACE 116 OF MASTER SOC 102
220
RECEIVE DEBUG DATA FROM OTHER SYSTEM SOCS AT THE MASTER SOC 102
230
TEMPORARILY STORE THE DEBUG DATA BY THE MASTER SOC 102
240
TRANSFER THE DEBUG DATA FROM THE MASTER SOC 102 TO THE HOST SYSTEM 110 VIA THE USER INTERFACE 116
250
ANALYZE THE TRANSFERRED DATA AT THE HOST SYSTEM 110

FIG. 2

SYSTEM-ON-CHIP WITH MASTER/SLAVE DEBUG INTERFACE

Matter enclosed in heavy brackets [ ] appears in the original patent but forms no part of this reissue specification; matter printed in italics indicates the additions made by reissue; a claim printed with strikethrough indicates that the claim was canceled, disclaimed, or held invalid by a prior post-patent action or proceeding.

RELATED APPLICATION

This *is a reissue application of U.S. applicaton Ser. No. 13/528,140 filed on Jun. 20, 2012, now U.S. Pat. No. 8,347,158, which* is a continuation application of U.S. application Ser. No. 12/913,236 filed on [Oct. 27, 2012,] *Oct. 27, 2010, now U.S. Pat. No. 8,234,531,* which is a continuation application of U.S. application Ser. No. 11/954,362 filed on Dec. 12, 2007, now U.S. Pat. No. 7,870,455.

More than one reissue application has been filed for the reissue of Pat. No. 8,347,158. The reissue applications are the present application and application Ser. No. 15/136,065, filed Apr. 22, 2016.

FIELD OF THE DESCRIPTION

The present description relates generally to data processing and debug systems, and, in particular, to a System-on-Chip (SOC) configuration that captures and transfers debug data directly from multiple system SOCs using an on-chip Master/Slave debug interface.

BACKGROUND

System-on-Chip (SOC) technology operates and controls various types of systems. In general, SOC technology is the assembling of all the necessary electronic circuits and parts for a system (such as a cell phone or digital camera) on a single integrated circuit (IC), generally known as a microchip. SOC devices greatly reduce the size, cost, and power consumption of the system.

During SOC development, debuggers are connected to the debug interfaces (e.g. JTAG port) of the SOCs. To allow synchronous debugging and to save connector pins, all SOCs of the system typically share one debug bus (e.g. CJTAG) and one connector to the debug tool hardware for debugging and testing procedures. Using the debug bus, the system can be debugged (control, status and trace) through a single connector.

Because of the very high degree of integration on a single IC, in many cases, the number of Input/Output (IO) signals off the SOC device are reduced. Furthermore, as chip sizes increase, the number of transistors on a chip increases much faster than the possible number of IO signals off the chip. In many modern chip designs the chip is said to be pad or IO limited, which means that based on the size of the chip, there is insufficient room for all the IO signals that the designers would like, or need, to have routed off the chip. In such environments, in order to lower costs, conserve space, and provide enhanced security, SOCs and Systems in Packages (SiPs) can be configured without debug interfaces making testing and debugging operations problematic. In such cases, analysis is either infeasible or significant effort is needed to solder a debug connector to the board at a time when analysis is needed.

SUMMARY

A System-on-Chip (SOC) debugging system comprising a plurality of SOCs connected to a shared bus, at least one of the plurality of SOCs being a master SOC and comprising a master/slave debug interface, wherein the master/slave debug interface is a bidirectional debug interface configured to initiate transactions on the shared bus and operable to send and receive debug data between the SOCs, wherein the debug data comprises trace data.

These and further objectives, features and advantages will become more apparent from the following description when taken in connection with the accompanying drawings

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates an exemplary SOC IC debugging system in accordance with a preferred embodiment; and

FIG. **2** is a flow diagram illustrating a method for debugging multiple SOC ICs through a user interface of a master SOC IC in accordance with a preferred embodiment.

