# Project Goals:

100% software compatible with the 65816. Bus and bus cycle similar but not exact. Easy development and modification.

# Development Language

The core has been developed in the Verilog language exclusively.

# Core Optimization

It should be noted that this core is not optimized for a small footprint. The core is fairly large being around 5000 6-LUTs. ( about 8,000 Logic cells).

# Top Module

An MPU module encapsulates the ‘816 core and provides additional functionality for the system, including system chip selects and counters.

|  |  |  |
| --- | --- | --- |
| FT816MPU.v |  |  |
|  | FT816.v – 65816 compatible core |  |
|  |  |  |

# Clocks

The core generates low speed clocks for interfacing to peripherals. Generated clocks have two non-overlapping phases. The clock rates are /4 and 1/32 of the core’s clock rate.

# Core Parameters

pIOAddress – This parameter determines the address range at which control registers are located. The default is $F000.

pZPAddress – This parameter determines where in memory the count value registers are located. The default is $0010.

# Module Ports

|  |  |  |  |
| --- | --- | --- | --- |
|  | In/Out |  | Active |
| CLK | I | Input clock 32 MHz |  |
| PHI11 | O | 1 MHz Output phase 1 (non-overlapping) |  |
| PHI12 | O | 1 MHz Output phase 2 (non-overlapping) |  |
| PHI81 | O | 8 MHz Output phase 1 (non-overlapping) |  |
| PHI82 | O | 8 MHz Output phase 2 (non-overlapping) |  |
| PHI31 | O | 32 MHz |  |
| RST | I | reset | Low |
| IRQ | I |  | Low |
| NMI | I |  | Low |
| ABORTB | I | abort | Low |
| E | O | emulation status |  |
| BE | I | bus enable (tri-state the output signals) |  |
| MLB | O | memory lock |  |
| MX | O | mode select (M/X bits of status reg) |  |
| CS0 | O | Chip select | Low |
| CS1 | O | Chip select | Low |
| CS2 | O | Chip select | Low |
| CS3 | O | Chip select | Low |
| CS4 | O | Chip select | Low |
| CS5 | O | Chip select | Low |
| CS6 | O | Chip Select Other | Low |
| VDA | O | valid data address |  |
| VPB | O | vector pull |  |
| VPA | O | valid program address |  |
| RW | O | read/write |  |
| RDY | I | bus ready |  |
| A0-A23 | O | Address bus |  |
| D0-D7 | I/O | data bus |  |
| CT0 | I | Counter pulse |  |
| CT1 | I | counter pulse |  |
| CT2 | I | Counter pulse |  |

# Programmable Chip Select Array

The programmable chip select array is located at $00F000. These registers are write-only. Once setup further programming of the chip select array can be disabled by writing ‘E0’ to register 15. Care should be taken to ensure that decoded address ranges don’t overlap, otherwise bus contention may result. A chip select can be disabled by not specifying a speed select for the chip select.

On reset the chip selects default to respond to the following addresses:

