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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 February 2007 (15.02.2007)

PCT

(10) International Publication Number
WO 2007/018553 A1

(51) International Patent Classification:
G06F 13/38 (2006.01) *G06F 3/00* (2006.01)

(21) International Application Number: PCT/US2005/032177

(22) International Filing Date: 8 September 2005 (08.09.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

11/199,577	8 August 2005 (08.08.2005)	US
11/199,562	8 August 2005 (08.08.2005)	US
11/199,564	8 August 2005 (08.08.2005)	US
11/199,576	8 August 2005 (08.08.2005)	US
11/199,567	8 August 2005 (08.08.2005)	US
11/199,372	8 August 2005 (08.08.2005)	US
11/199,560	8 August 2005 (08.08.2005)	US

(71) Applicant *(for all designated States except US)*: **COMMASIC INC.** [US/US]; Suite 301, 16644 West Bernadino Drive, San Diego, California 92127 (US).

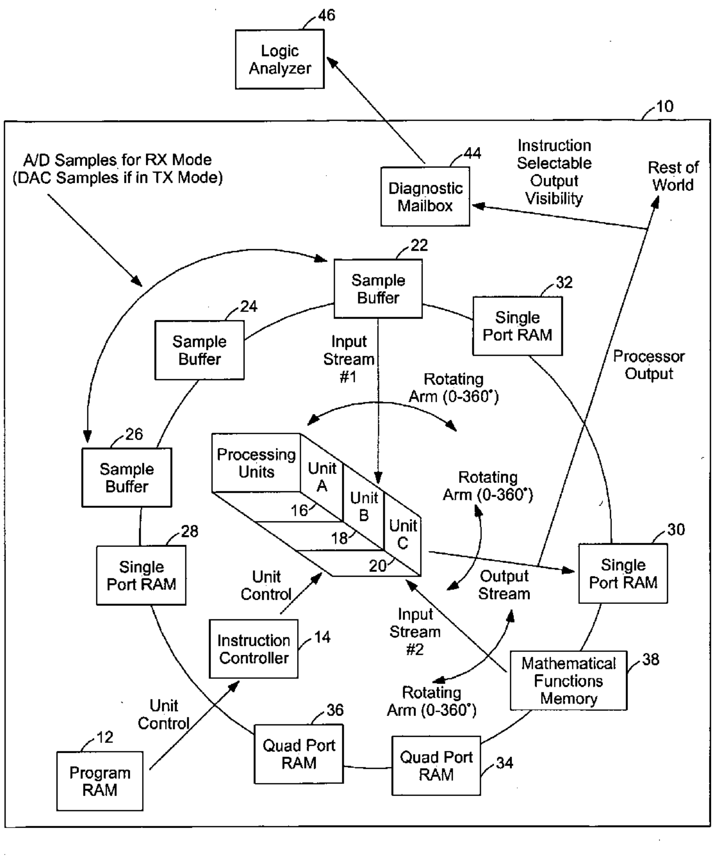
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **MYERS, Theodore John** [US/US]; 1422 Graham Street, San Diego, California 92109 (US). **BOESEL, Robert W.** [US/US]; 11530 Spruce Run Drive, San Diego, California 92131 (US). **NGUYEN, Tien Q.** [US/US]; 10673 Indigo Way, San Diego, California 92127 (US). **SINSUAN, Kenneth Canullas** [US/US]; 1587 Pleasanton Road, Chula Vista, California 91913 (US). **PRICE, Frederick Wales** [US/US]; 2528 El Gavilan Court, Carlsbad, California 92009 (US). **COHEN, Lewis Neal** [US/US]; 16974 Vinaruz Place, San Diego, California 92128 (US). **WERNER, Daniel Thomas** [US/US]; 5871 Torca Court, San Diego, California 92124 (US).

(74) Agent: ALBERT, Peter, G., Jr.; Foley & Lardner LLP, 11250 El Camino Real, Suite 200, San Diego, CA 92130 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI,

[Continued on next page]

(54) Title: MULTI-MODE WIRELESS BROADBAND SIGNAL PROCESSOR SYSTEM AND METHOD



(57) Abstract: A wireless broadband signal processing system includes a program memory, an instruction controller, and processing units. The system can also include sample buffers; single port memories; and quad port memories. The program memory stores programmed instructions used by the instruction controller. The processing units are configured to perform vector processes, such as demodulation processes. One processing unit can be configured for a convolution operation calculated each clock, another processing unit can be configured for FFT functionality where a Radix-4 butterfly is performed each clock, and yet another processing unit can be configured for other vector operations, such as de-spreading, vector addition, vector subtraction, dot product, and component-by-component multiplication. The system can collect diagnostic data, operate across multiple networks, reduces baseband circuitry, and maximizes multi-mode operations.

GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

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

Published:
— *with international search report*

MULTI-MODE WIRELESS BROADBAND SIGNAL PROCESSOR SYSTEM AND METHOD

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0001] The present invention is related to communication systems and methods. More particularly, the present invention relates to a multi-mode wireless broadband signal processor system and method.

DESCRIPTION OF THE RELATED ART

[0002] This section is intended to provide a background or context. The description herein may include concepts that could be pursued, but are not necessarily ones that have been previously conceived or pursued. Therefore, unless otherwise indicated herein, what is described in this section is not prior art to the claims in this application and is not admitted to be prior art by inclusion in this section.

[0003] Wireless devices continue to need the capability to handle increasingly high data rates. To accommodate multimedia content, for example, data rates for wireless devices may need to match broadband rates for hard-wired devices. Wireless device users increasingly demand multifunction, multi-technology devices to obtain different types of content and services via multiple wireless networking technologies.

[0004] Many attempts have been made to build broadband capability into small, handheld devices. For example, wireless data technology commonly known as Wi-Fi 802.11 provides high-speed capability to handle such demanding applications as high quality (high definition) streaming video and image content. However, conventional 802.11 implementations fail to meet user-acceptable power consumption parameters. Even the lowest power-consuming 802.11 implementations currently available severely limit "talk time" (active state during which voice, data, or video is being transferred) for battery operated devices.

[0005] Beyond devising an 802.11 implementation with acceptable power consumption, another challenge is to establish a wireless implementation that supports two or more networking modes of operation, such as 802.11, Bluetooth, Ultra Wideband (UWB), WiMax (802.16d and 802.16e), 802.20, and 3G and 4G cellular systems. Wireless devices need to be able to offer a variety of wireless networking technologies. The ability to operate according to multiple networking standards and technologies in a single device is referred to as "multi-mode" capability.

[0006] Most conventional mobile devices are either digital signal processor (DSP)-based, application specific integrated circuit (ASIC)-based, or an ASIC/DSP hybrid architecture. Several engineering considerations, such as power efficiency, design flexibility and cost, prevent either approach from being suitable for broadband wireless. Because of architectural limitations, conventional approaches may be able to provide high data rates, but only at the expense of power consumption, resulting in an unacceptably short battery life.

[0007] With new wireless standards being introduced everyday, traditional ASIC design is too inflexible to continually accommodate these rapidly evolving standards. Once the integrated circuit design cycle begins for a new standard, modifications that inevitably occur necessitate re-starting from scratch or re-spinning the ASIC chip. To provide the multiple wireless capabilities end users demand on a single device, ASIC and DSP approaches support multi-mode capability by simply stacking additional "processing circuitry" in parallel, significantly increasing device volume and manufacturer costs for each new mode.

[0008] There is a need for a communication system and architecture that provides for multi-mode communication with broadband performance and low power consumption. There is also a need for the ability to collect high diagnostic data of a communication device at a high frequency for observation and analysis with a logic analyzer. Further, there is a need to provide wireless communication devices that can function across multiple networks and multiple communication standards. Even further, there is a need to reduce baseband circuitry and improve ASIC algorithms to achieve ultra low power/cost advantage, resulting in performance processing gains and reductions in power consumption, gate count and silicon cost.

[0009] There is also a need for controlling input and output in such a communication system and a need for dynamically controlling rate connections to sample buffers in a multi-mode wireless processing system for handling multiple communication standards. Further, there is a need for interfacing with a processor in a multi-mode wireless processing system. Even further, there is a need for performing fast fourier transforms (FFTs) in a manner that minimizes power consumption. Even further, there is a need for the ability to prioritize the execution of tasks and their component operations based upon the context of the tasks. There is also a need for performing a convolution operation in a multi-mode wireless broadband system.

SUMMARY OF THE INVENTION

[0010] One exemplary embodiment relates to a method of obtaining processor diagnostic data. The method can include receiving a instruction, enabling write access of an output communication stream to a diagnostic memory, writing to the diagnostic memory at a first rate, and reading from the diagnostic memory at a second rate where the first rate is greater than the second rate.

[0011] Another exemplary embodiment relates to a system for obtaining processor diagnostic data. The system can include a memory containing instructions, a controller that receives and executes the instructions, and a diagnostic memory that receives communication data at a first rate and outputs the communication data at a second rate where the first rate is higher than the second rate.

[0012] Another exemplary embodiment relates to a method of controlling input and output in a multi-mode wireless processing system. The method can include receiving an instruction for communication in a multi-mode wireless processing system, and determining from a field in the received instruction whether a designated processing unit generates output data or receives input data.

[0013] Another exemplary embodiment relates to a configuration of input / output components for interfacing with a processing unit in a multi-mode wireless processing system. The configuration includes a plurality of general purpose inputs for supplying input data to a processing unit in a multi-mode wireless processing system; and a plurality of general purpose outputs for receiving output data generated by the processing unit in the multi-mode wireless processing system.

[0014] Another exemplary embodiment relates to a system for controlling input and output in a multi-mode wireless processing system. The system can include a memory including instructions in a multi-mode wireless processor system; and a controller that receives the instructions and determines from an instruction field whether a designated processing unit in the multi-mode wireless processing system generates output data or receives input data.

[0015] Another exemplary embodiment relates to a method of dynamically controlling rate connections to sample buffers in a multi-mode processing system. The method can include receiving an instruction for communication in a multi-mode wireless processing system and determining a rate at which a plurality of buffers are serially connected to elements external to the multi-mode wireless processing system for receipt or transmission of data.

[0016] Another exemplary embodiment relates to a system for dynamically controlling rate connections to sample buffers in a multi-mode processing system. The system can include a memory including instructions for multi-mode wireless processor communication in a multi-mode wireless processing system and a controller that receives instructions and determines a rate at which a plurality of buffers are serially connected to elements external to the multi-mode wireless processing system for receipt or transmission of data.

[0017] Another exemplary embodiment relates to a method of interfacing two processors. The method can include generating a read/write request at a first processor for accessing a memory that is not directly accessible to the first processor, receiving the read/write request at a second processor that has direct access to the targeted memory, completing a read/write operation at the second processor; and receiving at the first processor an indication that the read/write operation has been completed.

[0018] Another exemplary embodiment relates to a system for interfacing two processors. The system can include a first processor that generates a read/write request for accessing a memory that is not directly accessible to the first processor, a second processor that receives the read/write request, has direct access to the targeted memory, and completes a read/write operation, a target memory, and a means for communicating between the first processor and a second processor.

[0019] Another exemplary embodiment relates to an interface between two processors. The interface can include a means for generating a read/write request at a first processor, a means for setting status bits by either processor, a means for polling the status bits by both processors, and a means for communicating additional data between the two processors.

[0020] Another exemplary embodiment relates to a method of performing a fast fourier transform (FFT) in a multi-mode wireless processing system. The method can include loading an input vector into an input buffer, initializing a second counter and a variable N, where $N = \log_2$ (input vector size), and s is the value of the second counter, performing an FFT stage, and comparing s to N and performing additional FFT stages until $s = N$. The FFT stage can include performing vector operations on data in the input buffer, sending the results to an output buffer, advancing the value of the second counter, and switching roles of the input and output buffers. The vector operations in an FFT stage can include performing Radix-4 FFT vector operations on four input data at a time and multiplying the resulting output vectors with a Twiddle factor. The method of generating a Twiddle factor can include generating a control word for controlling manipulation of a Twiddle factor and determining whether a Twiddle factor needs to be accessed from a memory based upon the generated Twiddle address. If a Twiddle factor needs to be accessed, the method of generating a Twiddle factor can further include reading the Twiddle factor from the memory, manipulating the Twiddle factor based upon the control word, and storing a manipulated Twiddle factor in the processing unit.

[0021] Another exemplary embodiment relates to a system for performing a fast fourier transform (FFT) in a multi-mode wireless processing system. The system can include a memory for providing mathematical functions to the processing unit, a program memory containing instructions for executing an FFT algorithm, an instruction controller for receiving and executing instructions from the program memory, and a pair of buffers that alternate between acting as an input buffer and an output buffer in successive FFT stages of the FFT algorithm.

[0022] The processing unit in this exemplary embodiment can include a Radix-4 FFT engine that performs eight complex additions on four input vectors and generates four output vectors, a Twiddle multiplier for multiplying a generated output vector with an associated Twiddle factor, a serial-to-parallel converter for receiving the four input vectors serially from the input buffer and sending the four input vectors to the Radix-4 FFT engine in parallel, a parallel-to-serial converter for receiving the four generated output vectors in parallel and delivering the four output vectors serially to the Twiddle multiplier and output buffer, a set of registers for storing manipulated Twiddle factors in the processing unit, a Twiddle octant manipulator that manipulates Twiddle factors based upon a control word, a master counter used as a loop variable for monitoring progress of the FFT algorithm in a given FFT stage, a second counter used as a loop variable for keeping track of a current stage of the FFT algorithm, an input address generator that generates an input buffer address, the input buffer address being used as an output buffer address for all FFT stages except for when a last FFT stage is being performed and N is odd, where N = log2 (size of data in the input buffer), a Twiddle address generator for generating a preliminary Twiddle address, a

DiBit interleaving generator that generates the output buffer address for the last FFT stage if N is odd, and a Twiddle address multiplier for generating the control word and a final Twiddle factor address.

[0023] Another exemplary embodiment relates to a system for obtaining processor diagnostic data. The system can include a memory containing instructions, a controller that receives and executes the instructions, and a diagnostic memory that receives communication data at a first rate and outputs the communication data at a second rate where the first rate is higher than the second rate.

[0024] Another exemplary embodiment relates to a system for obtaining processor diagnostic data. The system can include a controller that receives instructions from a program memory and a diagnostic memory that is enabled to receive data by the controller based on the received instructions. The diagnostic memory receives the data at a first rate and outputs the data at a second rate where the first rate is higher than the second rate. The system further can include an external interface coupled to the diagnostic memory for communicating the data at the second rate.

[0025] Another exemplary embodiment relates to a method of switching between instruction contexts within a time interval. The method can include executing critical task operations that complete execution within a time interval, a critical task including a plurality of critical task operations, executing non-critical task operations that are able to cross a time interval boundary, a non-critical task including a plurality of non-critical task operations, and entering a sleep mode in which no critical task operations or non-critical task operations are executed, if the critical task operations and the non-critical task operations begun in the time interval have been completed before a following time interval begins.

[0026] Another exemplary embodiment relates to a method for performing a convolution operation in a multi-mode wireless processing system. The method can include loading an initial value and a stride value into an address generator, generating an address based on the initial value and the stride value, supplying the generated address to a series of memories, loading input data into a series of registers, multiplying the contents of each register with a value stored at the generated address in the memory associated with each register, adding up the resulting multiplication products, and generating output based on the resulting sum. The number of memories and registers are equal, each register having an associated memory.

[0027] Another exemplary embodiment relates to a system for performing a convolution operation in a multi-mode wireless processing system. The system can include an address generator for generating an address given an initial value and a stride value, a series of memories, a series of registers for storing an input value, a series of complex multipliers, the series of multipliers, registers, and memories being equal in number, each multiplier being associated with one register and one memory, each multiplier generating a product of contents of the associated register and a value stored at the generated address in the associated memory; and a complex adder tree for adding the series of products and producing a product sum.

[0028] Other exemplary embodiments are also contemplated, as described herein and set out more precisely in the appended claims.

BRIEF DESCRIPTION OF DRAWINGS

[0029] Fig. 1 is a diagram depicting a wireless broadband signal processing system in accordance with an exemplary embodiment.

[0030] Fig. 2 is a diagram depicting use of a diagnostic mailbox in the wireless broadband signal processing system of Fig. 1 in accordance with an exemplary embodiment.

[0031] Fig. 3 is a diagram depicting a mailbox diagnostic functionality implemented via a dual-port RAM in accordance with an exemplary embodiment.

[0032] Fig. 4 is a diagram of the processing by the wireless broadband signal processing system of Fig. 1 of an instruction including a general purpose input output (GPIO) instruction field in accordance with an exemplary embodiment.

[0033] Fig. 5 is a diagram of the wireless broadband signal processing system of Fig. 1 depicting general purpose input and output operations.

[0034] Fig. 6 is a diagram of the wireless broadband signal processing system of Fig. 1 depicting a dynamic configuration of a processing iteration duration.

[0035] Fig. 7 is a diagram depicting operations performed by an ARM processor and a wireless broadband signal processor (WBSP) processor utilized in the wireless broadband signal processing system of Fig. 1 in accordance with an exemplary embodiment.

[0036] Fig. 8 is a diagram depicting FFT operations performed in the wireless broadband signal processing system of Fig. 1 in accordance with an exemplary embodiment.

[0037] Fig. 9 is a diagram depicting functionalities of a processor performing an FFT algorithm in the wireless broadband signal processing system of Fig. 1.

