CYPRESS SEMICONDUCTOR

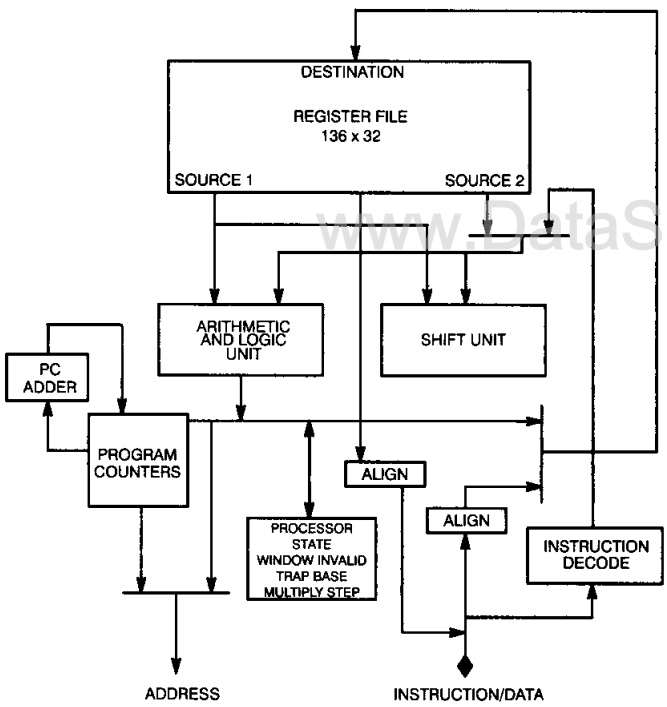
CY7C611A

32-Bit RISC Controller

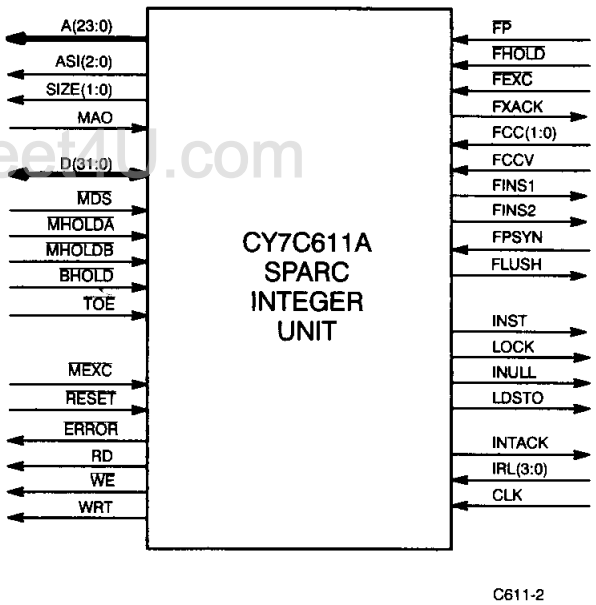
Features

- SPARC® processor optimized for embedded control applications
- Reduced Instruction Set Computer (RISC) architecture
  - Simple format instructions
  - Most instructions execute in a single cycle
- Very high performance
  - 40-ns instruction cycle with 4-stage pipeline
  - 18 sustained MIPS at 25 MHz
  - 240-ns worst-case interrupt response
- 136 32-bit registers
  - Eight overlapping windows of 24 registers each
  - Dividing registers into seperate register banks allows fast context switching
  - 8 global registers
- Hardware pipeline interlocks
- 16 prioritized interrupts levels
- Large address space
  - 24-bit address space
  - 3-bit address space indentifier
- Multitasking support
  - User/supervisor modes
  - Privileged instructions
- Artificial intelligence support
- Multiprocessing support
- High-performance floating-point processor interface
  - Concurrent execution of floating-point instructions
- 0.8-micron 2-layer metal CMOS technology
- 160-pin quad flat package
- Power
  - 3 watts maximum

Logic Block Diagram



Pin Configuration



Selection Guide

		CY7C611A-25
Maximum Operating Current (mA)	Commercial	600

SPARC is a registered trademark of SPARC International, Inc.