CS0 = $0070xx ; 256 bytes – 1MHz

CS1 = $0071xx ; 256 bytes – 1MHz

CS2 = $0072xx ; 256 bytes – 1MHz

CS3 = $0073xx ; 256 bytes – 1MHz

CS4 = $008xxx ; 32k bytes – 8MHz

CS5 = $01xxxx ; 32k bytes – 8MHz

CS6 = not (CS0 or CS1 or CS2 or CS3 or CS4 or CS5) ; full speed

Chip selects are active low.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reg |  | Bits Covered | Reset Value |  |
| F000 | CS0 | 15 to 8 | D0 |  |
| F001 | CS0 | 23 to 16 | 00 |  |
| F002 | CS1 | 15 to 8 | D1 |  |
| F003 | CS1 | 23 to 16 | 00 |  |
| F004 | CS2 | 15 to 8 | D2 |  |
| F005 | CS2 | 23 to 16 | 00 |  |
| F006 | CS3 | 15 to 8 | D3 |  |
| F007 | CS3 | 23 to 16 | 00 |  |
| F008 | CS4 | 22 to 15 | 01 |  |
| F009 | CS4 | 23 | 00 |  |
| F00A | CS5 | 22 to 15 | 02 |  |
| F00B | CS5 | 23 | 00 |  |
| F00C | SS1 |  | 0F | speed select 1MHz (1 = 1MHz) |
| F00D | SS8 |  | 30 | speed select 8MHz (1 = 8MHz) |
| F00E | SS32 |  | 00 | full speed select |
| F00F | EN |  |  | Write E0 to this register to disable the register set |

Bit 0 of the speed select register corresponds to CS0

Bit 1 of the speed select register corresponds to CS1

etc.

Only one bit should be set in a speed select for a chip select.

# Programmable Counters

There are three24 bit programmable counters present in the FT816MPU component. All three counters operate in an identical fashion. The counters may be programmed to count up or down using one of three clock sources. The first clock source is via software trigger by writing to the counter trigger register. The second source is automatic counting with automatic reload by the internal mpu clock. The third clock source is from an external count pulse. Counters reload automatically once the count expires. The count expires when the count reaches the limit while counting upwards, or when the count reaches zero while counting downwards.

The count value registers occupy zero page memory from $10 to $1F. Placing the value registers in zero page allows the counters to be used with zero page indirect addressing. The counter control registers occupy memory between $F010 to $F01F.

Counting may be disabled which allows the count value registers to act like memory locations.

|  |  |  |  |
| --- | --- | --- | --- |
| Reg |  |  | |
| F010 | count base/limit low byte | When counting up the counter may generate an interrupt when the limit value Is reached. The counter will then automatically reset to zero.  When counting down, the counter may generate an interrupt when it reaches zero. Then the counter is reloaded with a base count from this register. | |
| F011 | count base limit middle byte |
| F012 | count base / limit high byte |
| F013 | Bit |  |  |
| 0 | IRQ enable | 1 = irq enabled |
| 3:2 | count source | 00 = by software trigger only  01 = automatic on internal clock  10 = count external pulses |
| 4 | count direction | 1 = count up (0 to limit)  0 = count down (base to 0) |
|  | 5 to 7 | reserved | |
| F014-F017 | Identical to F010 to F013 except for counter #2 | | |
| F018-F01B | Identical to F010 to F013 except for counter #3 | | |
| F01C | Writing to this register triggers a count cycle for counter #1 | | |
| F01D | Writing to this register triggers a count cycle for counter #2 | | |
| F01E | Writing to this register triggers a count cycle for counter #3 | | |

|  |  |  |
| --- | --- | --- |
| $0010 to $0012 | low, mid, high bytes of counter value #1 |  |
| $0014 to $0016 | low, mid, high bytes of counter value #2 |  |
| $0018 to $001A | low, mid, high bytes of counter value #3 |  |

The counter values may be modified at any time by updating the value registers. However, the counter should be stopped prior to updating the registers in order to avoid a partial update of the counter. The counter may be stopped by setting the count source to software trigger only.

# Additional Core Features

## Store Bypassing

The FT816 core bypasses the second store cycle during a read-modify-write operation if the upper bits of the value have not changed.

## Long Branches

The core also features long branches. If the branch displacement byte is $FF then the next two bytes are used as a 16 bit branch displacement.