[0038] Fig. 10 is a diagram depicting operations performed in an address generation process for the FFT algorithm of Fig. 9.

[0039] Fig. 11 is a diagram depicting an exemplary input address mapping in accordance with an exemplary embodiment.,

[0040] Fig. 12 is a diagram depicting an exemplary Twiddle address mapping in accordance with an exemplary embodiment.

[0041] Fig. 13 is a diagram depicting interleaving mappings for a last stage process in accordance with an exemplary embodiment.

[0042] Fig. 14 is a diagram depicting a context switching operation in accordance with an exemplary embodiment.

[0043] Fig. 15 is a diagram timing of the context switching operation of Fig. 14.

[0044] Fig. 16 illustrates a processing unit in the wireless broadband signal processing system of Fig. 1.

[0045] Fig. 17 illustrates address operation logic from the processing unit of Fig. 16.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

[0046] Fig. 1 illustrates a wireless broadband signal processing system 10. The wireless broadband signal processing system 10 can include a program memory 12, an instruction controller 14, and processing units 16, 18, and 20. The system 10 can also include sample buffers 22, 24, and 26; single port memories 28, 30, and 32; and quad port memories 34 and 36. The program memory 12 stores programmed instructions used by the instruction controller 14. The processing units 16, 18, and 20 are configured to perform vector processes, such as demodulation processes. For example, the processing unit 16 can be configured for a convolution operation calculated each clock, the processing unit 18 can be configured for FFT functionality where a Radix-4 butterfly is performed each clock, and the processing unit 20 can be configured for other vector operations, such as de-spreading, vector addition, vector subtraction, dot product, and component-by-component multiplication. Additional, fewer, or different processing units can be included. In at least one exemplary embodiment, a memory 38 is included to provide mathematical functions to the processing units 16, 18, and 20. The memory 38 can be a read only memory (ROM).

[0047] The instruction controller 14 receives vector instructions from the program memory 12. Based on the received vector instruction, the instruction controller 14 can select port memories for input and output. Exemplary operations of the wireless broadband signal processing system 10 are described in U.S. Patent Application No. 10/613,476 entitled "Multi-Mode Method and Apparatus for Performing Digital Modulation and Demodulation" which is herein incorporated by reference in its entirety.

[0048] The wireless broadband signal processing system 10 further includes a diagnostic mailbox 44. The diagnostic mailbox 44 is a memory, such as a random access memory (RAM), coupled to the output of the processing units (as shown) or the input of the wireless broadband signal processing system 10. In either implementation, the diagnostic mailbox 44 receives communication data at a high frequency and transmits the communication data at a lower frequency to a logic analyzer 46 which creates a log of the contents of the diagnostic mailbox 44. The contents of the diagnostic mailbox 44 can then be reviewed and studied for an understanding of the operations of the wireless broadband signal processing system 10, performing debug operations or failure analysis, etc.

[0049] Fig. 2 illustrates the use of the diagnostic mailbox 44 according to an exemplary embodiment. In operation, the instruction controller 14 receives an instruction from the program memory 12. The instruction contains diagnostic mailbox fields with information on the type of instruction being communicated. The diagnostic mailbox field is set to a logical one (1) if the output stream is to be written to the diagnostic mailbox 44. The instruction controller 14 performs the necessary time alignment such that the diagnostic mailbox 44 is enabled for write access for the duration of the vector instruction output. The rate at which the write to the diagnostic mailbox 44 occurs is F_{wbsp}. The read operation from the diagnostic mailbox 44 occurs at a lower synchronous rate of F_{read} which is a rate supportable for off-chip access. In an exemplary embodiment, the synchronous rate of F_{read} is 40 MHz or less and is a factor of 5-10 lower than F_{wbsp}, which is 40 MHz or more. $F_{read} \geq NF_{wbsp}$ where N is the fraction of clocks which are associated with instructions whose diagnostic mailbox field is set to 1.

[0050] In an alternative embodiment, the instruction controller 14 enables write access to the diagnostic memory whenever the vector instruction received from the program memory 12 changes. This allows for the diagnostic mailbox 44 to provide a continual log of the output stream.

[0051] Fig. 3 illustrates a preferred embodiment in which the diagnostic mailbox is implemented via a dual-port RAM 54. Logic external to the dual port

RAM 54 (not shown) increments the read and write addresses sequentially after each access – with the exception that a wrap to 0 occurs when the address value exceeds the physical size of the RAM (e.g. The address sequence would be N-3, N-2, N-1, 0, 1, 2, ... where N is the number of accessible locations in the dual-port RAM 54). The dual port RAM 54 thus acts as a FIFO.

[0052] The write port of the dual port RAM 54 is enabled when the output of an instruction associated with a diagnostic-enabled instruction is generated. The read port of the dual port RAM 54 operates at a lower frequency than the write port. When A_write, the write address, is greater than A_read, the read address, the dual-port has valid information which is clocked out of the read port until A_write = A_read. If A_write becomes too large such that information is written over which has not been clocked out of the read port, an overflow indicator is set and latched which indicates an error condition.

[0053] In an exemplary embodiment, mailbox supporting logic 53 includes instructions that aid the dual-port RAM 54 in carrying out its operations. The mailbox supporting logic 53 receives write addresses and read addresses. Depending on this information, the mailbox supporting logic 53 can communicate an overflow indicator, which, as explained above, indicates that information is being written over in the dual-port RAM 54 (the diagnostic mailbox 44 is full). An empty indicator can be communicated to indicate that the dual-port RAM 54 is ready to receive data (the diagnostic mailbox 44 is empty). The mailbox supporting logic 53 communicates a read enable signal to the dual-port RAM 54 when the RAM data is to be communicated out via a diagnostic stream to the logic analyzer 46.

[0054] Fig. 4 illustrates the processing by the instruction controller 14 of an instruction received from the program memory 12 including a general purpose input output (GPIO) instruction field. A GPIO instruction field having N bits can indicate a GPI (General Purpose Input), GPO (General Purpose Output), or neither with a GPIO code of zero. An N-bit field can address up to a combination of 2^N-1 GPIs and GPOs. The GPIO code can trigger the instruction controller 14 to use GPI selection logic 55 or GPO selection logic 57.

[0055] A general purpose output (GPO) operation can be used to control communications to elements external to a wireless broadband signal processor (WBSP) utilized in the wireless broadband signal processing system 10. Examples of external elements include processors (such as the processor known as an ARM processor from ARM, Limited of Cambridge, England,) or RF transceivers. Additionally, registers associated with operation of the WBSP can be accessed using GPO operations, such as the PID register discussed below. When the GPIO code that is unique to an element is in the current instruction in program memory 12, the GPO selection logic 57 pulses an enable that is wired directly and uniquely to the element. The significance of the particular enable may vary depending on the element. Typically, the enable signals cause the element to latch the data on the output stream. Alternatively, an enable has significance in itself and allows the output stream to be sent directly to the element without being latched.

[0056] A general purpose input (GPI) operation can be used to receive input from elements external to the WBSP or from registers associated with operation of the WBSP. Examples of input operations include supporting the interface between the WBSP and an external processor (such as an ARM), recording the rate of frame errors. If the code asserted in the GPIO field of the instruction corresponds to a GPI, then the input stream is hooked into that particular element.

[0057] Fig. 5 illustrates the wireless broadband signal processing system 10 including the processing of an instruction having a general purpose input output (GPIO) instruction field. In one input or GPI operation, the sample buffer 22 communicates an input stream of communication data to one of the processing units 16, 18, and 20. In another input or GPI operation, an element 66 communicates an input stream of communication data to one of the processing units 16, 18, and 20.

[0058] Fig. 6 illustrates an exemplary dynamic configuration of a processing iteration duration (PID). The PID refers to the number of samples that are either written into the sample buffers 22, 24, and 26 in receive mode (from A/D) or read out of the sample buffers 22, 24, and 26 in transmit mode (to a DAC). Exemplary buffer techniques that can be utilized in the wireless broadband signal processing system 10 are described in U.S. Patent Application No. 10/613,897 entitled "Buffering Method and Apparatus for Processing Digital Communication Signals," which is herein incorporated by reference in its entirety.

[0059] The PID—the number of samples written into the sample buffers 22, 24, and 26—determines the rate at which the buffer scheme is advanced. In other terms, the PID is the program rate at which the sample buffers 22, 24, and 26 are connected to receive samples. A small PID represents a low latency situation in that the samples are available (on RX) or are made available (on TX) in a small amount of time; a larger PID allows for greater processing efficiency in that longer vector operations are allowed which is inherently more efficient (initial processing latencies for an instruction are amortized across more output data).

[0060] The parameters that determine the rate of the advance of the sample buffers 22, 24, and 26 is accessible via a GPIO instruction. When the GPIO field in the current instruction contains the value of 1, the output stream is routed to the register that controls the rate at which the sample buffers are advanced. As such, the ability of the instruction controller 14 to dynamically alter the PID allows for real-time tradeoffs between low and high latency. For example, a longer PID can be used when longer vector operations are in execution or anticipated to be executed. Additionally, some PIDs are inherently superior for standards that have a specific symbol rate (e.g., 4 microsecs is a natural fit for 802.11g).

[0061] Fig. 7 illustrates operations performed by a processor, such as the ARM processor, and a wireless broadband signal processor utilized with the wireless broadband signal processing system 10 according to at least one exemplary embodiment. Additional, fewer, or different operations may be performed depending on the particular embodiment or implementation.

[0062] According to at least one exemplary embodiment, the WBSP is employed as a signal processor and as such, needs to be under the control of a master processor, such as an ARM processor. The ARM processor thus needs to have the ability to read and write to the WBSP. The interface illustrated in Fig. 7 is entirely software defined and as such, is highly flexible. The ARM processor and WBSP can be programmed to define an interface that supports any protocol.

[0063] A "read" request is the mechanism for communicating the contents of a specific memory location inside of a specific WBSP buffer to the ARM processor. A "write" request is the mechanism for communicating from the ARM processor to the WBSP processor a specific value that is to be placed into a specific memory location inside of a specific buffer of the WBSP processor.

[0064] The "read" request supports information that the ARM processor may access from the WBSP processor for a variety of purposes, such as calibration, PHY statistics for host GUI Display (like RSSI), and dynamic algorithm inputs to ARM processing. The "write" request supports the communication of information that the ARM passes to the WBSP, such as DC Removal (I and Q) on TX, TX Power updates as a function of data rate, operating mode of modem 802.11 a/b/g (allows less processing for power consumption when dual acquisition is not required), and RSSI calculation active (again, allowing disabling for power consumption).

[0065] In State A1, the ARM processor initiates a request for a read or write request. In general, since the processors are operating asynchronously relative to each other, the WBSP processor is in State W1 which includes some general processing. Periodically, the WBSP processor transitions to State W2 to check the WBSP_STATUS bits. These bits are accessible as a GPI instruction. If WBSP_STATUS = 0, general processing resumes in State W1. If WBSP_STATUS is non-zero, then State W3 is transitioned where the ARM command is performed.

[0066] If the operation is a "read", the WBSP processor accesses the address specified in WBSP_ADDRESS. This one-dimensional address is translated into a two-dimensional WBSP address, including a buffer number and an address within the buffer. The contents of this location is accessed and the output stream is directed to the GPO associated with WBSP_DATA.

[0067] If the operation is a "write", the WBSP processor accesses the address specified in WBSP_ADDRESS. This one-dimensional address is translated into the two-dimensional WBSP address, including a buffer number and an address within the buffer. The value of WBSP_DATA is accessed via the GPI mechanism. The WBSP processor routes this value to the output stream which is destined for the decoded buffer number and address within the buffer.

[0068] In both the "read" and "write" cases, the value of WBSP_STATUS is reset to 0. Meanwhile, the ARM processor resumes its general processing in STATE A2. Periodically, the ARM processor checks the value of WBSP_STATUS via its MMIO register ARM_WBSP_ACCESS. When this value is 0, the ARM processor is aware that the "read" or "write" command has been completed. If this operation was a read, the ARM processor can access the read value in the WBSP_DATA register. Continued operation may occur (STATE A4) influenced by the "read" operation including the option of initiating another "read" or "write" command. Simultaneously, the WBSP operation may continue operation in STATE W3 influenced by the "write" operation.

[0069] Fig. 8 illustrates operations performed in an exemplary FFT algorithm performed in the wireless broadband signal processing system 10. Additional, fewer, or different operations may be performed in the algorithm depending on the particular embodiment or implementation. The FFT algorithm can be coded into a software program that resides in the program memory 12. In an operation 82, the data that is to undergo the FFT/IFFT transform is loaded into a buffer. Settings are initialized that govern the operation of subsequent operations. A second counter is initialized to two, and N is set to the $\log_2$ length of the input vector. In an operation 84, a GPIO instruction number 23 causes a reset of a master counter in processing unit 18. GPIO instruction number 13 signals the FFT length (N) to processing unit 18 (Fig. 1). The master counter is responsible for address generation as described in greater detail below.

[0070] In an operation 86, processing unit 18 performs a vector operation associated with the FFT/IFFT algorithm. In at least one embodiment, the upper limit of the length of the vector to be operated upon by the vector instruction is 128 words. For data lengths larger than 128 words, it is necessary to loop through the FFT/IFFT algorithm a sufficient number of times (e.g., if the data length is 2048 words, and the maximum vector length is 128 words, 16 iterations of the FFT/IFFT algorithm are required to perform the transform). In an operation 87, the value of the master counter is incremented only after the FFT/IFFT algorithm has operated on one 128 word segment of data (unless explicitly reset via a GPIO instruction 23) in operation 86.

[0071] In an operation 88, a second counter is advanced by two to proceed to the next stage of FFT/IFFT processing. Also, the INPUT and OUTPUT buffers are switched, enabling the cascading of processing between the FFT/IFFT stages. In an operation 89, if all the stages of the FFT/IFFT processing have been performed, then the FFT/IFFT transformed data is available for further processing by the processor.

[0072] Referring to Fig. 1, the memory 38 provides mathematical functions to the processing units 16, 18, and 20. In a preferred embodiment, the memory 38 is a read only memory (ROM). ROMs are relatively power consuming. As such, minimizing accesses to the memory 38 reduces the overall power required. In the FFT algorithm, it is necessary to access the memory 38 for mathematical functions, including Twiddle Factors used for the outputs of Radix-4 operations.

[0073] By a re-ordering of the segments of the input vector operated on by the FFT algorithm in a given stage, it is possible to use the same set of 3 Twiddle Factors for the outputs of successive Radix-4 operations. By way of example, consider a 4096-word FFT in which $\log_4(4096) = 6$ stages are required. For Stage 1, the 3 Twiddle Factors are accessed from the memory 38 every Radix-4 operation. It should be noted that the first output of the Radix-4 operation has a Twiddle Factor that is always unity, thus only 3 of the outputs are non-trivial. However, for the next stage or Stage 2 of the FFT algorithm, the same set of three Twiddle Factors may be used for 4 consecutive Radix-4 operations if the optimal address generation scheme is used as described below. For Stage 3 of the FFT algorithm, the same set of three Twiddle Factors may be used for 16 consecutive Radix-4 operations. For Stage 4, that number continues to grow geometrically to 64 consecutive Radix-4 operations.

[0074] Other design considerations can reduce the required amount of Twiddle Factor space in the memory 38. For example, since larger powers of 2 are supersets of the smaller powers of 2, only the Twiddle Factors corresponding to the largest FFT size need be stored. Thus, the Twiddle address generation supports all FFT sizes collapsed into a single table. The address generation scheme also supports reduction of the number of Twiddle Factors even for the largest FFT size. For example, taking an 8192-word FFT, adjacent Twiddle Factors are a factor of exp(j*2*pi/8192) different, which is too small to resolve in the fixed point representation of 10 bits. As such, a reduced set of Twiddle Factors are stored in which all odd values are discarded. By symmetry, the full unit circle of 2*pi radians can be constructed by storage of pi/4 (one octant) worth of Twiddle Factors. The unit circle reduces the storage requirement by an additional 1/8th. The Twiddle address generation coupled with the Twiddle Octant Manipulation Block (shown in processing unit 18 described with respect to Fig. 9) accomplishes this storage reduction.

[0075] Fig. 9 illustrates a more detailed view of the functionalities of the processor 18 described with reference to Fig. 1. In at least one embodiment, the processor 18 buffers four inputs (X1, X2, X3, and X4) for the ensuing Radix-4 FFT because the processor receives data serially from a single port RAM. The exception is the final Radix-2 stage on FFT sizes that are not an integral power of 4. In this case, only 2 inputs are buffered with X2 and X4 set to zero.

[0076] The Radix-4 FFT engine operates at a reduced clock rate relative to the rest of the wireless broadband signal processing system 10. In many embodiments, the Radix-4 FFT engine operates at the system clock frequency reduced by a factor of 4. The exception is the final Radix-2 stage on FFT sizes that are not an integral power of 4, in which case the system clock frequency is reduced by a factor of two. The Radix-4 FFT engine is optimized such that 8 complex additions can be performed to produce 4 outputs. The Radix-4 FFT engine includes 2 sets of cascaded adders. The first set of adders produces the following partial sums based on the 4 complex inputs:

$$
\begin{aligned}
P1 &= X1 + X3\\
P2 &= X1 - X3\\
P3 &= X2 + X4\\
P4 &= X2 - X4
\end{aligned}
$$

[0077] A second set of adders computes the outputs based upon the partial sums as:

$$Y1 = P1 + P3$$
$$Y2 = P2 - j*P4$$
$$Y3 = P1 - P3$$
$$Y4 = P2 + j*P4$$

[0078] where multiplication by j is implemented via switching I and Q and inverting the I output.