Overview

The CY7C611A controller is a high-speed CMOS implementation of the SPARC 32-bit RISC architecture processor optimized for embedded control applications. RISC architecture makes possible the creation of a processor which can execute instructions at a rate of one instruction per processor clock. The CY7C611A supports a tightly-coupled floating-point coprocessor capable of executing at a rate of 4 to 5 MFLOPS. The CY7C611A SPARC controller provides the following features:

Simple instruction format. All instructions are 32 bits wide and aligned on 32-bit boundaries in memory. Three basic instruction formats feature uniform placement of opcode and address fields.

Register intensive architecture. Most instructions operate on either two registers or one register and a constant, and place the result in a third register. Only load and store instructions access off-chip memory.

Large windowed register file. The processor has 136 on-chip 32-bit general purpose registers. Eight of these are global registers. The remaining 128 registers can be configured as four separate non-overlapping register banks or as eight overlapping sets of 24 registers each. The first configuration allows for extremely fast context switch times and the second provides for very low overhead procedure calls. The actual configuration and use of the registers is determined by the user's application.

Delayed control transfer. The processor always fetches the next instruction after a control transfer, and either executes it or annuls it depending on the state of a bit in the control transfer instruction. This feature allows compilers to rearrange code to place a useful instruction after a delayed control transfer and thereby take better advantage of the processor pipeline.

Concurrent floating point. Floating-point instructions can execute concurrently with each other and with non-floating-point instructions.

Fast interrupt response. Interrupt inputs are sampled on every clock cycle and can be acknowledged in one to three cycles. The first instruction of an interrupt service routine can be executed within six to eight cycles of receiving the interrupt request.

The 7C600 Family

The SPARC processor family consists of the CY7C601A and CY7C611A integer units and the CY7C602A floating-point unit. The CY7C601A and CY7C611A integer units are a high-speed implementation of the SPARC architecture, and are binary compatible with all SPARC processors. The CY7C602A is a high-performance floating-point unit that allows floating-point instructions to execute concurrently with the CY7C601A or the CY7C611A.

The CY7C611A is designed for embedded control and application specific systems. The CY7C611A communicates with external memory via a 24-bit address bus and a 32-bit data/instruction bus. In many dedicated controller applications, the CY7C611A can function by itself with high-speed local memory. The CY7C611A retains the signals supplied on the CY7C601A for discrete implementations of cache systems. The CY7C157A cache storage unit can be used with the CY7C611A to provide a zero wait-state memory system with no glue logic. The CY7C289 registered PROM provides a zero wait-state PROM memory for most accesses and requires no glue logic for interfacing to the CY7C611A.

Floating-Point Coprocessor Interface

The CY7C611A is the basic processing engine which executes all of the instruction set except for floating-point operations. The CY7C602A and CY7C611A operate concurrently. The CY7C602A recognizes floating-point instructions and places them in a queue while the CY7C611A continues to execute non-floating point instructions. If the CY7C602A encounters an instruction which will not fit in its queue, the CY7C602A holds the CY7C611A until the instruction can be stored. The CY7C602A contains its own set of registers on which it operates. The contents of these registers are transferred to and from external memory under control of the CY7C611A via floating-point load/store instructions. Processor interlock hardware hides floating-point concurrency from the compiler or assembly language programmer. A program containing floating-point computations generates the same results as if instructions were executed sequentially.

Multitasking Support

The CY7C611A supports a multitasking operating system by providing user and supervisor modes. Some instructions are privileged and can only be executed while the processor is in supervisor mode. Changing from user to supervisor mode requires taking a hardware interrupt or executing a trap instruction.

Interrupts and Traps

The CY7C611A supports both asynchronous traps (interrupts) and synchronous traps (error conditions and trap instructions). The occurrence of a trap causes the CY7C611A to fetch the beginning address of the trap routine from a trap table. The base address of the trap table is specified by a trap base register and the offset is a function of the trap type. After fetching the trap routine address, program control jumps to the trap routine. Traps are taken before the current instruction is executed and can therefore be considered to occur between instructions.

Registers

The following sections provide an overview of the CY7C611A registers. The CY7C611A has two types of registers; working registers (r registers), and control registers. The r registers provide storage for processes, and the control registers keep track of and control the state of the CY7C611A.