DETAILED DESCRIPTION

FIGS. **1** and **2**, discussed below, are by way of illustration only and should not be construed in any way to limit the scope of the claims. While described with respect to SOCs, those of skill in the art will understand that the principles may be implemented in any suitably arranged IC or system-in-package device.

Well-known circuits have been shown in block diagram form in order not to obscure the present description in unnecessary detail. Certain details regarding components of the SOCs described herein have been omitted insomuch as such details are not necessary to obtain a complete understanding of the present description and are within the skill of a person of ordinary skill in the relevant art.

FIG. **1** shows a printed circuit board (PCB) system **100** for debugging, testing and monitoring the performance information of several SOCs **102**, **104**, **106**. While not explicitly shown in the figures, SOC **102** can include a processor core, a bus interface, and a system bus for communicating information. SOC **102** may further incorporate a digital signal processing engine, a general purpose microprocessor to provide control functionality, an on-chip memory **126** and a memory controller (not shown) for accessing memory **126**. SOCs **104** and **106** may also include each of the above elements discussed with respect to SOC **102**.

SOCs **102**, **104** and **106** are connected to a shared debug bus **108**. PCB system **100** is connected to host a system **110** directly via SOC **102**. While host system **110** is shown independent of system **100**, for purposes of debugging and testing a plurality of SOCs, connection with host system **110** can be considered an integral part of debug system **100**. The host system **110** can be any type of computer (e.g., personal, mainframe, mini, networked, workstation, etc.) running host software that allows a user to target one or more components on SOCs **102**, **104**, **106** for debugging and to specify triggering parameters for tracing their processing cores.

SOC **102** includes a user interface **116** for connecting and communicating with host system **110**. User interface **116** can be any type of user interface (e.g., serial port, USB, etc.). Host system **110** communicates with SOC **102** through user interface **116** and with the SOCs **104** and **106** through debug bus **108**.

SOC **102** passes debug data, such as trace data, debug control signals and status data, to host system **110** through user interface **116**. SOCs **104** and **106** to be debugged by host system **110** pass their debug data through Master/Slave debug interface **120** of SOC **102**. Accordingly, SOC **102** takes on the role of a "Master" SOC and hereinafter will be referred to as

Master SOC **102**. Likewise, SOCs **104** and **106** act as slaves to Master SOC **102**. The Master/Slave debug interface **120** is configured to initiate transactions on the debug bus **108**. These transactions can include, but are not limited to, instructions to store data in memory, to read data from memory and to transfer data to and from host system **110**.

Master/Slave debug interface **120** is for instance a two-pin bidirectional debug interface as defined in IEEE 1149.7 (CJTAG) consisting of a bi-directional Debug Data pin and a Debug Clock pin. Such an interface allows transferring commands to the device to control the on-chip debug system and to read and write data.

As mentioned briefly above, the debug data which is gathered by monitoring one or more of SOCs **102**, **104**, and **106** can include trace data. A trace is useful when analyzing the behavior, or misbehavior, of an SOC or the SOCs processing core or cores. The trace can show problems in the programming of an SOC processing core and point to errors in the SOC hardware. The trace can be thought of as an external recording of the activity of the SOC that a user can play back with software tools on the host system **110** in order to understand specific internal operations the SOC took and why.

The trace of external IO signals of an SOC can be augmented with other data to give a user additional visibility into SOCs **102**, **104**, **106** internal operations. Bringing selected internal signals of SOCs **102**, **104**, **106** to the outside of the system **100** as additional output signals accomplishes this augmentation.