[0079] In general, there is no truncation in this operation.

[0080] The output of each scalar Twiddle factor multiplication is truncated to 11 bits. Therefore, the output of the complex multiplier is 12 bits. Bits [10:1] are mapped to the output of the processing unit 18. To reduce the rate at which Twiddle Factors are accessed, there are 3 storage registers 92 for storing the non-unity Twiddle factors. As further described below with respect to Figs. 10-13, the storage registers 92 only update when the Twiddle address transitions out of the Twiddle address generator mapping block. This transition is signaled to the storage registers 92 by the Twiddle Address transition indicator generated in operation 106, discussed in greater detail below. The multiplier 94 supports a bypass functionality on every 4th multiply when the unity Twiddle factor is to be applied. Based upon a 3-bit control word from a multiplier 110 shown in Fig. 10 and described below, the accessed Twiddle factor is manipulated by the Twiddle octant manipulator 90 as follows. The Twiddle factor is subjected to the cascaded effect of the 3 operations:

```
If Bit 1 xor Bit 2 == 1
    Swap I and Q of Twiddle Factor and negate real and imaginary
If Bit 2 == 1
    Negate Real of Twiddle Factor
If Bit 3 == 1
    Negate Both Real and Imaginary of Twiddle Factor
```

[0081] Fig. 10 illustrates operations performed in the address generation for the FFT algorithm described with reference to Fig. 9. Additional, fewer, or different operations may be performed depending on the particular embodiment or implementation. In an operation 104, the master counter information supplied by operation 102 is mapped by an input address generator to create an input address. Fig. 11 illustrates an exemplary mapping of the master counter information. As illustrated, the input address is populated according to N, the size of the input vector being transformed by the FFT algorithm. In the exemplary mapping illustrated in Fig. 11, the input address is 13 bits long where the highest-order 13-N bits are set to zero and $N = \log_2$ (FFT size), the next highest-order bits are s bits of the master counter where s = 2,4, ..., N-2, N (where N is even) and s = 2, 4, ..., N-1, N (where N is odd) and the lower-order bits of the input address are N-s bits of the master counter. Referring again to Fig. 10, once the input address is generated by operation 104, the input buffer receives the input address and, with the exception of the last stage described below, the output buffer also receives the input address.

[0082] In an operation 106, Twiddle factor addresses are generated. Fig. 12 illustrates an exemplary mapping for the Twiddle address. This exemplary mapping involves a re-shuffling of the input address generated in operation 104. The Twiddle address has 11 bits. The higher-order bits are the input address bits (N-s) to 1. The remaining lower-order bits of the Twiddle factor address (which is determined by subtracting the input address size, 11, by N-s) are set to zero.

[0083] In order to determine whether new Twiddle factors are needed and for the purpose of saving power, a transition determination is made to limit the number of accesses to memory 38 (such as a ROM). A Twiddle address transition indicator is generated by operation 106 which indicates that there is a change or transition in the Twiddle address and that new Twiddle factors are needed. The Twiddle address transition indicator is sent to the storage registers 92 in the processing unit 18 and the mathematical functions memory 38. When the memory 38 is accessed, three Twiddle factors are retrieved, manipulated as described above, and stored in the storage registers 92.

[0084] The following describes the population of the storage registers 92 with Twiddle factors and use of the Twiddle factors. In this process, the two least significant bits (LSB) of the master counter are multiplied with the Twiddle address using a multiplier 110. The product of this multiplication (13 bits in this exemplary embodiment) is separated into parts. Ten of the bits are provided as inputs to a summer 112 and a multiplexer 114. The summer 112 performs a subtraction of the ten bits from 512 and provides the result to an input 1 of the multiplexer 114. The other input of the multiplexer 114 (input 0) receives the ten bits from the multiplication result from the multiplier 110. One bit from the remaining bits from the multiplication result is used as a select to the multiplexer 114 and the 3 highest-order bits of the multiplication result are provided as the previously referenced control word to the Twiddle octant manipulator 90 in processor 18. The output of the multiplexer 114 is the address sent to the mathematical functions memory 38 for retrieving a Twiddle factor.

[0085] If the length of the input vector undergoing the FFT transform has a length which is odd power of 2 (non-integral multiple of 4), the output buffer receives an interleaved version of the input address formed in an operation 108. As illustrated in Fig. 13, the interleaving version of the input address depends on the value of N, which—as indicated above—represents $\log_2$ (FFT size). The 13 bits of the address provided to the output buffer includes zeros in the first 13-N bits, followed by the arrangement of the input address shown in Fig. 13. By design, the processing carried out and illustrated in Figs. 10-13 limits access to the memory 38 containing Twiddle factors, thereby saving power.

[0086] Fig. 14 illustrates operations performed in a context switching process carried out in the wireless broadband signal processing system 10. Additional, fewer, or different operations may be performed depending on the embodiment or implementation. In an operation 142, a critical task 1 operation is performed. A critical task is one or more operations, each operation needing to be completed before a new processing iteration during (PID) begins. For example, critical task 1 can include 802.11 operations that are performed when a processing iteration duration

(PID) instruction is received, each operation completing before a new PID is received. Once a critical task 1 operation is completed, a critical task 2 operation can be performed in an operation 144. For example, critical task 2 can be operations involved in copying DVB samples to an intermediate buffer. If a critical task 2 operation is completed before a non-critical task 3 is finished, a program induced context switch is performed in which a non-critical task operation is performed in operation 146. Non-critical operations may extend across PID boundaries. Such a non-critical task 3 can be a DVB demodulation. When a PID instruction is received, the induced context switch is ended. If the non-critical task is complete when critical task 2 is completed, a sleep mode is entered until the PID ends.

[0087] A conventional definition of context is a set of information from which a task may restart where it previously left off. During a context switch, the context of the "current" task is stored, and the context of the "next" task is loaded. The "current" task will be revisited at some future time by loading back in the previously stored context. The state of the WBSP is defined by a set of processor registers. In an illustrative example, a processor register is the Instruction Pointer, however there can be several additional processor registers. The WBSP incorporates sets of memory elements (e.g., hardware registers) for the complete description of a context. The number of sets of memory elements determines the maximum number of simultaneous contexts. In the WBSP, a context switch occurs when the information stored in a set of memory elements for a given context is loaded as the set of processor registers. In the WBSP, the entire set of memory elements is loaded into the processor registers in a single clock. At this point, the WBSP continues normal steady-state execution of instructions.

[0088] Fig. 15 depicts timing of the context switching process described with reference to Fig. 14. PID 1 initiates a critical task 1 operation. The critical task 1 operation is completed before PID 2 begins, allowing a critical task 2 operation and a non-critical task 3 operation to be performed. Upon receipt of PID 2, the non-critical task 3 is halted (although not completed yet) and critical task 1 operation is performed. Such a process continues where receipt of a PID triggers the execution of a critical task operation. The critical tasks operations are performed in order and if a new PID is not yet received, a non-critical task operation can be performed. As such, critical task operations are completed within the PID but inactive periods are utilized to execute non-critical tasks.

[0089] Fig. 16 illustrates a processing unit in the wireless broadband signal processing system 10. The processing unit can perform convolution operations (FIR filtering) and tap loading. An initial value and a stride value are provided to address generation logic 202. The address generation logic 202 generates addresses that are supplied to ROM 1, ROM 2, ROM 3, ROM 4, ROM 5, ROM 6, ROM 7 and ROM 8. Input data is received by the processing unit at an input shifter 204. The input shifter 204 performs the tap loading, loading the received data into registers 206, 208 and 212. The registers can be flip-flop structures.

[0090] Complex multiplication operations are carried out on data that has been loaded into the ROM structures at the locations corresponding to the addresses generated by the address generation logic 22 and the communication data. The products of these complex multiplication operations are summed by a complex adder tree 216. Multiplication beyond eight-fold parallel multiplication is allowed by a combine shifter 218 which feeds a combine stream into the complex tree adder 216. The convolution is thus built up by accumulating taps. The inclusion of the combine stream input into the complex tree adder 216 thus allows for dynamic range control. An output shifter 220 shifts data from the complex adder tree 216 as an output stream of data from the processing unit.

[0091] Fig. 17 illustrates address operation logic 202 from the processing unit of Fig. 16 in greater detail. An initialized address is received by the address generation logic 202 via a GPIO instruction. This initialized address is a current address. Addresses communicated to the ROM memory structures (Fig. 16) are the current address (A0), the current address plus a stride value, the current address plus a stride value times two, etc. As data is read from the ROM structures, the current address is incremented by the stride value. As such, incrementing the address is done automatically without needing to re-load the "top" or the value that the communication data is summed over.

[0092] The contents of ROM 1, ROM 2, ROM 3, ROM 4, ROM 5, ROM 6, ROM 7 and ROM 8 in Fig. 16 can be determined using the formulas below:

$$R_{A,n} = round\left(\frac{\sin x}{x} \times 512\right)$$

$$x = \frac{\pi \times A}{256} + (n-4) \times \pi$$

[0093] where R is the contents of the n-th ROM at address A and A is the address defined for value 0 through 255.

[0094] While several embodiments of the invention have been described, it is to be understood that modifications and changes will occur to those skilled in the art to which the invention pertains. Accordingly, the claims appended to this specification are intended to define the invention precisely.

CLAIMS

What is claimed is:

1. A method for obtaining processor diagnostic data, the method comprising:

receiving an instruction;

selectively enabling write access of an output stream to a diagnostic memory;

writing to the diagnostic memory at a first frequency; and

reading from the diagnostic memory at a second frequency, wherein the first frequency is greater than the second frequency.

2. The method of claim 1, further comprising communicating contents of the diagnostic memory to a logic analyzer.

3. The method of claim 1, wherein the diagnostic memory receives communication data from an outside source.

4. The method of claim 1, wherein the diagnostic memory receives communication data from a processing unit.

5. The method of claim 1, wherein write access of the output stream to the diagnostic memory is enabled when the received instruction changes.

6. The method of claim 1, wherein the first frequency is 40 MHz or more.

7. The method of claim 1, wherein the second frequency is 40 MHz or less.

8. The method of claim 1, wherein the received instruction comprises a diagnostic mailbox field.

9. The method of claim 8, wherein if the diagnostic mailbox field of a received instruction is set to one, the output stream of the received instruction is written to the diagnostic memory.

10. The method of claim 9, wherein the first frequency and the second frequency are chosen such that the second frequency is less than or equal to the first frequency times a fraction of clocks associated with instructions that have their diagnostic mailbox field set to one.

11. The method of claim 1, wherein the diagnostic memory is a random access memory (RAM) having at least one read port and at least one write port.

12. The method of claim 11, wherein the random access memory (RAM) is a dual-port RAM having one write port and one read port.

13. The method of claim 11, wherein read and write addresses applied to the diagnostic memory are automatically incremented after every read or write access to the diagnostic memory until either address matches a maximum RAM address at which point the read and write addresses wrap around to zero.

14. The method of claim 13, further comprising communicating an overflow indication when the diagnostic memory is full of data that has not been read and the received instruction indicates that the output stream is to be written to the diagnostic memory.

15. The method of claim 13, further comprising communicating an empty indication when all data stored in the diagnostic memory has been read.

16. A system for obtaining processor diagnostic data, the system comprising:

a memory containing instructions;

a controller that receives and executes the instructions, including selectively enabling write access of an output stream to a diagnostic memory; and a diagnostic memory that receives the output stream at a first frequency and delivers contents at a second frequency, wherein the first frequency is higher than the second frequency.

17. The system of claim 16, further comprising a logic analyzer, the logic analyzer receiving contents of the diagnostic memory.

18. The system of claim 16, wherein the diagnostic memory receives communication data from an outside source.

19. The system of claim 16, wherein the diagnostic memory receives communication data from a processing unit.

20. The system of claim 16, wherein the controller enables write access to the diagnostic memory when the received instructions change.

21. The system of claim 16, wherein the first frequency is 40 MHz or more.

22. The system of claim 16, wherein the second frequency is 40 MHz or less.

23. The system of claim 16, wherein the received instruction comprises a diagnostic mailbox field.

24. The system of claim 23, wherein if the diagnostic mailbox field of a received instruction is set to one, the output stream of the received instruction is written to the diagnostic memory.

25. The system of claim 24, wherein the first frequency and the second frequency are chosen such that the second frequency is less than or equal to the first frequency times a fraction of clocks associated with instructions that have their diagnostic mailbox field set to one.

26. The system of claim 16, wherein the diagnostic memory is a random access memory (RAM) having at least one read port and at least one write port.

27. The system of claim 26, wherein the random access memory (RAM) is a dual-port RAM having one write port and one read port.

28. The system of claim 26, wherein read and write addresses applied to the diagnostic memory are automatically incremented after every read or write access to the diagnostic memory until either address matches a maximum RAM address at which point the read and write addresses wrap around to zero.

29. The system of claim 28, further comprising communicating an overflow indication when the diagnostic memory is full of data that has not been read and the received instruction indicates that the output communication stream is to be written to the diagnostic memory.

30. The system of claim 28, further comprising communicating an empty indication when all data stored in the diagnostic memory has been read.

31. A method of controlling input and output in a multi-mode wireless processing system, the method comprising:

receiving an instruction for controlling an interface between an element external to a multi-mode wireless processing system and the multi-mode wireless processing system; and

determining from a field in the received instruction whether a designated processing unit generates output data or receives input data.

32. The method of claim 31, wherein the received instruction indicates recording of a frame rate of errors.

33. The method of claim 31, wherein the received instruction indicates managing of sample buffers internal to the multi-mode wireless processing system.

34. The method of claim 31, further comprising determining from the field in the received instruction a general purpose input as a source of the input data.

35. The method of claim 34, wherein the received instruction indicates a rate of communication between the designated processing unit and the general purpose input.

36. The method of claim 34, further comprising routing the input data from the source to the designated processing unit.

37. The method of claim 31, further comprising determining from the field in the received instruction a general purpose output as a destination for the generated output data.

38. The method of claim 37, wherein the received instruction indicates a rate of communication between the designated processing unit and the general purpose output.

39. The method of claim 37, further comprising routing the generated output data from the designated processing unit to the destination.

40. A configuration of input / output components for interfacing with a processing unit in a multi-mode wireless processing system, the configuration comprising:

a plurality of general purpose inputs for supplying input data to a processing unit in a multi-mode wireless processing system; and

a plurality of general purpose outputs for receiving output data generated by the processing unit in the multi-mode wireless processing system.

41. A system for controlling input and output in a multi-mode wireless processor, the system comprising:

a memory including instructions in a multi-mode wireless processor system, the instructions controlling an interface between an element external to the multi-mode wireless processing system and the multi-mode wireless processing system; and

a controller that receives the instructions and determines from an instruction field whether a designated processing unit in the multi-mode wireless processing system generates output data or receives input data.

42. The system of claim 41, wherein the received instruction indicates recording of a frame rate of errors.

43. The system of claim 41, wherein the received instruction indicates managing of sample buffers internal to the multi-mode wireless processing system.

44. The system of claim 41, wherein the controller further determines from the instruction field a general purpose input as a source of the input data.

45. The system of claim 44, wherein the controller routes the input data from the source to the designated processing unit.

46. The system of claim 44, wherein the received instruction indicates a rate of communication between the designated processing unit and the general purpose input.

47. The system of claim 41, wherein the controller further determines from the instruction field a general purpose output as a destination for the generated output data.

48. The system of claim 47, wherein the controller routes the output data from the designated processing unit to the destination.

49. The system of claim 47, wherein the received instruction indicates a rate of communication between the designated processing unit and the general purpose output.

50. A method of dynamically controlling rate connections to sample buffers in a multi-mode processing system, the method comprising:

receiving an instruction for communication in a multi-mode wireless processing system;

determining a rate at which a plurality of buffers are serially connected to elements external to the multi-mode wireless processing system for receipt or transmission of data; and

programming a plurality of registers that control the rate at which the plurality of buffers are serially connected to the external elements based on the received instruction.

51. The method of claim 50 wherein one registers controls the rate at which the plurality of buffers are serially connected to the external elements.

52. The method of claim 50, wherein the rate at which the plurality of buffers are serially connected to the external elements varies dynamically based on the received instruction.

53. The method of claim 50, wherein a field in the received instruction determines whether or not the rate at which the plurality of buffers are serially connected to the external elements is to be changed.

54. A system for dynamically controlling rate connections to sample buffers in a multi-mode processing system, the system comprising:

a memory including instructions for multi-mode wireless processor communication in a multi-mode wireless processing system; and

a controller that receives instructions and determines a rate at which a plurality of buffers are serially connected to elements external to the multi-mode wireless processing system for receipt or transmission of data; and

a plurality of registers that control the rate at which the plurality of buffers are serially connected to the external elements.

55. The system of claim 54, wherein the controller dynamically varies the rate at which the plurality of buffers are serially connected to the external elements based on the received instructions.

56. The system of claim 54, wherein one register controls the rate at which the plurality of buffers are serially connected to the external elements.

57. The system of claim 54, wherein the controller determines whether the plurality of registers is to be programmed based upon a field in the received instruction.

58. A method of interfacing two processors, the method comprising:

generating a read/write request at a first processor, wherein the read/write request targets a target memory for which the first processor has no direct access;

receiving the read/write request at a second processor, wherein the second processor has direct access to the target memory to be accessed by the read/write request;

completing a read/write operation at the second processor; and

receiving at the first processor an indication that the read/write operation has been completed.