Special r Registers. The utilization of four r registers is partially fixed by the instruction set. Global register r[0] is dummy register; it returns the value "0" when it is used as a source register, and it is not modified when used as a destination register. This feature makes the most common value easily available and eliminates the need for a clear register instruction. Another r register fixed by the instruction set is r[15]. Upon executing a CALL instruction, the address of the CALL instruction is written into r[15]. Upon entering a trap routine, registers r[17] and r[18] contain the PC and nPC.

r Register Addressing. r registers r8 through r31 are addressed internally using the register number and current window pointer (CWP) field of the processor status register (PSR; see next section). The CWP is essentially an index field for r register addressing, and acts as a pointer to a group of 24 registers. *Figure 1*

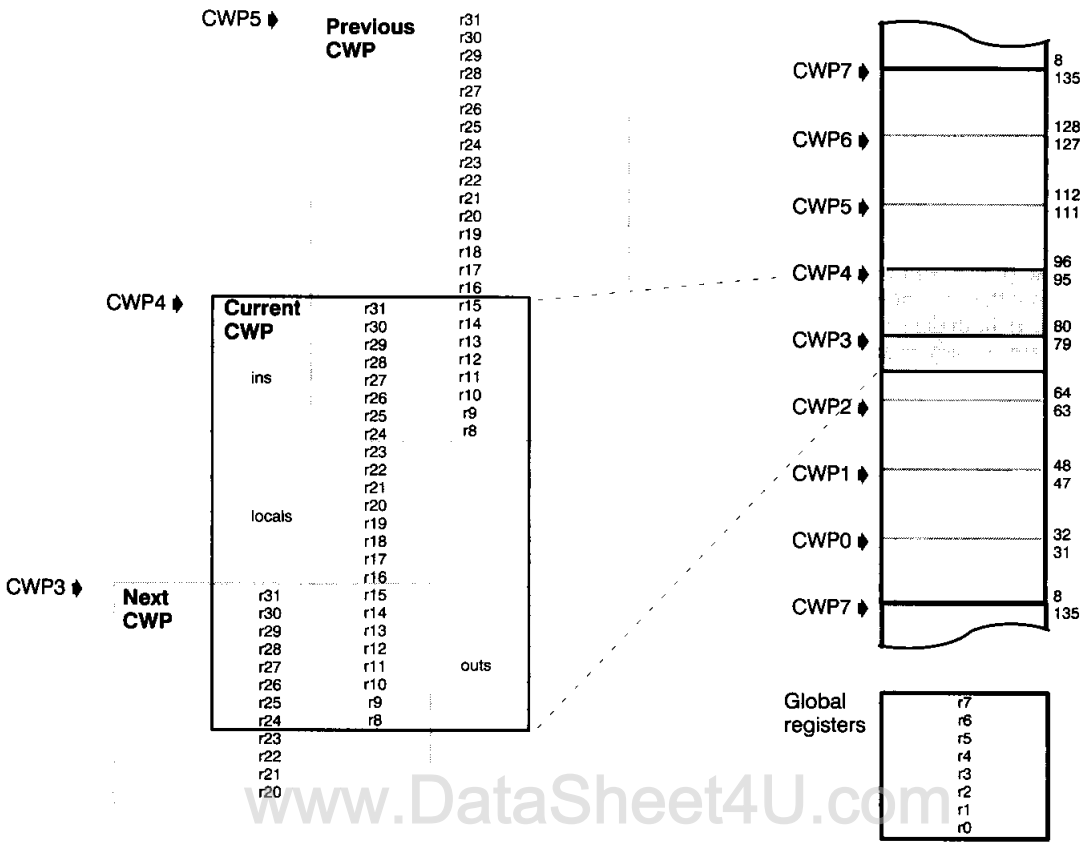


Figure 1. CWP Register Addressing

Registers (continued)

illustrates r register addressing using the CWP. Incrementing or decrementing the CWP changes the register offset by 16, thereby causing the register addressing to overlap by eight registers. Thisallows r24 through r31 of the current window to act as r8 through r15 of the previous window. Registers r0 through r7 do not use the CWP to address them, therefore they are global in nature.

The window invalid mask register (WIM) is used to disallow selected CWP values. Each bit of the least significant byte of the WIM register corresponds to a register window or CWP value. Incrementing or decrementing the CWP to a window invalidated by the WIM register causes the CY7C611A to cause a window underflow or window overflow trap. This is used in a register window environment to set the boundaries for software. The WIM register can also be used to set boundaries for register banks in a bank switching environment.

