Sierra Circuit Design, Inc.

65C02-COMPATIBLE HIGH PERFORMANCE
8-bit MICROPROCESSOR CORE-CELL IP-65C02

IP-65C02 User's Guide

Rev 1.0 7/16/1998

Overview

MOS Technologies, Inc. first produced the original 6502 processor in 1975. It was later redesigned to fix some design quirks and to add a few instructions. This new design was designated the 65C02. The strengths of the 6502/65C02 architecture are in its small size, flexible-addressing modes and its relatively orthogonal instruction set. These same features make it an excellent choice for an embedded core.

The IP-65C02 is a synthesizable 65C02-compatible core was independently designed from public domain and commercially available data sheets and books. The underlying micro-architecture of the core is different from the original design. This allows many instructions to complete in fewer clock cycles. For example, no instruction in the original design could complete in less than 2 cycles. In contrast, about 10% of the core's instruction set require only a single cycle (all accumulator addressing mode instructions and many implied mode instructions). Many multi-cycle instructions also require fewer cycles. A complete list of opcode timings is available as part of the documentation package.

Two interrupt lines are provided IRQ (maskable) and NMI (non-maskable). Interrupts are serviced after the currently executing instruction is complete. Since the longest instruction is 5 cycles long, and the IRQ takes another 5 cycles to save the current PC and flags and to load the PC with the interrupt vector, the maximum interrupt latency is 10 clock cycles.

A ready line is provided for interfacing with slow memory or peripherals. It also allows the user to single-step the core. RDY is sampled on the falling edge of the PHI2 input clock. If RDY is sampled low, the processor will hold its current state until RDY is sampled high.

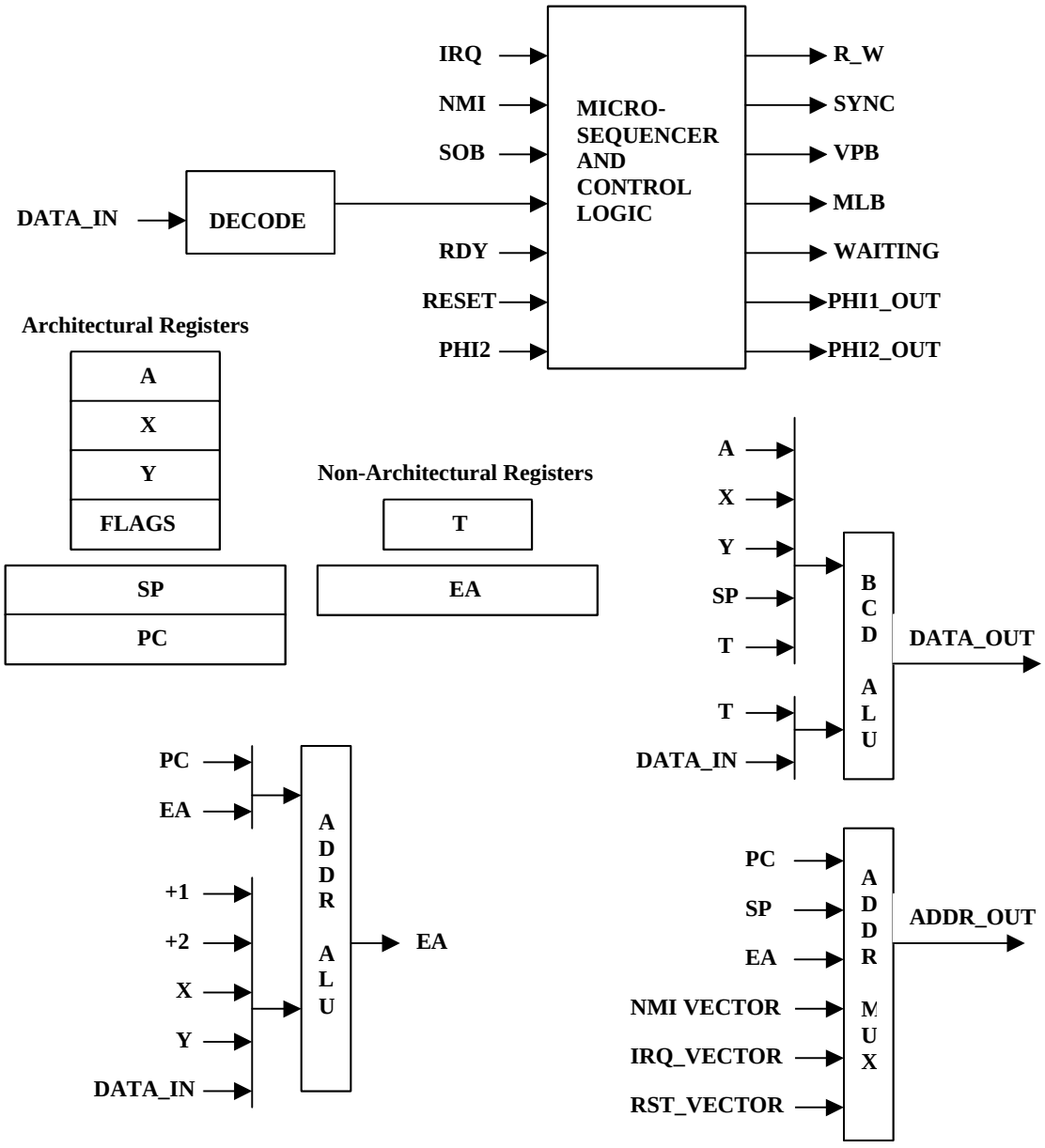
Only a single clock (PHI2) is required. All state transitions occur on the falling edge of this clock, so precise clock symmetry is not required. Your exact system clock rate will depend on your synthesis tools and memory access time, but in a modern CMOS process technology, system clock rates of 50 to 100 MHz should be easily achievable.