59. The method of claim 58, further comprising continuing operation at the first processor after receipt of read data from the second processor, the continuing operation being related to the read/write request, the read/write request being a read operation.

60. The method of claim 58, further comprising continuing operation at the second processor after completion of a write operation by the second processor, the continuing operation being related to the read/write request, the read/write request being the write operation.

61. The method of claim 58, further comprising:

generating a read/write address comprising a target buffer number and a target address within a target buffer, the target memory being the target buffer which is part of the second processor; and

receiving the read/write address at the second processor.

62. The method of claim 58, further comprising receiving write data at the second processor if the read/write request is a write request.

63. The method of claim 58, further comprising polling at the second processor for the read/write request of the first processor.

64. The method of claim 63, wherein the polling at the second processor is performed by periodic monitoring of status bits.

65. The method of claim 64, wherein a read/write request is indicated by the status bits being set to a nonzero value.

66. The method of claim 64, wherein the status bits are cleared upon completion of the read/write operation by the second processor.

67. The method of claim 58, further comprising polling at the first processor for indication that the read/write operation has been completed.

68. The method of claim 67, wherein the polling at the first processor is performed by periodic monitoring of the status bits.

69. A system for interfacing two processors, the system comprising:

a first processor that generates a read/write request, wherein the read/write request targets a target memory for which the first processor has no direct access;

a second processor that receives the read/write request and completes a read/write operation, wherein the second processor has direct access to the target memory to be accessed by the read/write request;

a target memory; and

a means for communicating between the first processor and a second processor.

70. The system of claim 69, wherein the second processor is a multi-mode wireless processor.

71. The system of claim 69, wherein the first processor is an ARM processor.

72. The system of claim 69, wherein the target memory is a part of the second processor.

73. The system of claim 69, wherein the second processor receives write data if the read/write request is a write request.

74. The system of claim 69, wherein after receipt of read data from the second processor the first processor performs operations influenced by the read/write request, the read/write request being a read operation.

75. The system of claim 69, wherein after the second processor completes a write operation, the second processor performs operations influenced by the read/write request, the read/write request being the write operation.

76. The system of claim 69, wherein the target memory is a buffer.

77. The system of claim 76, wherein the first processor generates a read/write address comprising a target buffer number and a target address within the target buffer, the target memory being the target buffer which is part of the second processor, and the second processor receives the generates read/write address.

78. The system of claim 69, wherein the second processor polls for the read/write request of the first processor.

79. The system of claim 78, wherein the polling by the second processor is performed by periodic monitoring of status bits.

80. The system of claim 79, wherein a read/write request is indicated by the status bits being set to a nonzero value.

81. The system of claim 69, wherein the first processor polls for the indication that the read/write operation has been completed.

82. The system of claim 81, wherein the first processor polling is performed by periodic monitoring of status bits.

83. The system of claim 82, wherein the status bits are cleared upon completion of the read/write operation by the second processor.

84. An interface between two processors, the interface comprising:

means for generating a read/write request at a first processor;

means for setting status bits by either the first processor or a second processor;

means for polling the status bits by the first processor;

means for polling the status bits by the second processor; and

means for communicating additional data between the first processor and the second processor.

85. The interface of claim 84, wherein the second processor polls the status bits on a periodic basis.

86. The interface of claim 84, wherein the first processor sets the status bits to zero to indicate a read/write request.

87. The interface of claim 84, wherein if the read/write request is for a write operation, the first processor further supplies write data to the second processor.

88. The interface of claim 84, wherein the second processor clears the status bits when a requested read/write operation has been completed.

89. The interface of claim 84, wherein the second processor sends write data to the first processor when the second processor has completed a write operation.

90. The interface of claim 84, wherein the first processor polls the status bits on a periodic basis.

91. The interface of claim 84, wherein first processor supplies an address to the second processor as part of the read/write request.

92. The interface of claim 91, wherein the address comprises a target buffer number, and a target address within a target buffer.

93. A Fast Fourier Transform (FFT) method in a multi-mode wireless processing system, the method comprising:

loading an input vector into an input buffer;

initializing a second counter and a variable N, where $N = \log_2$ (input vector size), and s is a value of the second counter;

performing an FFT stage, the FFT stage comprising:

performing vector operations on data in the input buffer and sending results to an output buffer, the data in the input buffer comprising a plurality of segments;

advancing the value of the second counter; and

switching roles of the input and output buffers; and

comparing s to N, and performing additional FFT stages until $s = N$.

94. The method of claim 93, wherein the second counter is initialized to two, advanced by two in the FFT stage, and is set to N in a last FFT stage if N is odd.

95. The method of claim 93, wherein the vector operations operate on one segment of the data in the input buffer at a time until all of the segments have been operated on.

96. The method of claim 93, wherein the vector operations comprise:

loading four input data from the input buffer into a processing unit;

performing Radix-4 FFT vector operations with a Radix-4 FFT engine on the four input data loaded in the processing unit, the Radix-4 FFT engine accepting four input vectors and generating four output vectors;

multiplying the four generated output vectors with a Twiddle factor, each output vector having an associated Twiddle factor, the Twiddle factor having a real component and an imaginary component; and

bypassing multiplication of the output vectors when the associated Twiddle factor is unity.

97. The method of claim 96, further comprising bypassing multiplication of the first output vector.

98. The method of claim 96, wherein if N is odd and the last FFT stage is being executed, two input data are loaded from the input buffer into the processing unit and are used as the first and third Radix-4 FFT engine input vectors, the second and fourth Radix-4 FFT engine input vectors being set to zero.

99. The method of claim 96, wherein the four input data loaded into the processing unit are received serially by the processing unit and provided in parallel to the Radix-4 FFT engine, and the four output vectors of the Radix-4 FFT engine are received in parallel from the Radix-4 FFT engine and written to the output buffer serially.

100. The method of claim 96, wherein the processing unit operates at a multi-mode wireless processing system clock frequency reduced by a factor of four, except for when a last FFT stage is being performed and N is odd, in which case the processing unit operates at the multi-mode wireless processing system clock frequency reduced by a factor of two.

101. The method of claim 100, wherein a master counter is used as a loop variable that is initialized, advanced, and compared to a length of the data in the input buffer to determine when all of the segments of the data in the input buffer data have been operated on.

102. The method of claim 101, wherein input buffer addresses are generated as follows: bits N to (s+1) of the master counter are mapped bits (N-s) to 1 of the input buffer address, bits s to 1 of the master counter are mapped to bits N to (N-s+1) of the input buffer address, and remaining highest-order bits of the input buffer address are set to zero, where bit 1 is the lowest-order bit of the input buffer address and the master counter.

103. The method of claim 102, wherein the input address is 13 bits.

104. The method of claim 102, wherein an output buffer address is equal to the input buffer address for all of the FFT stages except for the last FFT stage in which the output buffer address is generated as follows: bits 13 to 13-N bits of the output buffer address are set to zero, and if N is even, bits N to 1 of the output buffer follow a first mapping sequence $I_2, I_1, I_4, I_3, \ldots I_N, I_{N-1}$, and if N is odd, bits N to 1 of the output buffer follow a second mapping sequence $I_1, I_3, I_2, I_5, I_4, \ldots I_N, I_{N-1}$, where I is the input buffer address, and where bit 1 is the lowest-order bit of the output buffer, where bit 1 is the lowest-order bit of the input and output buffer addresses.

105. The method of claim 96, wherein the Twiddle factor is generated by:

generating a preliminary Twiddle address;

generating a control word for controlling manipulation of the Twiddle factor;

generating a final Twiddle address;

determining whether the Twiddle factor needs to be accessed from a memory based upon the preliminary Twiddle address; and

if the Twiddle factor needs to be accessed:

reading the Twiddle factor from the memory at the final Twiddle address;

manipulating the Twiddle factor based upon the control word; and storing a manipulated Twiddle factor in the processing unit.

106. The method of claim 105, wherein the manipulated Twiddle factor stored in the processing unit is stored in a register.

107. The method of claim 105, wherein the preliminary Twiddle address is generated as follows: highest-order (N-s) bits of the preliminary Twiddle address are mapped to bits (N-s) to 1 of the input buffer address, and remaining lower-order bits of the preliminary Twiddle address are set to zero, where bit 1 is the lowest-order bit of the input buffer address.

108. The method of claim 105, wherein the preliminary and final Twiddle addresses are 11 bits.

109. The method of claim 105, wherein the control word is three highest-order bits of a product between the preliminary Twiddle address and two lowest-orders bits of the master counter.

110. The method of claim 105, wherein the Twiddle factor is manipulated according to the control word bits as follows:

first, if bit 1 of the control word XOR bit 2 of the control word = 1, the real and the imaginary components of the Twiddle factor are swapped and the real and imaginary components of the Twiddle factor are negated;

second, if bit 2 of the control word = 1, the real component of the Twiddle factor are negated; and

third, if bit 3 of the control word = 1, the real and imaginary components of the Twiddle factor are negated.

111. The method of claim 105, wherein the final Twiddle address is generated by:

multiplying the preliminary Twiddle address by two lowest-order bits of the master counter and generating a product;

subtracting bits 9 to 0 of the product from 512 and producing a remainder, where bit 0 is the least significant bit of the product,

sending the remainder to a first input of a 2:1 multiplexer, bits 9 to 0 of the product to a second input of the 2:1 multiplexer, and bit 10 of the product to a select input of the 2:1 multiplexer, the final Twiddle address being an output of the 2:1 multiplexer.

112. A system for performing a Fast Fourier Transform (FFT) in a multi-mode wireless processing system, the system comprising:

a processing unit for performing vector operations;

a memory for providing mathematical functions to the processing unit;

a program memory containing instructions for executing an FFT algorithm;

an instruction controller for receiving and executing instructions from the program memory; and

a pair of buffers that alternate between acting as an input buffer and an output buffer in successive FFT stages of the FFT algorithm, data in the input buffer comprising a plurality of segments.

113. The system of claim 112, wherein the memory providing mathematical functions contains Twiddle factors.

114. The system of claim 112, wherein the memory providing mathematical functions is a ROM.

115. The system of claim 112, wherein the processing unit comprises:

a Radix-4 FFT engine that performs eight complex additions on four input vectors and generates four output vectors;

a Twiddle multiplier for multiplying a generated output vector with an associated Twiddle factor, the Twiddle factor having a real component and an imaginary component;

a serial-to-parallel converter for receiving the four input vectors serially from the input buffer and sending the four input vectors to the Radix-4 FFT engine in parallel;

a parallel-to-serial converter for receiving the four generated output vectors in parallel and delivering the four output vectors serially to the Twiddle multiplier and output buffer;

a set of registers for storing manipulated Twiddle factors in the processing unit;

a Twiddle octant manipulator that manipulates Twiddle factors based upon a control word;

a master counter used as a loop variable for monitoring progress of the FFT algorithm in a given FFT stage;

a second counter used as a loop variable for keeping track of a current stage of the FFT algorithm, where s is a value of the second counter;

an input address generator that generates an input buffer address, the input buffer address being used as an output buffer address for all FFT stages except for when a last FFT stage is being performed and N is odd, where N = log2 (size of data in the input buffer);

a Twiddle address generator for generating a preliminary Twiddle address;

a DiBit interleaving generator that generates the output buffer address for the last FFT stage if N is odd;

a Twiddle address multiplier for generating the control word;

a summer for subtracting bits 9 to 0 of the product generated by the Twiddle address multiplier from 512 and generating a remainder; and

a 2:1 multiplexer for generating the final Twiddle address from the remainder and the product generated by the Twiddle address multiplier.

116. The system of claim 115, wherein the second counter is initialized to two, advanced by two in the FFT stage, and is set to N in the last FFT stage if N is odd.

117. The system of claim 115, wherein the processing unit operates on one segment of the data in the input buffer at a time until all of the segments have been operated on.

118. The system of claim 115, further comprising a multiplier bypass indicator for indicating when the Twiddle multiplier is to be bypassed.

119. The system of claim 115, wherein when N is odd and the last FFT stage is being executed, the serial-to-parallel converter receives two input data from the input buffer and the two received input data become the first and third Radix-4 FFT engine input vectors, and the second and fourth Radix-4 FFT engine input vectors are set to zero.

120. The system of claim 115, wherein the processing unit operates at a multi-mode wireless process system clock frequency reduced by a factor of four, except for when the last FFT stage is being performed and N is odd in which case the processing unit operates at the system clock frequency reduced by a factor of two.

121. The system of claim 115, wherein the input buffer address are generated as follows: bits N to (s+1) of the master counter are mapped bits (N-s) to 1 of the input buffer address, bits s to 1 of the master counter are mapped to bits N to (N-s+1) of the input buffer address, and remaining highest-order bits of the input buffer address are set to zero, where bit 1 is the least significant bit of the master counter and input buffer address.

122. The system of claim 115, wherein the input buffer address is 13 bits.

123. The system of claim 115, wherein the output buffer address is equal to the input buffer address for all FFT stages except for the last FFT stage in which the output buffer address is generated as follows: bits 13 to 13-N bits of the output buffer are set to zero, and if N is even, bits N to 1 of the output buffer follow a first mapping sequence I_2, I_1, I_4, I_3, … I_N, I_{N-1}, and if N is odd, bits N to 1 of the output buffer follow a second mapping sequence I_1, I_3, I_2, I_5, I_4, … I_N, I_{N-1}, where I is

the input buffer address, where bit 1 is the lowest-order bit of the input and output buffer addresses.

124. The system of claim 115, wherein the Twiddle address generator determines if new Twiddle factors need to be accessed from the memory providing mathematical functions and generates a Twiddle address transition indicator indicating that new Twiddle factors need to be accessed from the memory providing mathematical functions, the Twiddle address transition indicator being sent to the set of registers.

125. The system of claim 115, wherein the preliminary Twiddle address is generated as follows: highest-order (N-s) bits of the preliminary Twiddle address are mapped to bits (N-s) to 1 of the input buffer address, and remaining lower-order bits of the preliminary Twiddle address are set to zero, where bit 1 is the lowest-order bit of the input buffer address.

126. The system of claim 115, wherein the preliminary and final Twiddle addresses are 11 bits.

127. The system of claim 115, wherein the control word is three highest-order bits of the product of the preliminary Twiddle address and two lowest-orders bits of the master counter.

128. The system of claim 115, wherein the Twiddle factor is manipulated according to the control word as follows:

first, if bit 1 of the control word XOR bit 2 of the control word = 1, the real and the imaginary components of the Twiddle factor are swapped, and the real and imaginary components of the Twiddle factor are negated;

second, if bit 2 of the control word = 1, the real component of the Twiddle factor is negated;

third, if bit 3 of the control word = 1, both the real and imaginary components of the Twiddle factor are negated.

129. The system of claim 115, wherein the remainder is sent to a first input of the 2:1 multiplexer, bits 9 to 0 of the product generated by the Twiddle address multiplier are sent to a second input of the 2:1multiplexer, bit 10 of the product generated by the Twiddle address multiplier is sent to a select input of the 2:1 multiplexer, and the final Twiddle address is an output of the 2:1 multiplexer.

130. A method for switching between instruction contexts within a time interval in a multi-mode wireless broadband processing system, the method comprising:

executing critical task operations, each critical task operation executing within a time interval, a critical task comprising a plurality of critical task operations;

executing non-critical task operations, execution of each non-critical task operation being able to cross a time interval boundary, a non-critical task comprising a plurality of non-critical task operations; and

entering a sleep mode in which no critical task operations or non-critical task operations are executed, if the critical task operations and the non-critical task operations begun in the time interval have been completed before a following time interval begins.

131. The method of claim 130, wherein the non-critical task operations are not begun in the time interval until the critical task operations have been executed.

132. The method of claim 130, wherein at least one critical task operation is executed within the time interval.

133. The method of claim 130, wherein a context is stored in hardware registers.

134. The method of claim 130, wherein a number of sets of hardware registers determines a maximum number of simultaneous contexts.

135. A system for switching between instruction contexts within a time interval in a multi-mode wireless broadband processing system, the system comprising:

a memory containing instructions, the instructions comprising critical and non-critical task operations, a critical task comprising a plurality of critical task operations and a non-critical task comprising a plurality of non-critical task operations respectively; and

a controller that receives and executes the instruction, the critical task operations being executed within a time interval, and execution of each non-critical task operation being able to cross a time interval boundary,

wherein a multi-mode wireless broadband processing system comprising the controller and the memory enters a sleep mode in which no critical task operations or non-critical task operations are executed, if the critical task operations and the non-critical task operations begun in the time interval have been completed before a following time interval begins.

136. The system of claim 135, wherein the non-critical task operations are not begun in the time interval until the critical task operations have been executed.

137. The system of claim 135, wherein at least one critical task operation is executed within the time interval.

138. The system of claim 135, wherein a context is stored in memory elements.

139. The system of claim 138, wherein a number of sets of memory elements determines a maximum number of simultaneous contexts.

140. A method for performing a convolution operation in a multi-mode wireless processing system, the method comprising:

loading an initial value and a stride value into an address generator;

generating an address based on the initial value and the stride value;

supplying the generated address to a series of memories;

loading input data into a series of registers, the series of registers being equal in number to the series of memories, each register being associated with one memory;

multiplying contents of each register with a value stored at the generated address in the memory associated with each register and generating a series of products;

adding the series of products and producing a product sum; and

generating an output stream from the product sum.