CY7C611A Control Registers. The CY7C611A's control registers contain various addresses and pointers used by the system to control its internal state. They include the program counters (PC and nPC), the processor state register (PSR), the window invalid mask register (WIM), the trap base register (TBR), and the Y register. The following paragraphs briefly describe each:

Processor Status Register (PSR). The processor status register contains fields that describe and control the state of the CY7C611A. *Figure 2* illustrates the bit assignments for the PSR.

IU Implementation and IU Version Numbers. These are read-only fields in the PSR. The version number is set to "0001" and the implementation number is set to binary "0011".

Integer Condition Codes. The integer condition codes consist of four flags: negative, zero, overflow, and carry. These flags are set by the conditions occurring during integer logic and arithmetic operations.

Enable Floating-Point Unit (EF bit). This bit is used to enable the floating-point unit. If a floating-point operation (FPop) is encountered and the EF bit is cleared (i.e., FPU disabled), a floating-point disabled trap is generated.

Processor Interrupt Level (PIL). This four bit field sets the CY7C611A interrupt level. The CY7C611A will only acknowledge interrupts greater than the level indicated by the PIL field. Bit 11 is the MSB; bit 8 is the LSB.

Supervisor Mode (S). S = 1 indicates that the CY7C611A is in supervisor mode. Supervisor mode can only be entered by a software or hardware trap.

Previous Supervisor Mode (PS). This bit indicates the state of the supervisor bit before the most recent trap.

Trap Enable (ET). This bit enables or disables the CY7C611A traps. This bit is automatically set to 0 (traps disabled) upon entering a trap. When ET = 0, all asynchronous traps are ignored.

If a synchronous trap occurs when ET = 0, the CY7C611A enters error mode.

Current Window Pointer (CWP). The r registers are addressed by the Current Window Pointer (CWP), a field of the Processor Status Register (PSR) that points to the 24 active local registers. It is incremented by a RESTORE instruction and decremented by a SAVE instruction. Note that the globals are always accessible regardless of the CWP. In the overlapping configuration each window shares its ins and outs with adjacent windows. The outs from a previous window (CWP +1) are the ins of the current window, and the outs from the current window are the ins for the next window (CWP −1). In both the windowed and register bank configurations globals are equally available and the locals are unique to each window.

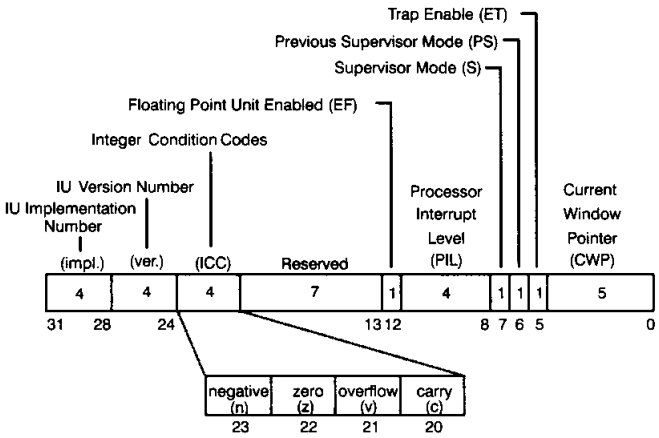


Figure 2. Processor State Register

Program Counters (PC and nPC). The program counter (PC) holds the address of the instruction being executed, and the next program counter (nPC) holds the address of the next instruction to be executed.

Trap Base Register (TBR). The trap base register contains the base address of the trap table and a field that provides a pointer into the trap table.

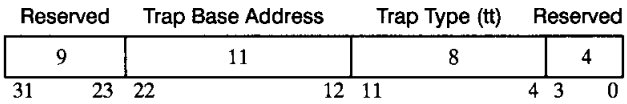


Figure 3. Trap Base Register

Window Invalid Mask Register (WIM). The window invalid mask register determines which windows are valid and which window accesses cause window_overflow and window_underflow traps.

Y register. The Y register is used to hold the partial product during execution of the multiply-step instruction (MULSCC).

Pin Description

The integer unit's external signals fall into three categories:

1. memory subsystem interface signals,
2. floating-point unit interface signals, and
3. miscellaneous I/O signals.