Opcode Map – 8 bit mode W65C816 compatible

|  |  |
| --- | --- |
|  | = W65C816S instrucitons |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | -0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -A | -B | -C | -D | -E | -F |
| 0- | BRK | ORA (d,x) | COP | ORA d,s | TSB d,r | ORA d | ASL d | ORA [d] | PHP | OR #i8 | ASL acc | PHD | TSB abs | ORA abs | ASL abs | ORA AL |
| 1- | BPL disp | ORA (d),y | ORA (d) | ORA (d,s),y | TRB d,r | OR d,x | ASL d,x | ORA [d],y | CLC | OR abs,y | INA | TAS | TRB abs | ORA abs,x | ASL abs,x | ORA AL,x |
| 2- | JSR abs | AND (d,x) | JSL abs24 | AND d,s | BIT d | AND d | ROL d | AND [d] | PLP | AND #i8 | ROL acc | PLD | BIT abs | AND abs | ROL abs | AND AL |
| 3- | BMI disp | AND (d),y | AND (d) | AND (d,s),y | BIT d,x | AND d,x | ROL d,x | AND [d],y | SEC | AND abs,y | DEA | TSA | BIT abs,x | AND abs,x | ROL abs,x | AND AL,x |
| 4- | RTI | EOR (d,x) | WDM | EOR d,s | MVP | EOR d | LSR d | EOR [d] | PHA | EOR #i8 | LSR acc | PHK | JMP abs | EOR abs | LSR abs | EOR AL |
| 5- | BVC disp | EOR (d),y | EOR (d) | EOR (d,s),y | MVN | EOR d,x | LSR d,x | EOR [d],y | CLI | EOR abs,y | PHY | TCD | JML abs24 | EOR abs,x | LSR abs,x | EOR AL,x |
| 6- | RTS | ADC (d,x) | PER | ADC d,s | STZ d | ADC d | ROR d | ADC [d] | PLA | ADC #i8 | ROR acc | RTL | JMP (abs) | ADC abs | ROR abs | ADC AL |
| 7- | BVS disp | ADC (d),y | ADC (d) | ADC (d,s),y | STZ d,x | ADC d,x | ROR d,x | ADC [d],y | SEI | ADC abs,y | PLY | TDC | JMP (abs,x) | ADC abs,x | ROR abs,x | ADC AL,x |
| 8- | BRA disp | STA (d,x) | BRL disp | STA d,s | STY d | STA d | STX d | STA [d] | DEY | BIT # | TXA | PHB | STY abs | STA abs | STX abs | STA AL |
| 9- | BCC disp | STA (d),y | STA (d) | STA (d,s),y | STY d,x | STA d,x | STX d,y | STA [d],y | TYA | STA abs,y | TXS | TXY | STZ abs | STA abs,x | STZ abs,x | STA AL,x |
| A- | LDY #i8 | LDA (d,x) | LDX #i8 | LDA d,s | LDY d | LDA d | LDX d | LDA [d] | TAY | LDA #i8 | TAX | PLB | LDY abs | LDA abs | LDX abs | LDA AL |
| B- | BCS disp | LDA (d),y | LDA (d) | LDA (d,s),y | LDY d,x | LDA d,x | LDX d,y | LDA [d],y | CLV | LDA abs,y | TSX | TYX | LDY abs,x | LDA abs,x | LDX abs,x | LDA AL,x |
| C- | CPY #i8 | CMP (d,x) | REP # | CMP d,s | CPY d | CMP d | DEC d | CMP [d] | INY | CMP #i8 | DEX | WAI | CPY abs | CMP abs | DEC abs | CMP AL |
| D- | BNE disp | CMP (d),y | CMP (d) | CMP (d,s),y | PEI | CMP d,x | DEC d,r | CMP [d],y | CLD | CMP abs,y | PHX | STP | JML (a) | CMP abs,x | DEC abs,x | CMP AL,x |
| E- | CPX #i8 | SBC(d,x) | SEP # | SBC d,s | CPX d | SUB d | INC d | SBC [d] | INX | SBC #i8 | NOP | XBA | CPX abs | SBC abs | INC abs | SBC AL, |
| F- | BEQ disp | SBC (d),y | SBC(r) | SBC (d,s),y | PEA | SUB d,x | INC d,r | SBC [d],y | SED | SBC abs,y | PLX | NAT | JSR (abs,x) | SBC abs,x | INC abs,x | SBC AL,x |