141. The method of claim 140, wherein the generated address is initially set to the initial value.

142. The method of claim 140, wherein the registers are flip-flop structures.

143. The method of claim 140, wherein the memories are ROMs.

144. The method of claim 140, wherein the multiplications of register contents with memory contents are performed in parallel.

145. The method of claim 140, wherein the addition of products is performed by a complex adder tree.

146. The method of claim 140, wherein input from a combine shifter is included in the product sum.

147. The method of claim 140, wherein a value R stored at an address A of the memory n is determined as follows:

$$R_{A,n} = round\left(\frac{\sin x}{x} \times 512\right)$$

$$x = \frac{\pi \times A}{256} + (n-4) \times \pi$$

and A is defined for values 0 through 255.

148. The method of claim 140, wherein there are eight memories and eight registers.

149. The method of claim 140, further comprising:

performing subsequent multiplications between contents of each register and the value stored at the generated address in the memory associated with each register, the generated address being increased by the stride value in the subsequent multiplications;

adding the products of the subsequent multiplications and producing subsequent product sums; and

generating subsequent output streams based on the subsequent product sums.

150. The method of claim 149, wherein the address generator increments the generated address by the stride value automatically.

151. A system for performing a convolution operation in a multi-mode wireless processing system, the system comprising:

an address generator for generating an address given an initial value and a stride value;

a series of memories;

a series of registers for storing an input value;

a series of complex multipliers, the series of multipliers, registers, and memories being equal in number, each multiplier being associated with one register and one memory, each multiplier generating a product of contents of the associated register and a value stored at the generated address in the associated memory; and

a complex adder tree for adding the series of products and producing a product sum.

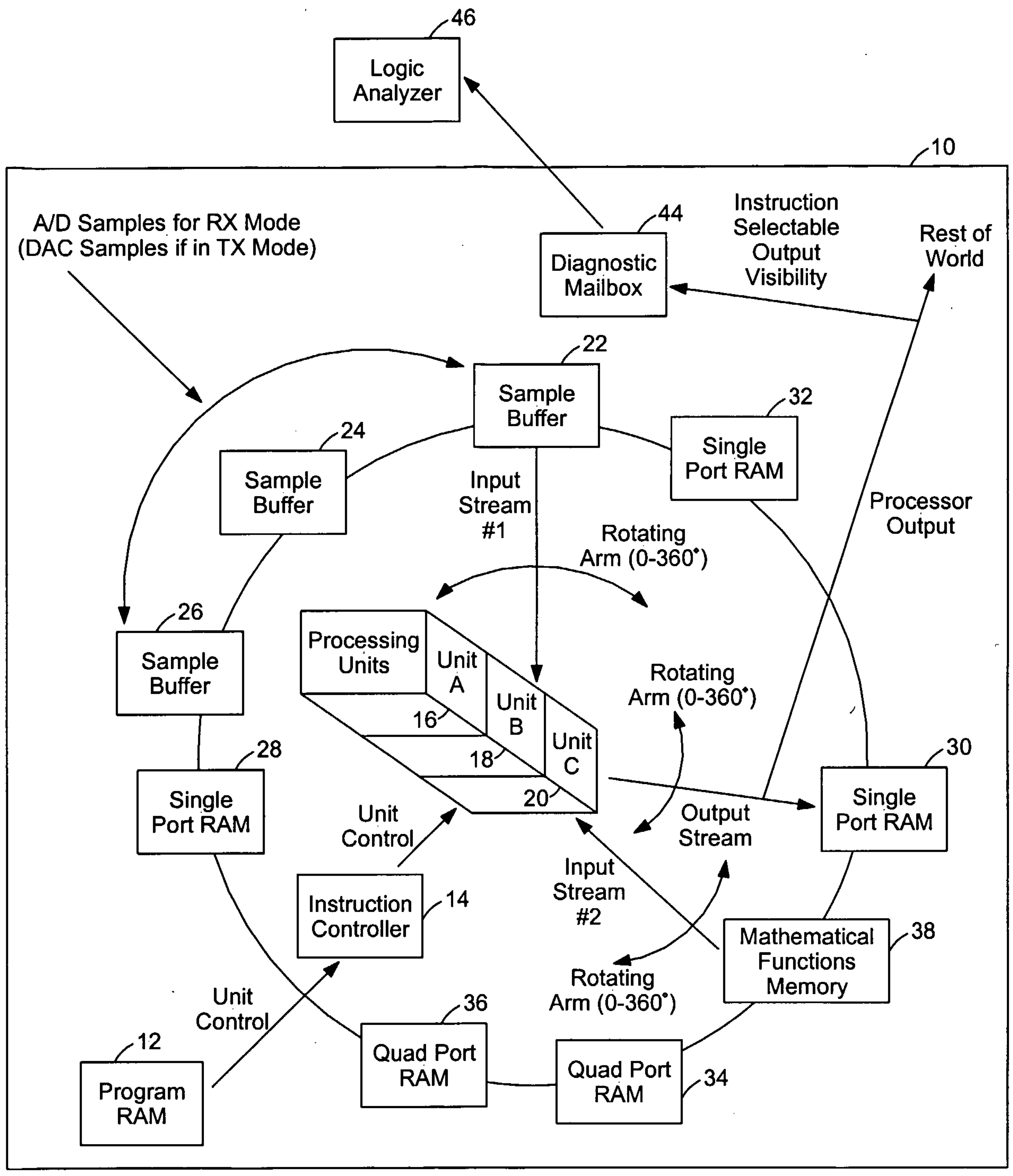

46
Logic Analyzer
10
A/D Samples for RX Mode (DAC Samples if in TX Mode)
44
Diagnostic Mailbox
Instruction Selectable Output Visibility
Rest of World
22
Sample Buffer
24
Sample Buffer
32
Single Port RAM
Input Stream #1
Rotating Arm (0-360°)
Processor Output
26
Sample Buffer
Processing Units
Unit A
Unit B
Unit C
16
18
20
Rotating Arm (0-360°)
28
Single Port RAM
30
Single Port RAM
Unit Control
Output Stream
Input Stream #2
14
Instruction Controller
38
Mathematical Functions Memory
Unit Control
Rotating Arm (0-360°)
36
Quad Port RAM
34
Quad Port RAM
12
Program RAM

FIG. 1

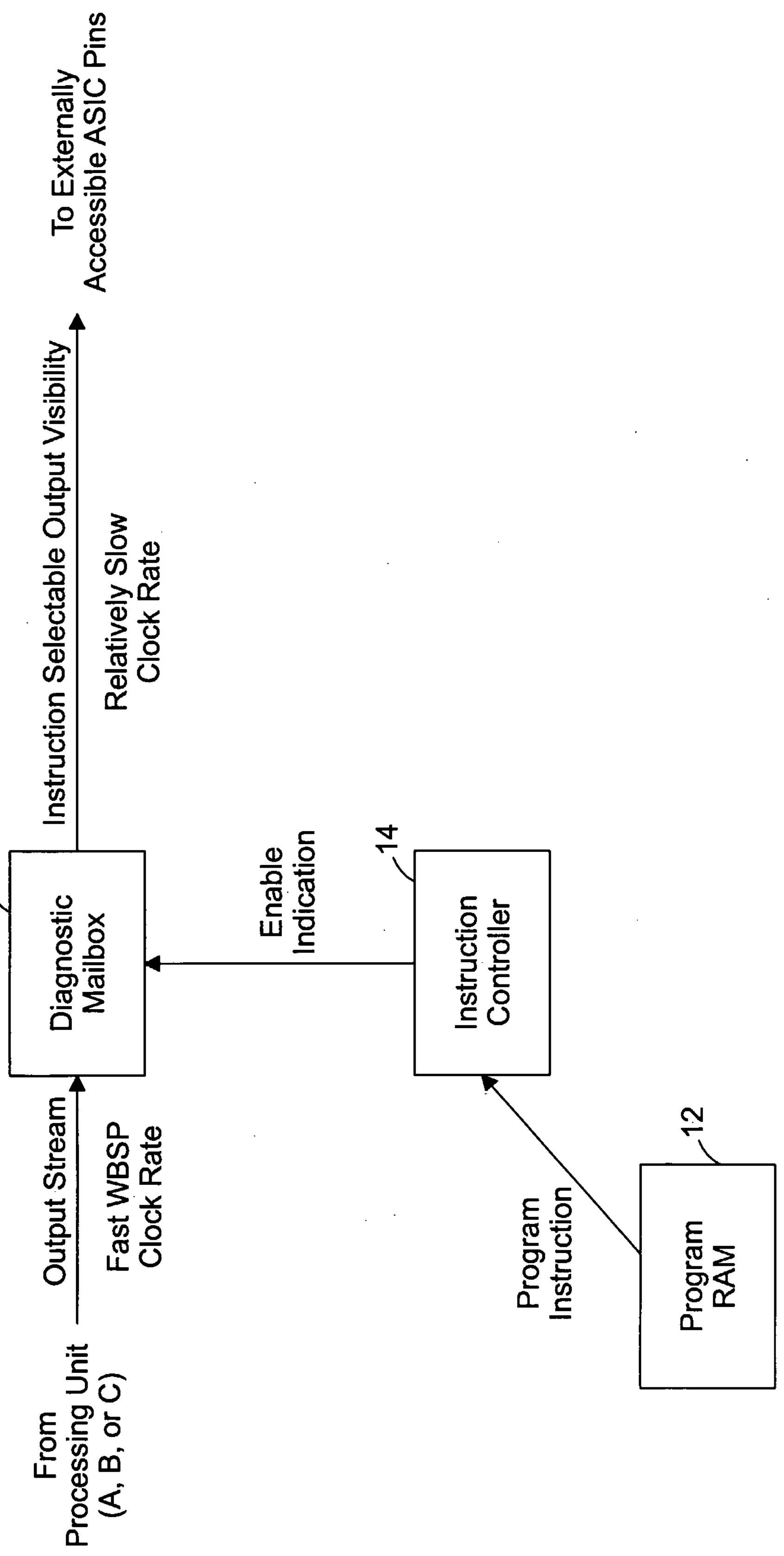

From Processing Unit (A, B, or C)
Output Stream
Fast WBSP Clock Rate
44
Diagnostic Mailbox
Instruction Selectable Output Visibility
Relatively Slow Clock Rate
To Externally Accessible ASIC Pins
Enable Indication
14
Instruction Controller
Program Instruction
12
Program RAM

FIG. 2

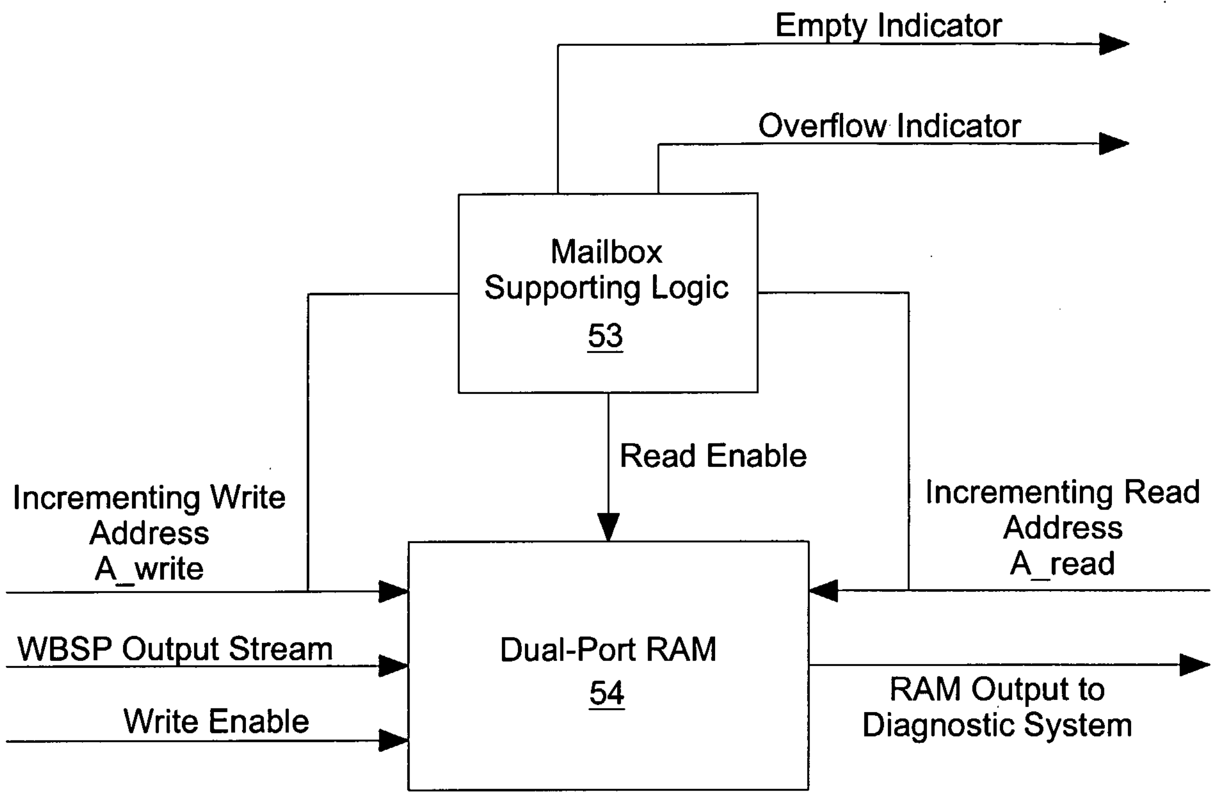

Empty Indicator
Overflow Indicator
Mailbox Supporting Logic 53
Read Enable
Incrementing Write Address A_write
Incrementing Read Address A_read
WBSP Output Stream
Write Enable
Dual-Port RAM 54
RAM Output to Diagnostic System