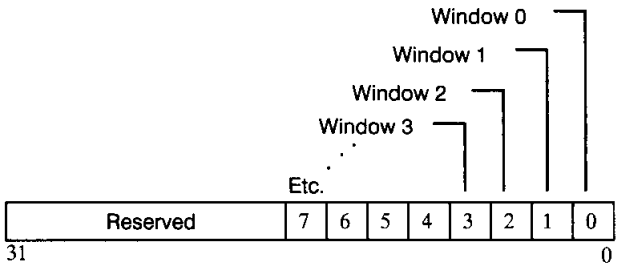


Figure 4. Window Invalid Mask

These are described in the following sections. Paragraphs after the tables describe each signal. Signals that are active LOW are marked with an overbar; all others are active HIGH. For example, $\overline{WE}$ is active LOW, while RD is active HIGH.

Memory Subsystem Interface Signals

The memory interface signals consist of 27 bits of address (24 bits of address and a three-bit address space identifier), 32 bits of bidirectional data lines, and two bits to identify the size (byte, halfword, word, or double word) of data bus transactions.

A[23:0]—These 24 bits are the addresses of instructions or data and they are sent out "unlatched" by the CY7C611A. Assertion of the MAO signal during a cache miss will force the integer unit to put the previous (missed) address on the address bus. A[23:0] pins are three-stated if the $\overline{TOE}$ signal is deasserted.

ASI[2:0]—These three bits are the address space identifier for an instruction or data access to the memory. ASI[2:0] are sent out "unlatched" by the integer unit. The value on these pins during any given cycle is the address space identifier corresponding to the memory address on the A[23:0] pins at that cycle. Assertion of the MAO signal during a cache miss will force the integer unit to put the previous address space identifier on the ASI[2:0] pins. ASI[2:0] pins are three-stated if the $\overline{TOE}$ signal is deasserted. Normally, the encoding of the ASI bits is as shown in *Table 1*. The remaining codes are software generated.

Table 1. ASI Bit Assignment

Address Space Identifier (ASI)	Address Space
000	User Instruction
010	User Data
001	Supervisor Instruction
011	Supervisor Data

D[31:0]—D[31:0] is the bidirectional data bus to and from the integer unit. The data bus is driven by the integer unit during the execution of integer store instructions and the store cycle of atomic load/store instructions. Similarly, the data bus is driven by the floating-point unit only during the execution of floating-point store instructions. The store data is sent out unlatched and must be latched externally before it is used. Once latched, store data is valid during the second data cycle of a store single access, the second and third data cycle of a store double access, and the third data cycle of an atomic load store access. The alignment for load and store instructions is done inside the processor. A double word is aligned on an eight-byte boundary, a word is aligned on a four-byte boundary, and a half word is aligned on a two-byte boundary. D(31) corresponds to the most significant bit of the least

significant byte of the 32-bit word. If a double-word, word, or half-word load or store instruction generates an improperly aligned address, a memory address not aligned trap will occur.

Memory Subsystem Interface Signals (continued)

Instructions and operands are always expected to be fetched from a 32-bit wide memory.

SIZE[1:0]. These two bits specify the data size associated with a data or instruction fetch. Size bits are sent out "unlatched" by the CY7C611A. The value on these pins at any given cycle is the data size corresponding to the memory address on the A[23:0] pins in that cycle. SIZE[1:0] remains valid on the bus during all data cycles of loads, stores, load_doubles, store_doubles and atomic load stores. Since all instructions are 32-bits long, SIZE[1:0] is set to "10" during all instruction fetch cycles. Encoding of the SIZE[1:0] bits is shown in *Table 2.*

Table 2. Size Bit Assignment

SIZE1	SIZE0	Data Transfer Type
0	0	Byte
0	1	Halfword
1	0	Word
1	1	Word (Load/Store Double)