System **100** includes an arbitrary number of SOCs with each SOC sharing a single debug bus **108** (e.g. CJTAG). With SOCs **102**, **104**, **106** connected to one shared debug bus **108**, a single SOC (i.e. SOC **102**) can take the role of a tool hardware front-end for the debug bus **108**. Referring to FIG. **1**, SOC **102** plays this role and hence can be referred to as "master SOC **102**". On the physical level the direction of all signals is reversed for the master SOC **102** compared to a conventional setup, where all SOCs including the master SOC are accessed from the debug tool over the on-board debug connector. In accordance with a preferred embodiment, the debug interface of the master SOC **102** is operable in two different modes: In a default mode (e.g. reset value) it is a slave and operates like the debug interface of any other SOC controlled from the debug tool over a hardware interface, board connector and debug bus; in the second mode it is a master, which is enabled internally, the signal directions are reversed (e.g. clock output instead of input) and this master controls the slave debug interfaces of all other SOCs. In the later mode, a debug tool need not be attached at the debug connector on the board.

Master SOC **102** acts as a bus bridge between the host system **110**, connected over the user interface **116**, and the debug bus **108**. Thus, the host system **110** accesses the SOCs **104** and **106** indirectly through Master SOC **102**, with reversed direction of its debug interface. Master SOC **102** effectively replaces the need to connect conventional debug tool hardware to the PCB to carry out testing and debug procedures. Instead, Master SOC **102** is equipped with a debug monitor (not shown). The debug monitor is preferably software running on Master SOC **102**.

This debug monitor is for instance a process running on the processor of Mater SOC **102**. It is activated by the operating system. When debug requests arrive at the User Interface **116**, the debug monitor analyzes the requests, schedules transfers over the Debug Bus **108** and sends back the results over the user interface **116**. To limit the impact on the real-time behavior of the system these tasks can be distributed over different Interrupt Service Routines and real time processes. Those of skill in the art will realize that other implementations with less software and more hardware parts of the bus bridge functionality are also possible.

System **100** is functional without restrictions for debug control and status data exchange, which has low to medium latency and bandwidth requirements. Trace data requires much higher bandwidth. If the available bandwidth of the user interface **116** is on average lower than is needed for the trace data, an on-chip trace buffer (not shown) on Master SOC **102** can be used to capture such traces from the other SOCs **104** and **106** for a short period of time. If the available bandwidth of the user interface **116** is on average higher than needed for the trace data, then the full trace can be output over user interface **116**.

FIG. **2** shows a method of debugging system **100** of FIG. **1**. The method begins with connecting a host system **110** to the user interface **116** of the master SOC **102** (step **210**). Next, debug data representing the activity of one or more SOCs **104**, **106** is received by master SOC **102** via debug bus **108** (step **220**). This data can be temporarily stored by master SOC **102** (step **230**). Master SOC **102** transfers the debug data via its user interface **116** to the host system **110** for analysis (step **240**). Finally, the data is received by host system **110** where it can be analyzed by the end user (step **250**).

One skilled in the art will appreciate that additional variations may be made in the above description without departing from the spirit and scope of the description which is defined by the claims which follow.

What is claimed is:

1. A System-on-Chip (SOC) debugging system comprising a plurality of SOCs connected to a shared bus, at least one of the plurality of SOCs being a master SOC and comprising a **[**master/slave**]** *bidirectional* debug interface *operable in a first mode as a slave and in a second mode as a master, and* wherein the **[**master/slave**]** debug interface is **[**a bidirectional debug interface**]** configured to initiate transactions on the shared bus and operable to send and receive debug data between the SOCs, wherein the debug data comprises trace data.

2. The SOC debugging system of claim **1**, wherein the master SOC comprises a user interface.

3. The SOC debugging system of claim **2**, wherein the user interface is a USB interface.

4. The SOC debugging system of claim **2**, wherein the user interface is a network interface.

5. The SOC debugging system of claim **2**, wherein the user interface is a wireless interface.

6. The SOC debugging system of claim **1**, wherein the transactions include instructions to store data in memories of the SOCs, and to read data from the memories of the SOCs.

7. The SOC debugging system of claim **6**, wherein the master SOC comprises a user interface, and the transactions further include instructions to transfer data to and from a host system via the user interface.