FIG. 3

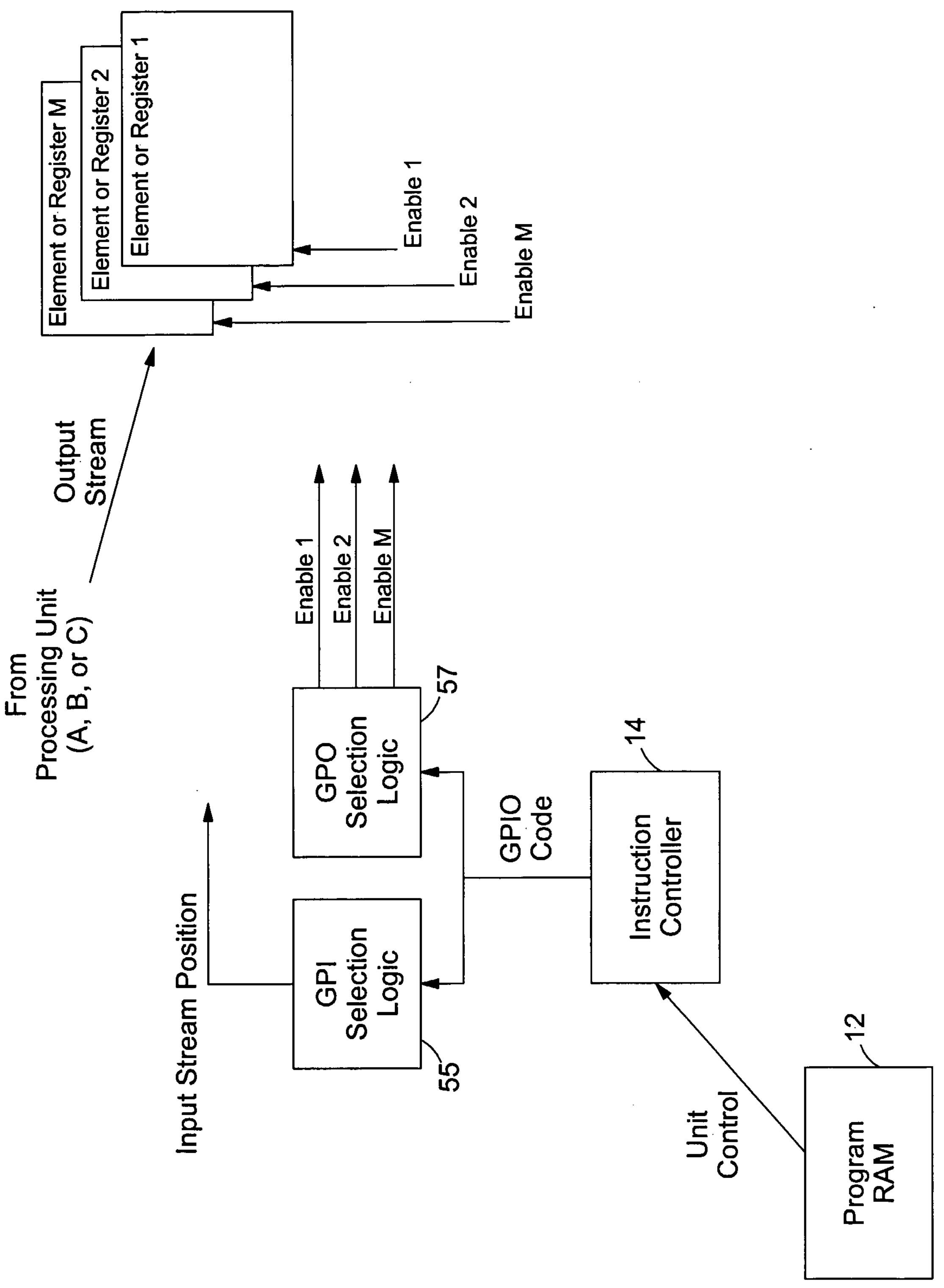
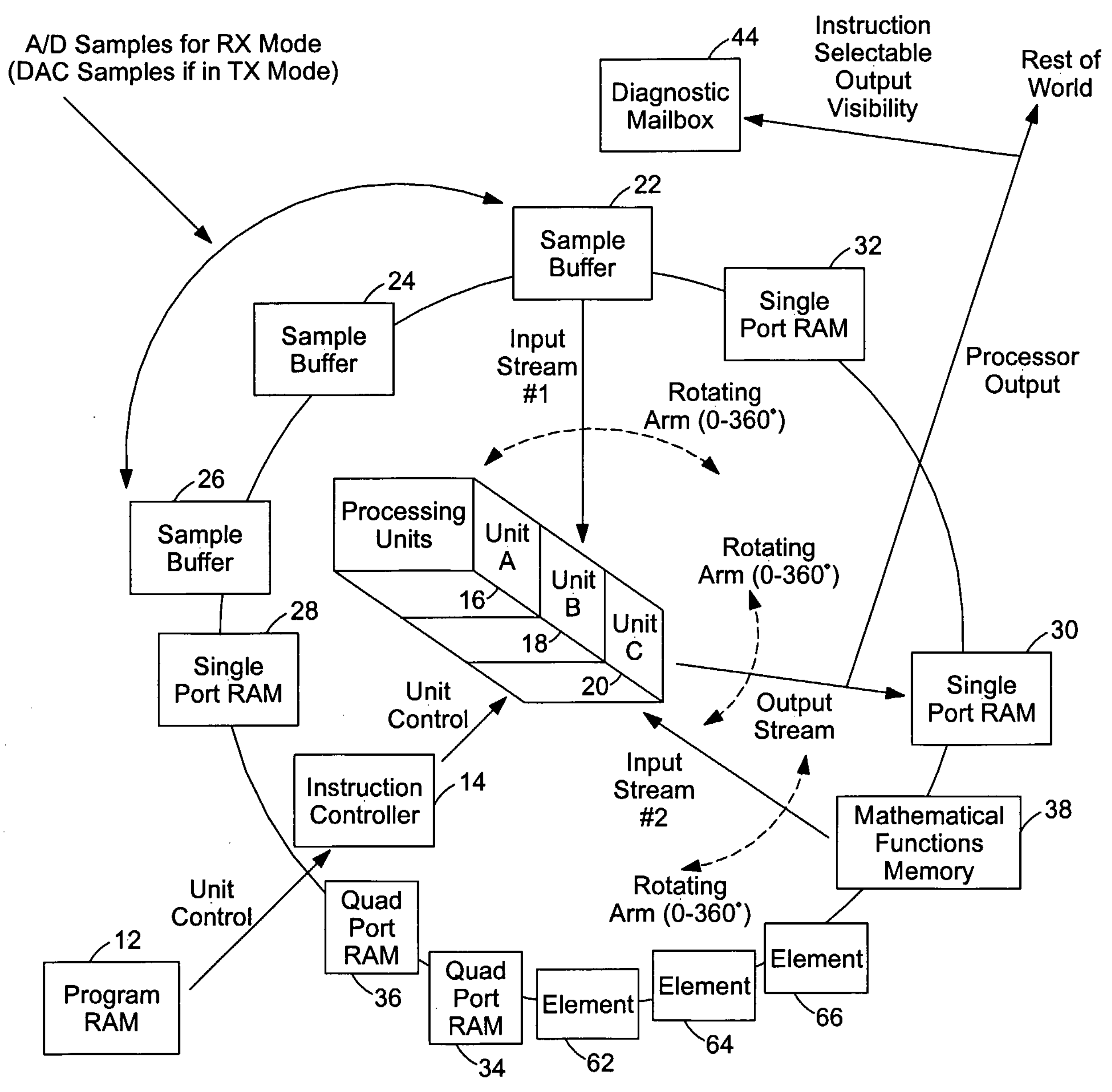

From Processing Unit (A, B, or C)
Output Stream
Element or Register M
Element or Register 2
Element or Register 1
Enable 1
Enable 2
Enable M
Input Stream Position
GPI Selection Logic
55
GPO Selection Logic
57
Enable 1
Enable 2
Enable M
GPIO Code
Instruction Controller
14
Unit Control
Program RAM
12

FIG. 4



A/D Samples for RX Mode
(DAC Samples if in TX Mode)
44
Diagnostic Mailbox
Instruction Selectable Output Visibility
Rest of World
22
Sample Buffer
32
Single Port RAM
24
Sample Buffer
Input Stream #1
Processor Output
Rotating Arm (0-360°)
26
Sample Buffer
Processing Units
Unit A
Unit B
Unit C
16
18
20
Rotating Arm (0-360°)
28
Single Port RAM
30
Single Port RAM
Unit Control
Output Stream
14
Instruction Controller
Input Stream #2
Mathematical Functions Memory
38
Unit Control
Rotating Arm (0-360°)
12
Program RAM
Quad Port RAM
36
Quad Port RAM
34
Element
62
Element
64
Element
66

FIG. 5

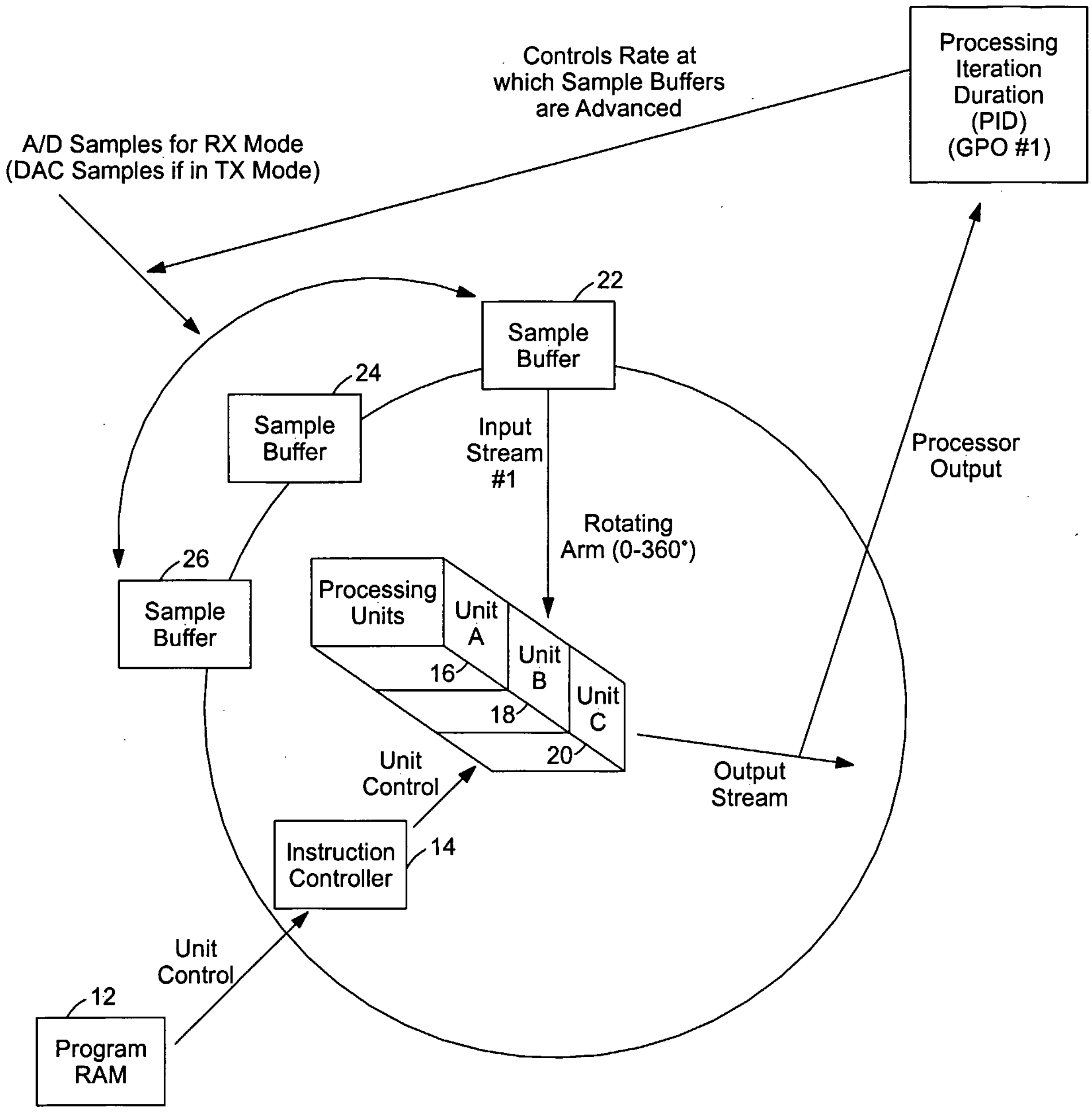

Controls Rate at which Sample Buffers are Advanced
Processing Iteration Duration (PID) (GPO #1)
A/D Samples for RX Mode (DAC Samples if in TX Mode)
22
Sample Buffer
24
Sample Buffer
Input Stream #1
Processor Output
Rotating Arm (0-360°)
26
Sample Buffer
Processing Units
Unit A
Unit B
Unit C
16
18
20
Unit Control
Output Stream
Instruction Controller
14
Unit Control
12
Program RAM

FIG. 6

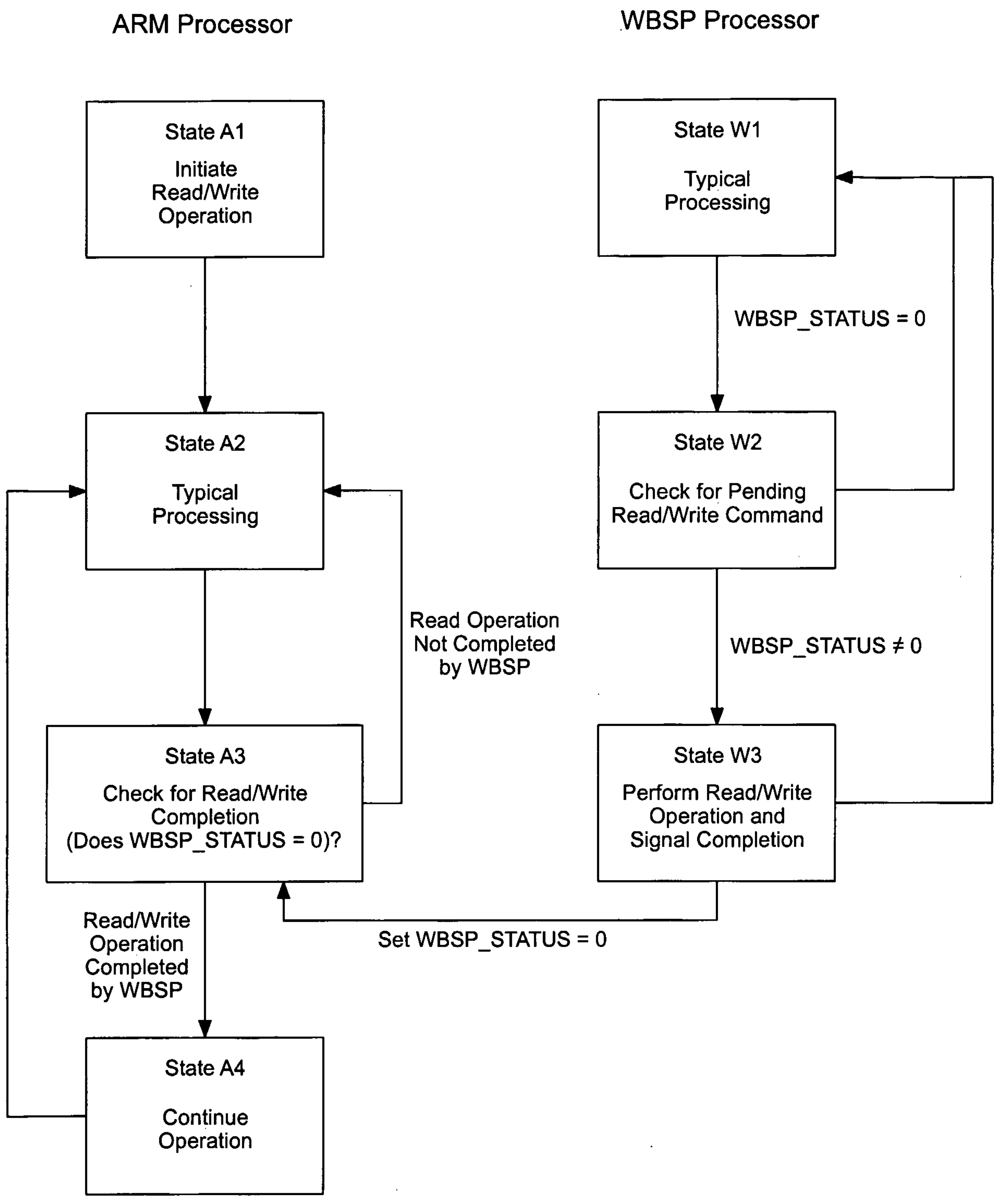

ARM Processor
WBSP Processor
State A1
Initiate Read/Write Operation
State A2
Typical Processing
Read Operation Not Completed by WBSP
State A3
Check for Read/Write Completion (Does WBSP_STATUS = 0)?
Read/Write Operation Completed by WBSP
State A4
Continue Operation
State W1
Typical Processing
WBSP_STATUS = 0
State W2
Check for Pending Read/Write Command
WBSP_STATUS ≠ 0
State W3
Perform Read/Write Operation and Signal Completion
Set WBSP_STATUS = 0

FIG. 7

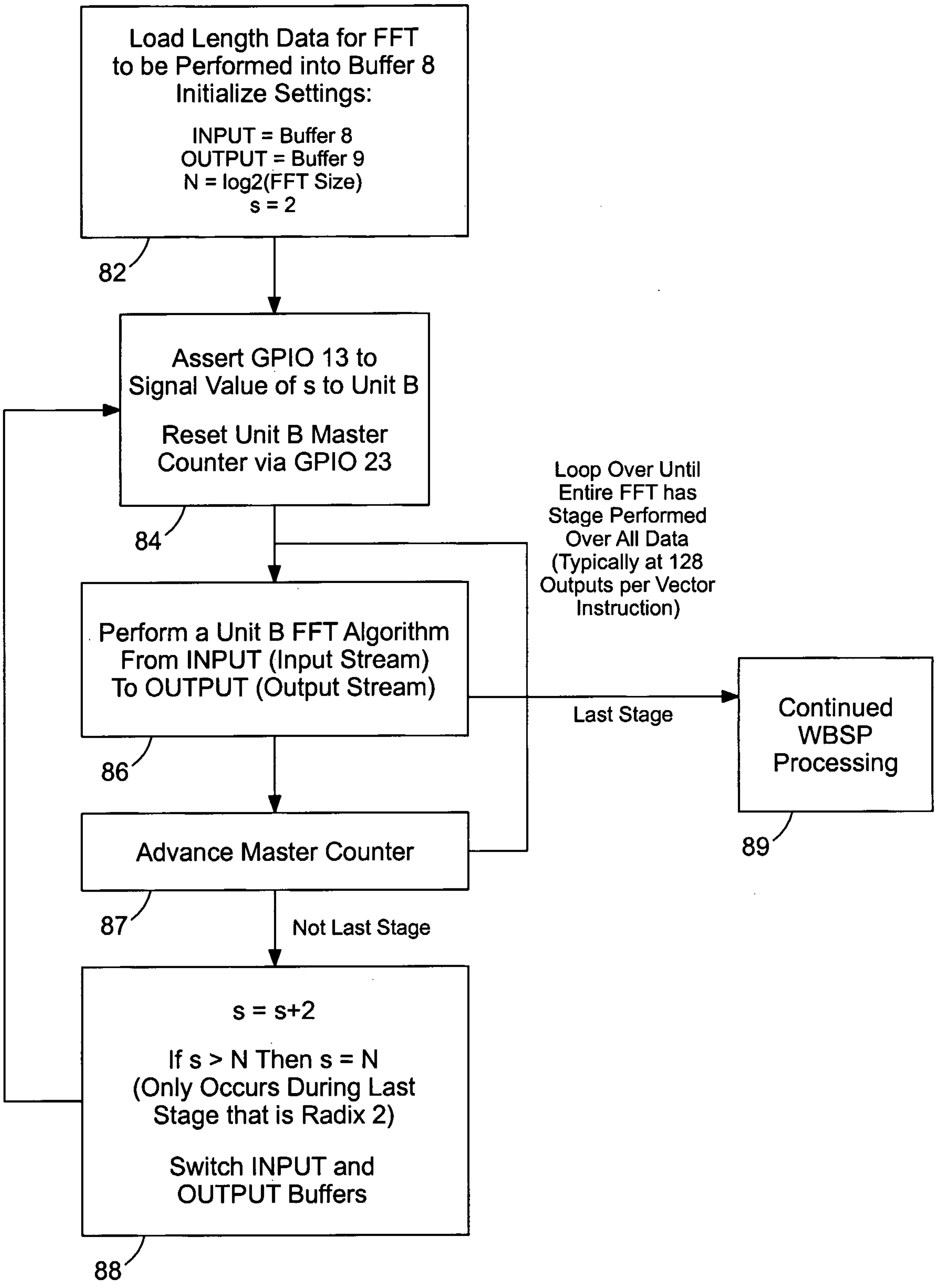

Load Length Data for FFT to be Performed into Buffer 8 Initialize Settings:
INPUT = Buffer 8
OUTPUT = Buffer 9
N = log2(FFT Size)
s = 2
82
Assert GPIO 13 to Signal Value of s to Unit B
Reset Unit B Master Counter via GPIO 23
84
Loop Over Until Entire FFT has Stage Performed Over All Data (Typically at 128 Outputs per Vector Instruction)
Perform a Unit B FFT Algorithm From INPUT (Input Stream) To OUTPUT (Output Stream)
86
Last Stage
Continued WBSP Processing
89
Advance Master Counter
87
Not Last Stage
s = s+2
If s > N Then s = N (Only Occurs During Last Stage that is Radix 2)
Switch INPUT and OUTPUT Buffers
88