$\overline{\text{MHOLDA}}$ or $\overline{\text{MHOLDB}}$. The processor pipeline will be frozen while $\overline{\text{MHOLDA}}$ is asserted and the CY7C611A outputs will revert to and maintain the value they had at the rising edge of the clock in the cycle before $\overline{\text{MHOLDA}}$ was asserted. $\overline{\text{MHOLDA}}$ is used to freeze the clock to both the integer and floating-point units during a cache miss (for systems with cache) or when a slow memory is accessed. This signal must be presented to the processor chip at the beginning of each processor clock cycle and be stable during the high time of the processor clock. Either $\overline{\text{MHOLDA}}$ or $\overline{\text{MHOLDB}}$ can be used for stopping the processor during a cache miss or memory exception. $\overline{\text{MHOLDB}}$ has the same definition as $\overline{\text{MHOLDA}}$. The processor hardware uses the logical "OR" of all hold signals (i.e., $\overline{\text{MHOLDA}}$, $\overline{\text{MHOLDB}}$, and $\overline{\text{BHOLD}}$) to generate a final hold signal for freezing the processor pipeline. All HOLD signals are latched (transparent latch) in the CY7C611A before they are used.

$\overline{\text{BHOLD}}$. $\overline{\text{BHOLD}}$ is asserted by the I/O controller when an external bus master requests the data bus. Assertion of this signal will freeze the processor pipeline. External logic should guarantee that after deassertion of $\overline{\text{BHOLD}}$, the data at all inputs to the chip is the same as what it was before $\overline{\text{BHOLD}}$ was asserted. This signal must be presented to the processor chip at the beginning of each processor clock cycle and be stable during the high time of the processor clock since the CY7C611A processes the $\overline{\text{BHOLD}}$ input through a transparent latch before it is used. $\overline{\text{BHOLD}}$ should be used only for bus access requests by an external device since the $\overline{\text{MDS}}$ and $\overline{\text{MEXC}}$ signals are not recognized while this input is active. $\overline{\text{BHOLD}}$ should not be deasserted while LOCK is asserted.

$\overline{\text{MDS}}$. Assertion of this signal will enable the clock input to the on-chip instruction register (during an instruction fetch) or to the load result register (during a data fetch). In a system with cache, $\overline{\text{MDS}}$ is used to signal the processor when the missed data (cache miss) is ready on the bus. In a system with slow memories, $\overline{\text{MDS}}$ is used to signal the processor when the read data is available on the bus. $\overline{\text{MDS}}$ must be asserted only while the processor is frozen by either the $\overline{\text{MHOLDA}}$ or $\overline{\text{MHOLDB}}$ input signals. The CY7C611A samples the $\overline{\text{MDS}}$ signal via an on-chip transparent latch before it is used. The $\overline{\text{MDS}}$ signal is also used for strobing memory exceptions. In other words, $\overline{\text{MDS}}$ should be asserted whenever $\overline{\text{MEXC}}$ is asserted (see $\overline{\text{MEXC}}$ definition).

$\overline{\text{MEXC}}$. This signal is asserted by the memory (or cache) controller to initiate an instruction (or data) exception trap. $\overline{\text{MEXC}}$ is latched in the processor at the rising edge of CLK and is used in the following cycle. If $\overline{\text{MEXC}}$ is asserted during an instruction fetch cycle, an instruction access exception is generated, and if $\overline{\text{MEXC}}$ is asserted during a data fetch cycle, a data access exception trap is generated. The $\overline{\text{MEXC}}$ signal is used during ($\overline{\text{MHOLD}}$) in conjunction with the $\overline{\text{MDS}}$ signal to indicate to the CY7C611A that the memory system was unable to supply valid instruction or data. If $\overline{\text{MDS}}$ is applied without $\overline{\text{MEXC}}$, the CY7C611A accepts the contents of the data bus as valid information, but when $\overline{\text{MDS}}$ is applied with $\overline{\text{MEXC}}$ an exception trap is generated and the contents of the data bus is ignored by the CY7C611A. (In other words, $\overline{\text{MHOLD}}$ and $\overline{\text{MDS}}$ must be LOW when $\overline{\text{MEXC}}$ is asserted.) $\overline{\text{MEXC}}$ must be deasserted in the same clock cycle in which $\overline{\text{MHOLD}}$ is released.

RD. This signal specifies whether the current memory access is a read or write operation. It is sent out "unlatched" by the integer unit and must be latched externally before it is used. RD is set to "0" only during address cycles of store instructions including the store cycles of atomic load store instructions. This signal, when used in conjunction with SIZE[1:0] and LDSTO, can be used to check access rights of bus transactions. In addition, the RD signal may be used to turn off the output drivers of data RAMs during a store operation. For atomic load store instructions the RD signal is "1" during the first address cycle (read cycle), and "0" during the second and third address cycles (write cycle).