The core has been tested for 65C02 compatibility by using both a simulation test suite and an FPGA validation board. The validation board adapts an Altera 10K30 FPGA into the 65C02 40-pin-DIP footprint. This board was then plugged into an Apple IIe. Several versions of Apple DOS were booted and many application programs were run successfully.

Features

- 65C02 instruction-set compatible
  - 70 Instructions, 212 opcodes
  - 15 addressing modes
  - Decimal arithmetic instructions
  - Branch-on-bit-set/reset instructions
  - Semaphore test-and-set/reset-bit instructions
  - Fewer clock cycles per instruction than the original 6502 design
  - 64K byte address space
- Small and fast technology-independent design
- Clean-room synthesizable VHDL design
- Fully static synchronous design
- No internal three-state busses
- Easy to modify and add custom instructions
- Easy to use test bench

Block Diagram



VHDL Design Partitioning

Synthesizable Design Files

65ctop.vhd

This is the top-level design file. It contains only the tri-state drivers for the data bus and very little else. This is also where other pin-driver logic would reside and will probably need to be customized, depending on the user's silicon foundry.

65c02.vhd

This is where most of the 65C02 instruction set architecture (ISA) logic resides. It contains a combination of combinatorial and sequential logic.

decode.vhd

This is the instruction decoder. It is purely combinatorial logic that takes the 8-bit opcode in and generates control lines for one of 70 instructions and 15 addressing modes.

addr.vhd

This is the 16-bit address-computation logic. It calculates the addresses of instruction operands, as well as the next program counter address. This is a purely combinatorial-logic block.

alu.vhd

This is the 8-bit BCD-capable ALU. This is primarily a combinatorial block with the exception of a couple of state flip-flips that keep track of decimal adjust operations.

mytypes.vhd

This is where all of the design-specific type definitions reside. These include the macro-opcodes and the micro-opcodes (e.g. ALU micro-ops).

Non-synthesizable Testbench Files

65csim.vhd

This is the top-level simulation file. It instantiates 65c02.vhd and mem.vhd.

mem.vhd

This is a dual-ported memory model that reads a 65c02-assembler Intel-hex formatted file called mem.dat at simulation startup time.

Running the testbench

Running the testbench is very simple. First assemble your 65c02-source code using any assembler which can generate Intel hex files (e.g. as65.exe which is available free from the Simtel archives). Then copy the generated Intel hex or rename it to mem.dat. This file must reside in the same directory as your simulation model.