FIG. 8

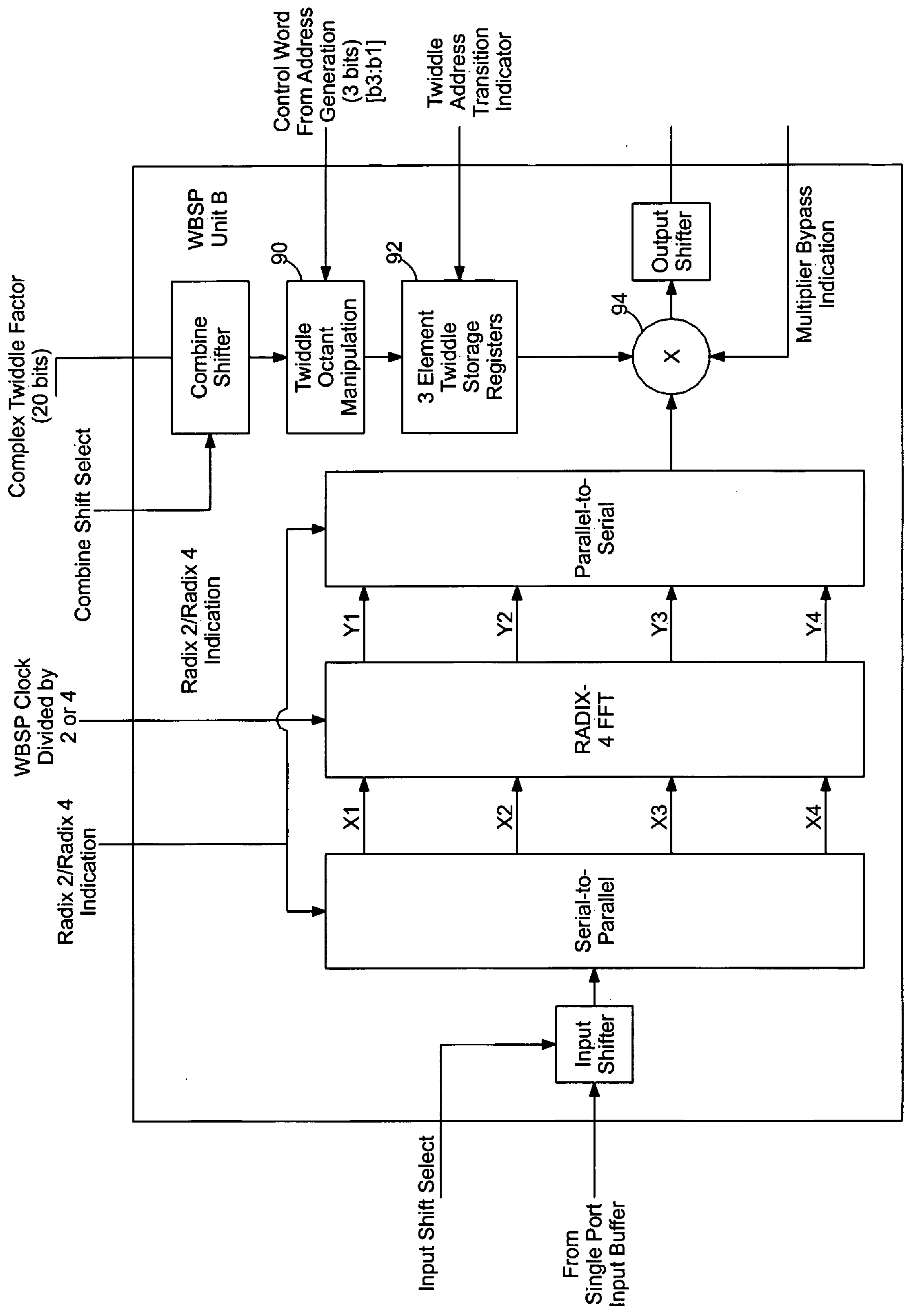

WBSP Unit B
Complex Twiddle Factor (20 bits)
Combine Shift Select
Combine Shifter
Control Word From Address Generation (3 bits) [b3:b1]
90
Twiddle Octant Manipulation
Twiddle Address Transition Indicator
92
3 Element Twiddle Storage Registers
94
X
Output Shifter
Multiplier Bypass Indication
WBSP Clock Divided by 2 or 4
Radix 2/Radix 4 Indication
Radix 2/Radix 4 Indication
Serial-to-Parallel
X1
X2
X3
X4
RADIX-4 FFT
Y1
Y2
Y3
Y4
Parallel-to-Serial
Input Shift Select
Input Shifter
From Single Port Input Buffer

FIG. 9

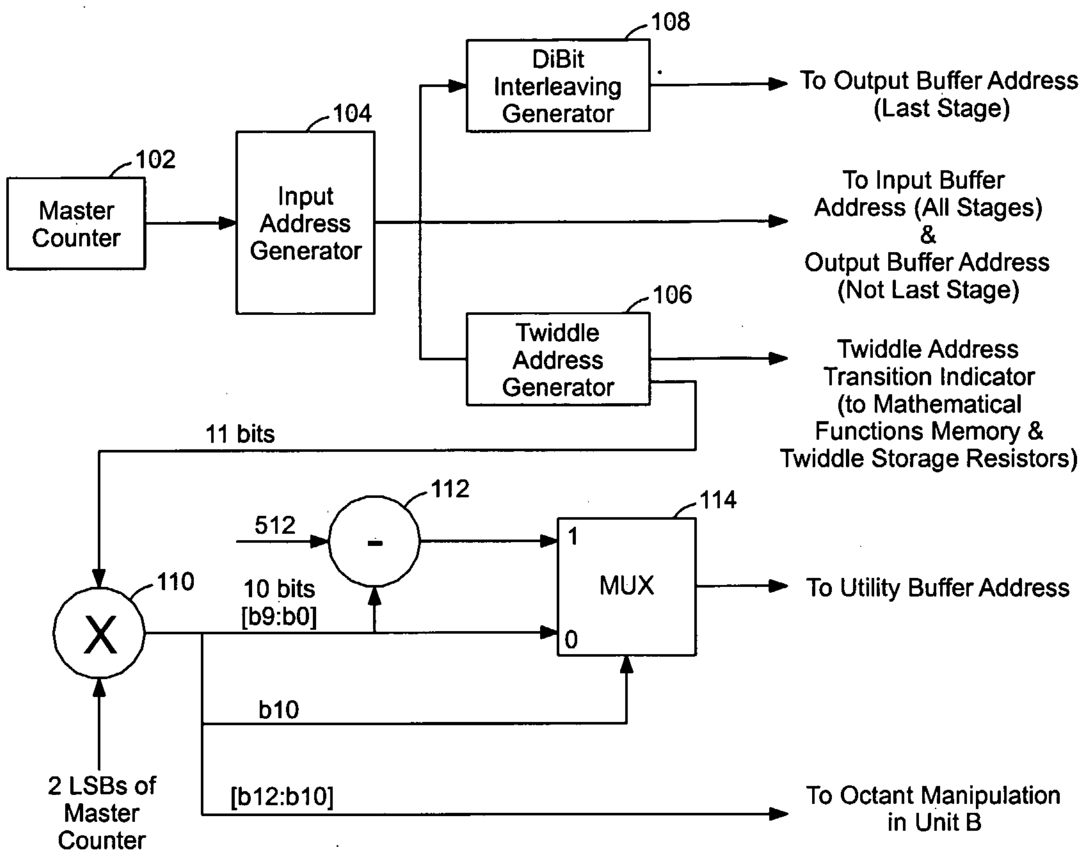

108
DiBit Interleaving Generator
To Output Buffer Address (Last Stage)
104
Input Address Generator
102
Master Counter
To Input Buffer Address (All Stages) & Output Buffer Address (Not Last Stage)
106
Twiddle Address Generator
Twiddle Address Transition Indicator (to Mathematical Functions Memory & Twiddle Storage Resistors)
11 bits
112
114
512
-
1
MUX
0
To Utility Buffer Address
110
X
10 bits [b9:b0]
b10
2 LSBs of Master Counter
[b12:b10]
To Octant Manipulation in Unit B

FIG. 10

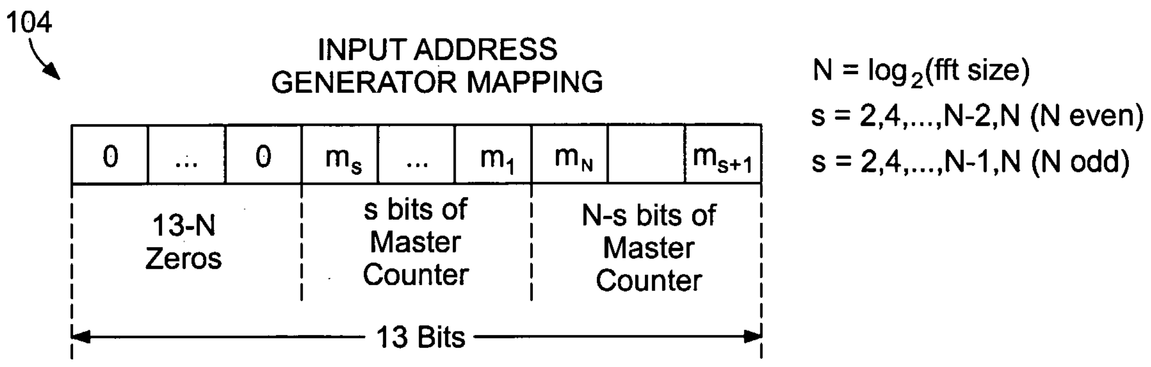

104
INPUT ADDRESS GENERATOR MAPPING
0
...
0
m_s
...
m_1
m_N
m_{s+1}
13-N Zeros
s bits of Master Counter
N-s bits of Master Counter
13 Bits
$N = \log_2$(fft size)
s = 2,4,...,N-2,N (N even)
s = 2,4,...,N-1,N (N odd)

FIG. 11

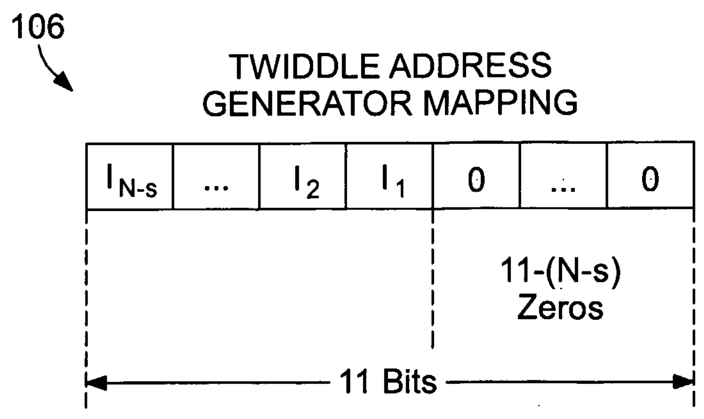

106
TWIDDLE ADDRESS GENERATOR MAPPING
I_{N-s}
...
I_2
I_1
0
...
0
11-(N-s) Zeros
11 Bits

FIG. 12

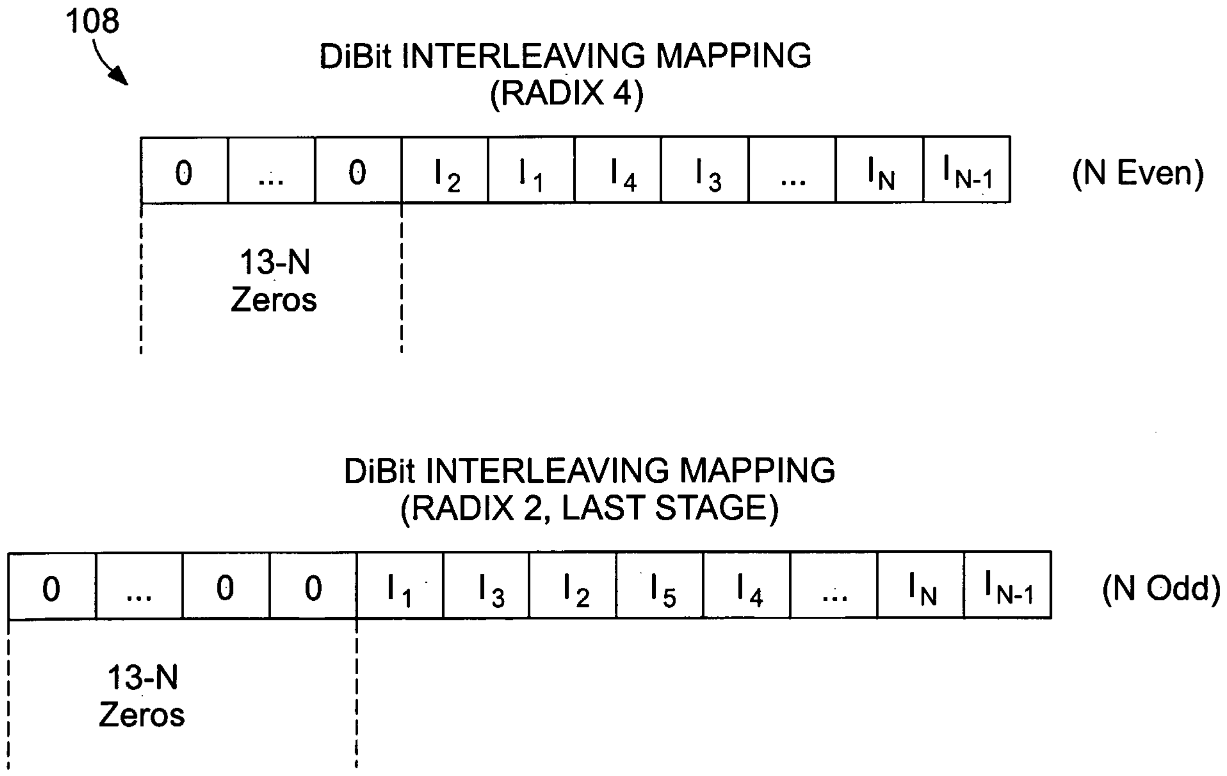

108
DiBit INTERLEAVING MAPPING (RADIX 4)
0
...
0
I_2
I_1
I_4
I_3
...
I_N
I_{N-1}
(N Even)
13-N Zeros
DiBit INTERLEAVING MAPPING (RADIX 2, LAST STAGE)
0
...
0
0
I_1
I_3
I_2
I_5
I_4
...
I_N
I_{N-1}
(N Odd)
13-N Zeros

FIG. 13

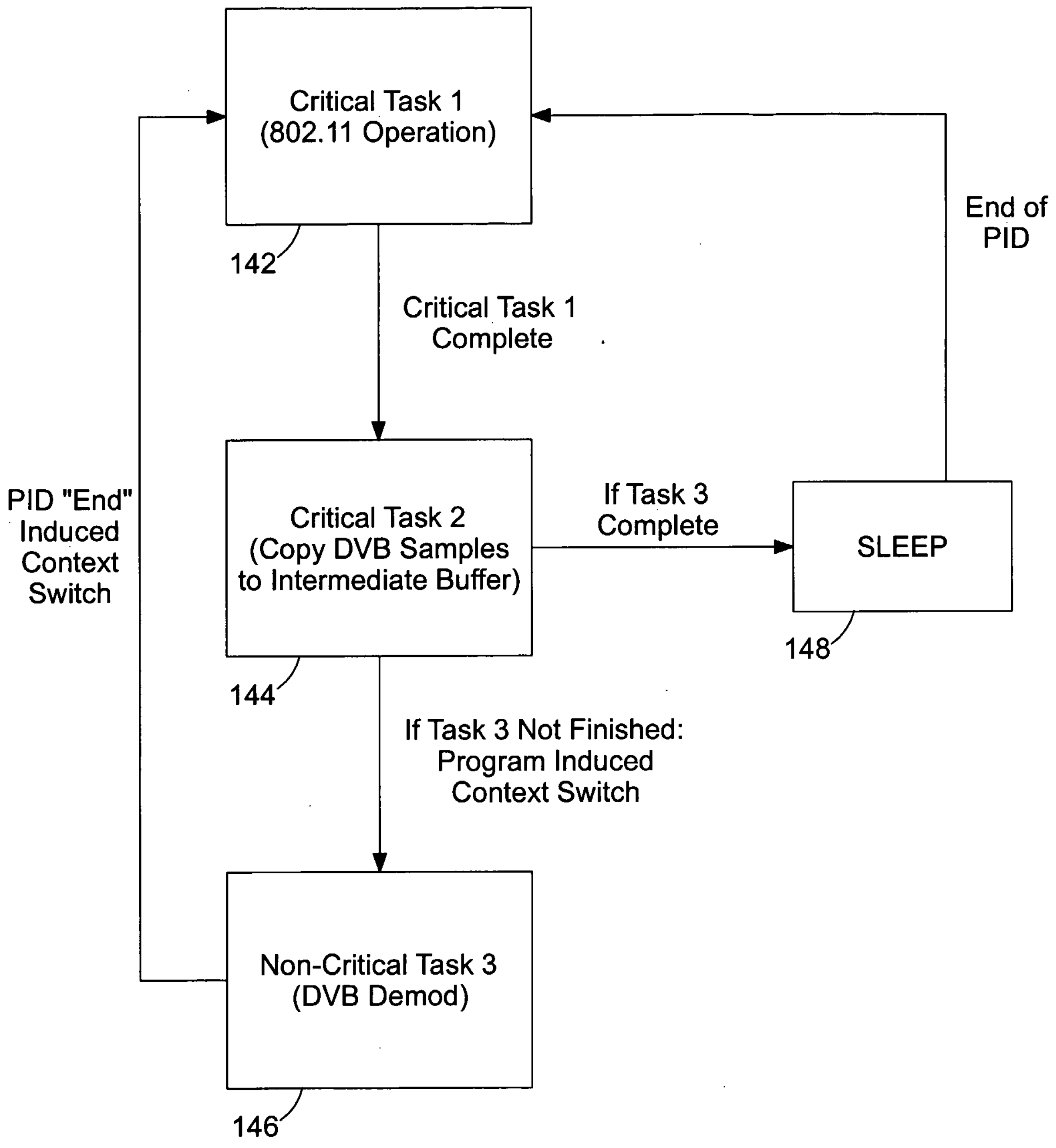

Critical Task 1
(802.11 Operation)
142
End of PID
Critical Task 1 Complete
PID "End" Induced Context Switch
Critical Task 2
(Copy DVB Samples to Intermediate Buffer)
If Task 3 Complete
SLEEP
148
144
If Task 3 Not Finished: Program Induced Context Switch
Non-Critical Task 3
(DVB Demod)
146