$\overline{\text{WE}}$. This signal is asserted by the integer unit during the second address cycle of store single instructions, the second and third address cycles of store double instructions, and the the third data cycle of atomic load/store instructions. The $\overline{\text{WE}}$ signal is sent out "unlatched" and must be latched externally before it is used. The $\overline{\text{WE}}$ signal may be externally qualified by HOLD signals (i.e., $\overline{\text{MHOLDA}}$ and $\overline{\text{MHOLDB}}$) to avoid writing into the memory during memory exceptions.

WRT. This signal is asserted (set to "1") by the processor during the first address cycle of single or double integer store instructions, the first data cycle of single or double floating-point store instructions, and the second data cycle of atomic load/store instructions. WRT is sent out "unlatched" and must be latched externally before it is used.

LDSTO. This signal is asserted by the integer unit during the data cycles of atomic load store operations. LDSTO is sent out "unlatched" by the integer unit and must be latched externally before it is used.

LOCK. This signal is set to "1" when the processor needs the bus for multiple cycle transactions such as atomic load/store, double loads and double stores. The LOCK signal is sent "unlatched" and should be latched externally before it is used. The bus may not be granted to another bus master as long as the LOCK signal is asserted (i.e., $\overline{\text{BHOLD}}$ should not be asserted in the following processor clock cycle when LOCK=1).

INULL. Assertion of INULL indicates that the current memory

Memory Subsystem Interface Signals (continued)

access (whose address is held in an external latch) is to be nullifiedby the processor. INULL is intended to be used to disable cachemisses (in systems with cache) and to disable memory exception generation for the current memory access (i.e., $\overline{MDS}$ and $\overline{MEXC}$ should not be asserted for a memory access when INULL=1). INULL is a latched output and is active during the same cycle as the address which it nullifies. INULL is asserted under the following conditions: During the second cycle of a store instruction, or whenever the CY7C611A address is invalid due to an external or internal exception. If a floating-point unit or coprocessor unit is present in the system INULL should be ORed with the FNULL and CNULL signals from these units.

Floating-Point Interface Signals

The floating-point/coprocessor unit interface is a dedicated group of connections between the CY7C611A and the CY7C602A. Note that no external circuits are required between the CY7C611A and the CY7C602A; all traces should connect directly. The interface consists of the following signals:

$\overline{FP}$. This signal indicates whether or not a floating-point unit exists in the system. The $\overline{FP}$ signal is normally pulled up to VDD by a resistor. It is grounded when the CY7C602A chip is present. The integer unit generates a floating-point disable trap if $\overline{FP}$ = 1 during the execution of a floating-point instruction, FBfcc instruction or floating-point load and store instructions.

FCC[1:0]. These bits are taken as the current condition code bits of the CY7C602A. They are considered valid if FCCV=1. During the execution of the FBfcc instruction, the processor uses these bits to determine whether the branch should be taken or not. FCC[1:0] are latched by the processor before they are used.

FCCV. This signal should be asserted only when the FCC[1:0] bits are valid. The floating-point unit deasserts FCCV if pending floating-point compare instructions exist in the floating-point queue. FCCV is reasserted when the compare instruction is completed and the floating-point condition codes FCC[1:0] are valid. The integer unit will enter a wait state if FCCV is deasserted (i.e., FCCV = "0"). The FCCV signal is latched (transparent latch) in the CY7C611A before it is used.

$\overline{FHOLD}$. This signal is asserted by the floating-point unit if a situation arises in which the CY7C602A cannot continue execution. The floating-point unit checks all dependencies in the Decode stage of the instruction and asserts $\overline{FHOLD}$ (if necessary) in the next cycle. This signal is used by the integer unit to freeze the instruction pipeline in the same cycle. The CY7C602A must eventually deassert $\overline{FHOLD}$ in order to unfreeze the integer unit's pipeline. The $\overline{FHOLD}$ signal is latched (transparent latch) in the CY7C611A before it is used.