Synthesis Scripts

Here's an example of a simple synthesis script for Synopsis.

```
/* ============================================ */
/* Synopsys to LSI G10 script example           */
/* ============================================ */

sh date

/* ============================================ */
/* Read the library database                    */
/* ============================================ */

read -f db /tools/lsi/lsitk_3.0/lib/synopsys/lcbg10p/lcbg10p_wccomv.db

/* ============================================ */
/* Analyze the design                           */
/* ============================================ */

analyze -f vhdl mytypes.vhd
analyze -f vhdl alu.vhd
analyze -f vhdl addr.vhd
analyze -f vhdl decode.vhd
analyze -f vhdl 65c02.vhd
read    -f vhdl 65ctop.vhd

current_design IP_65C02
compile -map_effort high -boundary_optimization

/* ============================================ */
/* generate the report files                    */
/* ============================================ */

report_area >  lsiarea.log
report_constraints -all_violators
report_timing -max_paths 10 > lsitime.log
check_design > lsichek.log

/* ============================================ */
/* Write the database                           */
/* ============================================ */

write -f db -hier -o db/IP_65C02.db IP_65C02

sh date
quit
```


Appendix A (Interface Signals)

65ctop.vhd and 65c02.vhd Interface Signals

```
ADDR_OUT   : out   std_logic_vector(15 downto 0);

DATA_IN    : in    std_logic_vector( 7 downto 0); --65c02.vhd  only
DATA_OUT   : out   std_logic_vector( 7 downto 0); --65c02.vhd  only

DATA       : inout std_logic_vector( 7 downto 0); --65ctop.vhd only


R_W        : out std_logic;   -- Read/Write status
SYNC       : out std_logic;   -- Opcode fetch status
VPB        : out std_logic;   -- Vector pull status (active low)
MLB        : out std_logic;   -- Memory Lock status (active low)
WAITING    : out std_logic;   -- Waiting for Interrupt status
RDY        : in  std_logic;   -- Ready input
SOB        : in  std_logic;   -- Set Overflow Bit (active-low)
RESET      : in  std_logic;   -- Reset input (active-low)
IRQ        : in  std_logic;   -- Interrupt Request (active-low)
NMI        : in  std_logic;   -- Non-maskable IRQ  (active-low)
PHI1_OUT   : out std_logic;   -- opposite phase as PHI2
PHI2_OUT   : out std_logic;   -- same phase as PHI2
PHI2       : in  std_logic;   -- System Clock
```

ADDR_OUT is the 16-bit address bus.

DATA_IN and **DATA_OUT** are kept separate to allow for the use of Dual-ported RAM cells, but are easily combined into a single tri-state bus.

DATA is the combined tri-state data bus.

R_W is the Read/Write status pin (1=read, 0=write)

SYNC is active (high) during opcode-fetch cycles (1=fetch)

VPB is active (low) during the 2 clock cycles of an interrupt sequence during which the interrupt vector is being loaded.

MLB is active (low) during read-modify-write cycles.

WAITING goes active (high) after the processor executes a WAI (Wait for Interrupt) instruction; it stays active until an interrupt occurs.

RDY is sampled at the falling edge of the PHI2 clock. If it is low, the processor will wait in an idle state until RDY goes high.

SOB is the active low set-overflow-bit input. It is sampled on the rising edge of PHI2. If SOB active the overflow flag bit will be set.

RESET is the active low system reset input. RESET is internally synchronized to PHI2 and debounced to reject glitches less than one system clock wide. The IP_65C02 also waits for 511 clocks after RESET goes inactive before starting to fetch instructions. This is done to wait for other system components to complete their initialization sequences. If this wait is not needed, it can be easily removed.

IRQ is the active low maskable interrupt request input. IRQ is sampled at the falling edge of PHI2 during an opcode fetch cycle. It is ignored if the interrupt disable flag is set.

NMI is the active low non-maskable interrupt request input. NMI is sampled at the falling edge of PHI2 during an opcode fetch cycle.

PHI1_OUT is the inverted and buffered output clock.

PHI2_OUT is the non-inverted buffered output clock.

PHI2 is the system clock input. All internal timing is referenced to PHI2.