FIG. 14

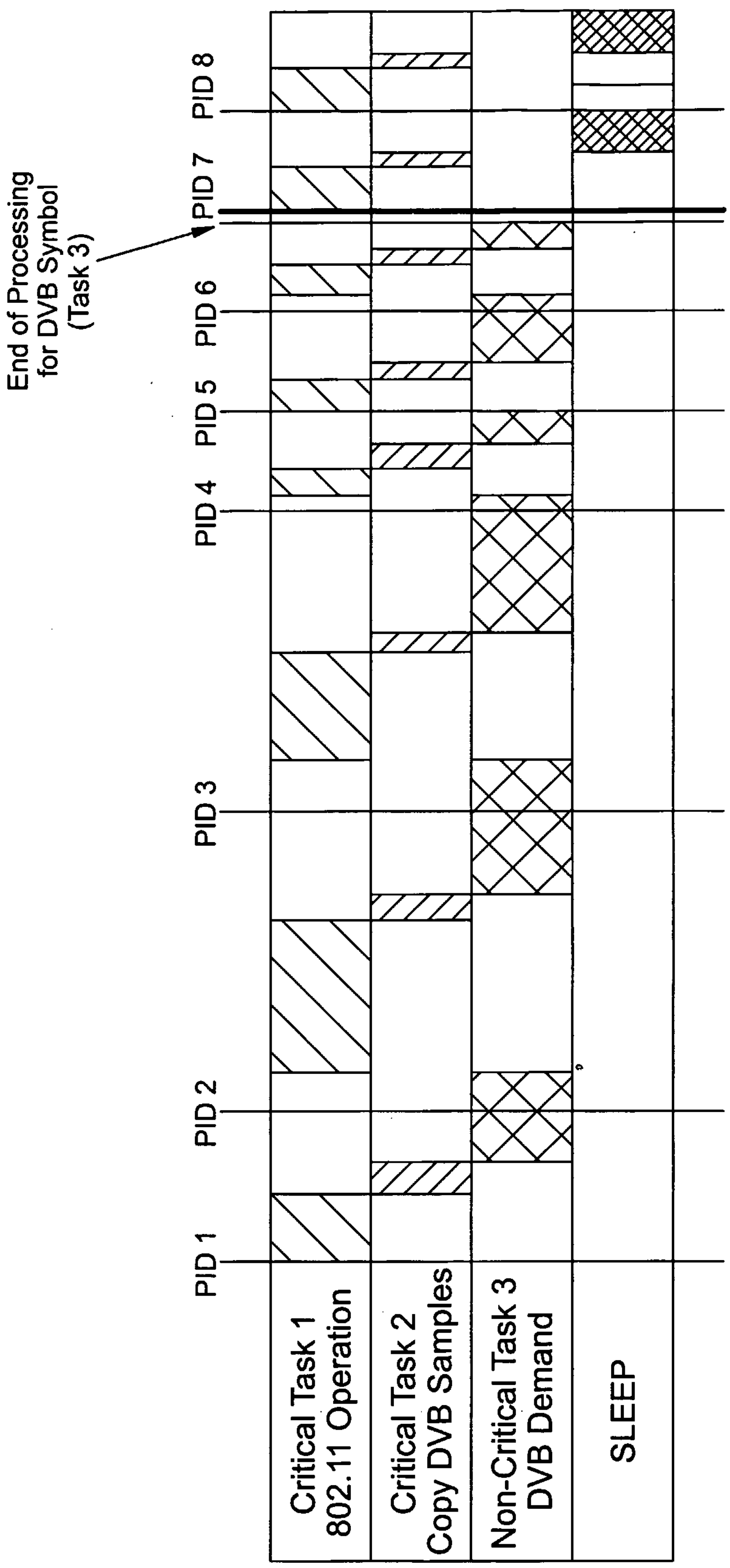

Critical Task 1
802.11 Operation
Critical Task 2
Copy DVB Samples
Non-Critical Task 3
DVB Demand
SLEEP
PID 1
PID 2
PID 3
PID 4
PID 5
PID 6
PID 7
PID 8
End of Processing
for DVB Symbol
(Task 3)

FIG. 15

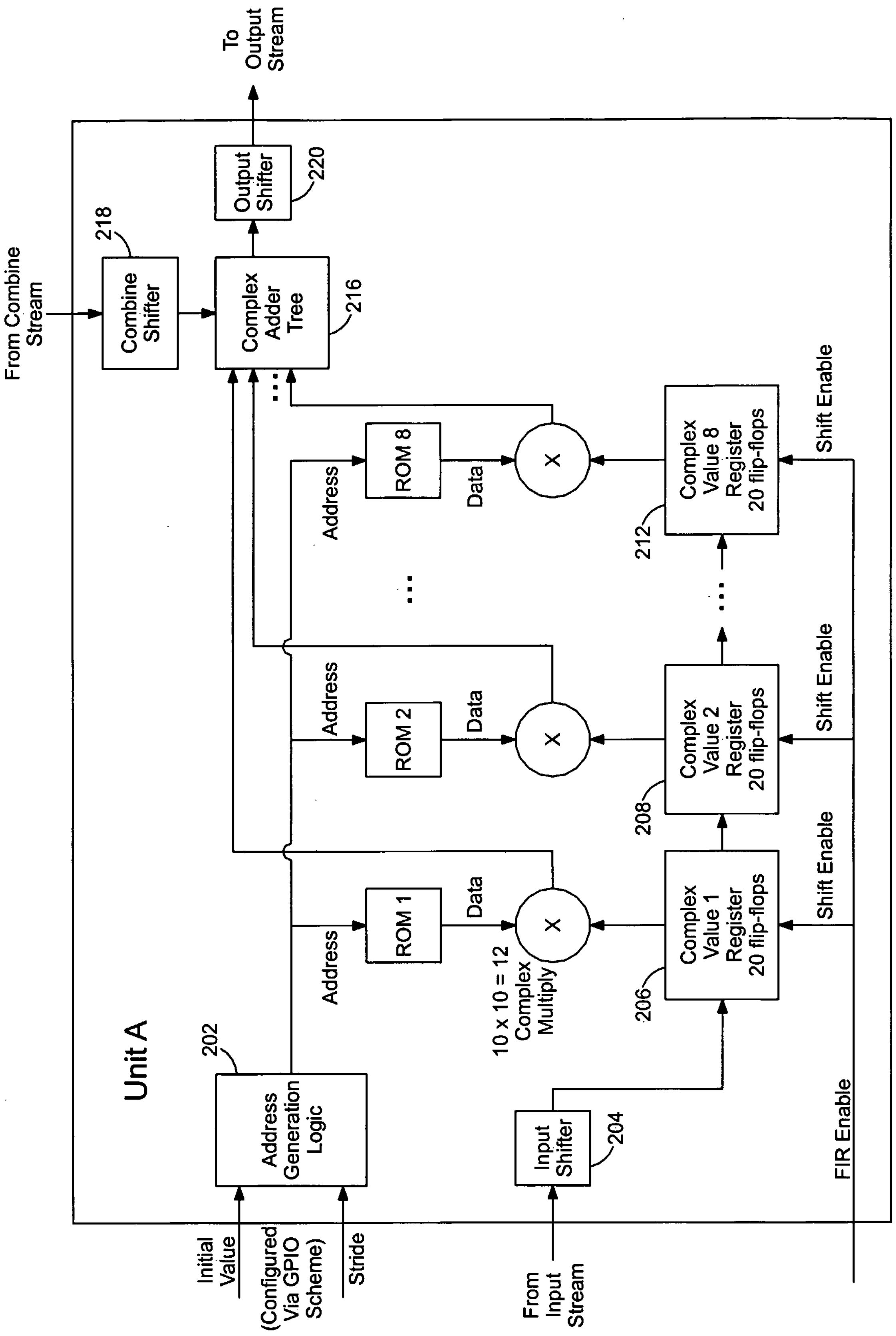

To Output Stream
From Combine Stream
218
Combine Shifter
Complex Adder Tree
216
Output Shifter
220
Unit A
202
Address Generation Logic
Initial Value (Configured Via GPIO Scheme)
Stride
Address
ROM 1
Data
10 x 10 = 12 Complex Multiply
ROM 2
ROM 8
X
Input Shifter
204
From Input Stream
Complex Value 1 Register 20 flip-flops
206
Complex Value 2 Register 20 flip-flops
208
Complex Value 8 Register 20 flip-flops
212
Shift Enable
FIR Enable

FIG. 16

Address Generation Logic Detail

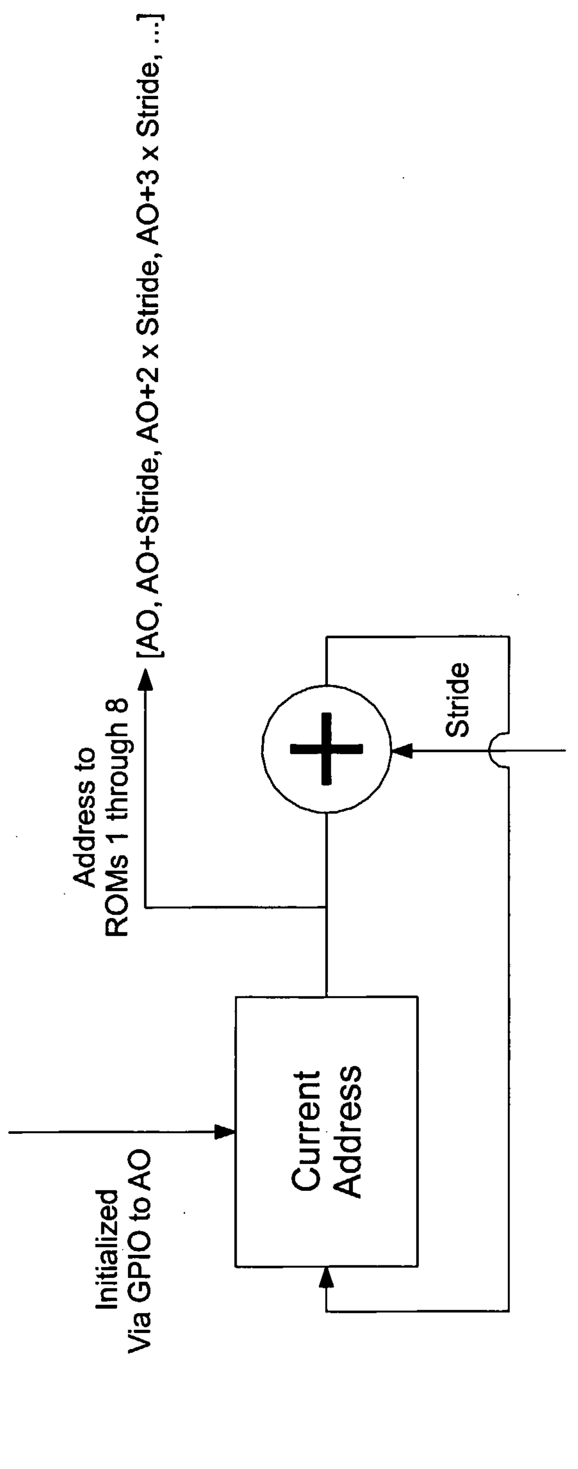


FIG. 17

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US05/32177

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 13/38 (3/00) (2006.01)
USPC - 710/52
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
US: 700/002, 004, 005, 007, 012, 019, 020-029; 710/052; 713/375, 400, 401, 501, 502, 503; 714/021, 740-769, 801-805; 719/313-315, 319-320, 718
IPC(8): G06F 005/*, 009/044, 009/046, 011/10, 011/16, 013/00, 013/38; G11C 029/00; H03M 013/00 (2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
MICROPAT; DIALOGPRO; IEEEXPLORE; GOOGLE SCHOLAR

Search terms: mailbox; memory; RAM; SRAM; dual port; clock; processor; shared; address

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6, 397,273 B2 (CHILTON) 28 May 2002 (28.05.2002) entire document	1-30
Y	US 2005/0044457 A1 (JEDDELOH) 24 February 2005 (24.02.2005) paragraphs 003, 004	1-30
Y	'Understanding Bandwidth and Latency' (STOKES) November 2002 (11.2002) ARS Technica pages 1-5	7, 10, 22, 25
A	US 5,884,055 A (TUNG et al) 16 March 1999 (16.03.1999) column 5, lines 16-32	6, 7, 10, 21, 22, 25
A	US 4,096,567 (MILLARD et al) 20 June 1978 (20.06.1978) entire document	1-30
A	US 2002/0116595 A1 (MORTON) 22 August 2002 (22.08.2002) entire document	1-30
A	US 6,810,308 B2 (SHAJII et al) 26 October 2004 (26.10.2004) entire document	1-30
A	US 5,220,668 A (BULLIS) 15 June 1993 (15.06.1993) entire document	1-30
A	US 6,880,070 B2 (GENTIEU et al) 12 April 2005 (12.04.2005) entire document	1-30
A	US 2004/0210797 A1 (KIMELMAN et al) 21 October 2004 (21.10.2004) entire document	1-30
A	US 5,483,640 A (ISFELD et al) 09 January 1996 (09.01.1996) entire document	1-30
A	US 6,785,892 B1 (MILLER et al) 31 August 2004 (31.08.2004) entire document	1-30

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier application or patent but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	"&" document member of the same patent family

Date of the actual completion of the international search 25 May 2006	Date of mailing of the international search report 27 JUL 2006
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Blaine R Copenheaver Telephone No. 571-272-7774

Supplemental Box:

Continuations of Box III:
This application contains the following inventions or groups of inventions which are not so linked as to form a single general inventive concept under PCT Rule 13.1. In order for all inventions to be examined, the appropriate additional examination fees must be paid.

Group I, claims 1-30, drawn to obtaining processor diagnostic data.
Group II, claims 31-39 and 41-49, drawn to controlling input and output in a multi-mode wireless processign system.
Group III, claim 40, drawn to a configuration of input and output componenets of a processing unit.
Group IV, claims 50-57, drawn to dynamically controlling rate connections to sample buffers.
Group V, claims 58-92, drawn to interfacing two processors.
Group VI, claims 93-129, drawn to a Fast Fourier Transform (FFT) method in a multi-mode wireless processign system.
Group VII, claims 130-139, drawn to switching between instruction contexts within a time interval.
Group VIII, claims 140-151, drawn to performing a convolution operation.

The inventions listed as Groups I, II, III, IV, V, VI, VII and VIII do not relate to a single general inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons: the special technical feature of the Group I invention is reading and writing to diagnostic memory using different frequencies as claimed therein, the special technical feature of the Group II invention is controlling an interface of a multi-mode wireless processign system by determining whether a designated processing unit generates output data or receives input data as claimed therein, the special technical feature of the Group III invention is a processor configuration having general purpose inputs and outputs as claimed therein, the special technical feature of the Group IV invention is controlling the rate a plurality of sample buffers serially connect to external elements of the system as claimed therein, the special technical feature of the Group V invention is performing read and write operations between two processors as claimed therein, the special technical feature of the Group VI invention is system that performs FFT using an input vector, counter and various stages of operations as claimed therein, the special technical feature of the Group VII invention is switching between instruction contexts within a time interval including executing critical and non-critical operations and entering a sleep mode as claimed therein and the special technical feature of the Group VIII invention is performing a convolution operation using a stride value and address generator as claimed therein.

Since none of the special technical features of the Group I, II, III, IV, V, VI, VII and VIII inventions is found in more than one of the inventions, unity of invention is lacking.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US05/32177

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:
See Supplemental Box

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:
1-30

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.