$\overline{FEXC}$. Assertion of this signal indicates that a floating-point exception has occurred. $\overline{FEXC}$ must remain asserted until the integer unit takes the trap and acknowledges the CY7C602A via FXACK signal. Floating-point exceptions are taken only during the execution of floating-point instructions, FBfcc instruction and floating-point load and store instructions. $\overline{FEXC}$ is latched in the integer unit before it is used. The CY7C602A should deassert $\overline{FHOLD}$ if it detects an exception while $\overline{FHOLD}$ is asserted. In this case $\overline{FEXC}$ should be asserted a cycle before $\overline{FHOLD}$ is deasserted.

INST. This signal is asserted by the integer unit whenever a new instruction is being fetched. It is used by the CY7C602A to latch the instruction on the D[31:0] bus into the CY7C602A instruction buffer. The CY7C602A needs two instruction buffers (D1 and D2) to save the last two fetched instructions. When INST is asserted a new instruction enters into the D1 buffer and the old instruction in D1 enters into the D2 buffer.

FLUSH. This signal is asserted by the integer unit and is used by the CY7C602A to flush the instructions in its instruction registers. This may happen when a trap is taken by the integer unit. Instructions that have entered into the floating-point queue may continue their execution if FLUSH is raised as a result of a trap or exception other than floating-point exceptions.

FINS1. This signal is asserted by the integer init during the decode stage of a CY7C602A instruction if the instruction is in the D1 buffer of the CY7C602A chip. The CY7C602A uses this signal to latch the instruction in D1 buffer into its execute stage instruction register.

FINS2—This signal is asserted by the integer unit during the decode stage of a CY7C602A instruction if the instruction is in the D2 buffer of the CY7C602A chip. The CY7C602A uses this signal to latch the instruction in D2 buffer into its execute stage instruction register.

FXACK—This signal is asserted by the integer unit in order to acknowledge to the CY7C602A that the current $\overline{FEXC}$ trap is taken. The CY7C602A must deassert $\overline{FEXC}$ after it receives an asserted level of FXACK signal so that the next floating-point instruction does not cause a "repeated" floating-point exception trap.

Miscellaneous I/O Signals

These signals are used by the CY7C611A to control external events or to receive input from external events. This interface consists of the following signals:

IRL[3:0]. The data on these pins defines the external interrupt level. IRL[3:0]=0000 indicates that no external interrupts are pending. The integer unit uses two on-chip synchronizing latches to sample these signals on the rising edge of CLK. A given interrupt level must remain valid for at least two consecutive cycles to be recognized by the integer unit. IRL[3:0]=1111 signifies an non-maskable interrupt. All other interrupt levels are maskable by the PIL field of the Processor State Register (PSR). External interrupts should be latched and prioritized by the external logic before they are passed to the integer unit. The external interrupt latches should keep the interrupts pending until they are taken (and acknowledged) by the integer unit. External interrupts can be acknowledged by software or by the Interrupt Acknowledge (INTACK) output.

INTACK—This signal is asserted by the integer unit when an external interrupt is taken.

$\overline{RESET}$—Assertion of this pin will reset the integer unit. The $\overline{RESET}$ signal must be asserted for a minimum of eight processor clock cycles. After a reset, the integer unit will start fetching from address 0. The $\overline{RESET}$ signal is latched by the integer unit before it is used.

$\overline{ERROR}$—This signal is asserted by the integer unit when a trap is encountered while traps are disabled via the ET bit in the PSR.

Miscellaneous I/O Signals (continued)

In this situation the integer unit saves the PC and nPC registers, sets the tt value in the TBR, enters into an error state, asserts the $\overline{\text{ERROR}}$ signal and then halts. The only way to restart the processor trapped in the error state, is to trigger a reset by asserting the $\overline{\text{RESET}}$ signal.

$\overline{\text{TOE}}$—This signal is used to force all output drivers of the processor chip into a high-impedance state. It is used to isolate the chip from the rest of the system for debugging purposes. *This pin should be tied LOW for normal operation.*

FPSYN—This pin is a mode pin which is used to allow execution of additional instructions in future designs. It should be normally kept deasserted (FPSYN=0) to disable the execution of these instructions.

Document #: 38–R–10003–A

CLK—CLK is a 50% duty-cycle clock used for clocking the CY7C611A's pipeline registers. It is HIGH during the first half of the processor cycle, and LOW during the second half. The rising edge of CLK defines the beginning of each pipeline stage in the CY7C611A chip.