8. The SOC debugging system of claim **1**, wherein the **[**master/slave**]** debug interface is a two pin debug interface including one pin for a clock signal and another pin for bidirectional data.

9. The SOC debugging system of claim **1**, wherein the at least one master SOC can be configured to function as a tool hardware front-end for the shared **[**debug**]** bus.

10. The SOC debugging system of claim **1**, wherein the at least one master SOC functions as a bus bridge between a host system and the shared **[**debug**]** bus.

11. The SOC debugging system of claim **[1]** *2*, wherein the at least one master SOC is connected to a host system through

the user interface and accesses the non-master SOCs indirectly through the at least one master SOC.

12. The SOC debugging system of claim **1**, wherein the at least one master SOC includes software acting as a debug monitor.

13. The SOC debugging system of claim **1**, wherein the at least one master SOC includes a trace buffer for capturing traces from the other SOCs.

14. A method for debugging a System-on-Chip (SOC) integrated circuit, the method comprising:

transmitting debug data by a bidirectional debug interface of a master SOC over a shared bus between the master SOC and at least one slave SOC; and

transmitting the debug data by a user interface of the master SOC between the master SOC and an external system coupled to the user interface of the master SOC,

wherein the bidirectional debug interface of the master SOC is operable in a default mode as a slave bidirectional debug interface and in a non-default mode as a master bidirectional debug interface.

15. The method of claim 14, wherein the bidirectional debug interface of the master SOC operates in the non-default mode as the master bidirectional debug interface by controlling a slave debug interface of the at least one slave SOC.

16. The method of claim 14, wherein the bidirectional debug interface of the master SOC operates in the default mode as the slave bidirectional debug interface by being controlled from a debug tool located external to the SOC integrated circuit.

17. The method of claim 14, wherein the user interface of the master SOC is a USB interface.

18. The method of claim 14, wherein the user interface of the master SOC is a network interface.

19. The method of claim 14, wherein the user interface of the master SOC is a wireless interface.

20. The method of claim 14, further comprising initiating instructions on the shared bus by the bidirectional debug interface of the master SOC to store data in a memory of the at least one slave SOC, and to read data from the memory of the at least one SOC.

21. The method of claim 14, wherein the bidirectional debug interface of the master SOC is a two pin debug interface including one pin for a clock signal and another pin for bidirectional data.

22. The method of claim 21, wherein the bidirectional debug interface of the master SOC operates in the non-default mode as the master bidirectional debug interface by controlling a slave debug interface of the at least one slave SOC, and wherein the clock signal is output from the master SOC to the at least one slave SOC via the one pin for the clock signal.

23. The method of claim 14, wherein the master SOC functions as a bus bridge between the external system and the shared bus.

24. The method claim 14, wherein the external system accesses the at least one slave SOC indirectly through the master SOC.

25. The method of claim 14, wherein the master SOC includes a debug monitor.

26. The method of claim 25, wherein the debug monitor performs a method comprising:

analyzing a request received by the user interface of the master SOC from the external system;

scheduling transfer of the debug data over the shared bus from the at least one slave SOC to the bidirectional debug interface of the master SOC; and

transmitting the debug data received from the at least one slave SOC via the user interface of the master SOC to the external system.

27. The method claim 14, further comprising capturing trace data from the at least one slave SOC by a trace buffer of the master SOC, wherein the debug data comprises the trace data.

28. The method of claim 14, further comprising the storing debug data in a memory of the master SOC.

29. The method of claim 27, further comprising:

transferring the trace data from the trace buffer via the user interface of the master SOC to the external system; and

analyzing the trace data by the external system.

30. The method of claim 14, further comprising transmitting the debug data by the bidirectional debug interface of the master SOC over the shared bus between the master SOC and a plurality of slave SOCs.

31. The method of claim 14, wherein the debug data comprises trace data, debug control signals, and status data.

* * * * *