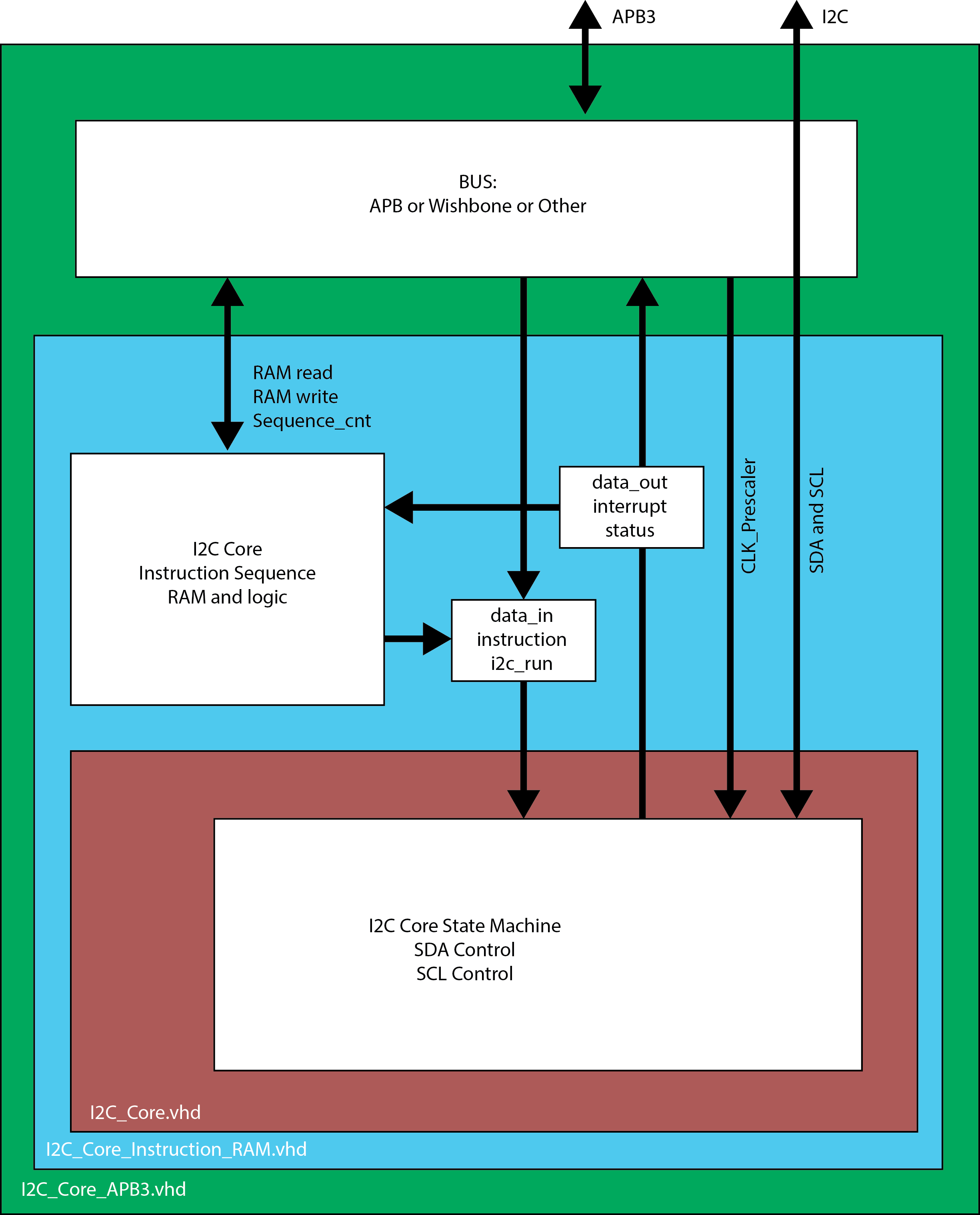
I2C Core Design Doc

# Intent

The intent of this block is to handle all interaction with the I2C SDA and SCL lines. This will take



# Ports

## Instruct

|  |  |
| --- | --- |
| Bit | Function |
| 2 - 0 | 000 : NOP, No change to SDA or SCL but Interrupt will fire after 1 I2C clock  001 : Start, if I2C bus is idle, Start event will be sent.  010 : Stop, if status shows this I2C is in control, Stop signal will be sent.  011 : Repeated Start, if status shows this I2C is in control, RStart signal will be sent.  100 : Data will be written to the I2C bus, SDA Enable is enabled.  101 : Data will be read from the I2C bus, SDA Enable is disabled. |

## Status\_out

|  |  |
| --- | --- |
| Bit | Function |
| 1 - 0 | 00 : Idle  01 : Busy doing action according to instruction bits  10 : Interrupt active, waiting for next command, holding I2C bus  11 : Interrupt active with ACK failed, waiting for next command, holding I2C bus |

# I2C Core – Base

## Ports

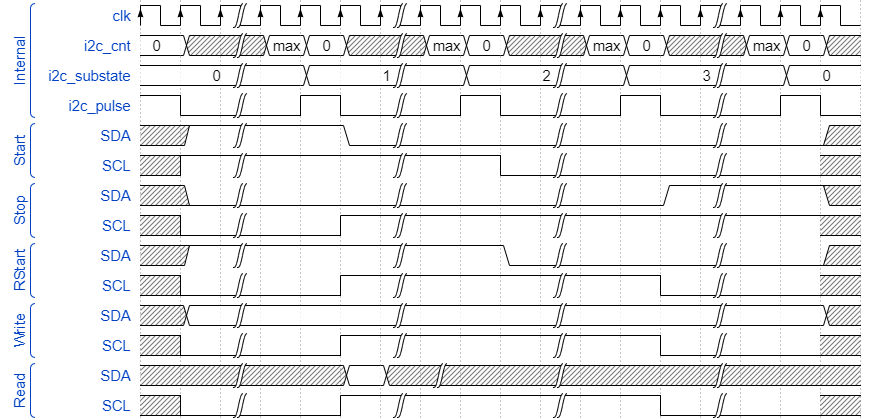
|  |  |  |  |
| --- | --- | --- | --- |
| Port | Size | Dir | Function |
| PCLK | 1 | In | Peripheral clock, full Freq |
| RSTn | 1 | In | Reset = 0, asynchronous |
| Initiate | 1 | In | Rising edge (0 to 1) triggers an I2C instruction |
| Instruct | 8 | In | Instruction signals to determine I2C operation (START, STOP, etc.) |
| Clk\_div\_in | 16 | In | PCLK prescaler to generate I2C SCL clock |
| i2c\_bus\_busy | 1 | Out | Indicates whether the I2C bus is busy from any master. |
| I2c\_int | 1 | Out | Indicates when the I2C Core is waiting for a command |
| status\_out | 2 | Out | Status output, links to an internal register that represents the current status of the I2C\_Core. |
| data\_in | 8 | In | Data input to be transmitted, must be stable for the duration of the write operation indicated by status\_out = “01” |
| data\_out | 8 | Out | Data output, when Receive byte is complete. Can be read at any time but may be gibberish if status\_out = “01” |
| SDAI | 1 | In | SDA line input, Y from BiBuf |
| SDAO | 1 | Out | SDA line output, D to BiBuf |
| SDAE | 1 | Out | SDA output enable, E to BiBuf |
| SCLI | 1 | In | SCL line input, Y from BiBuf |
| SCLO | 1 | Out | SCL line output, D to BiBuf |
| SCLE | 1 | Out | SCL output enable, E to BiBuf |

## Internal Register

### Manual Control Registers

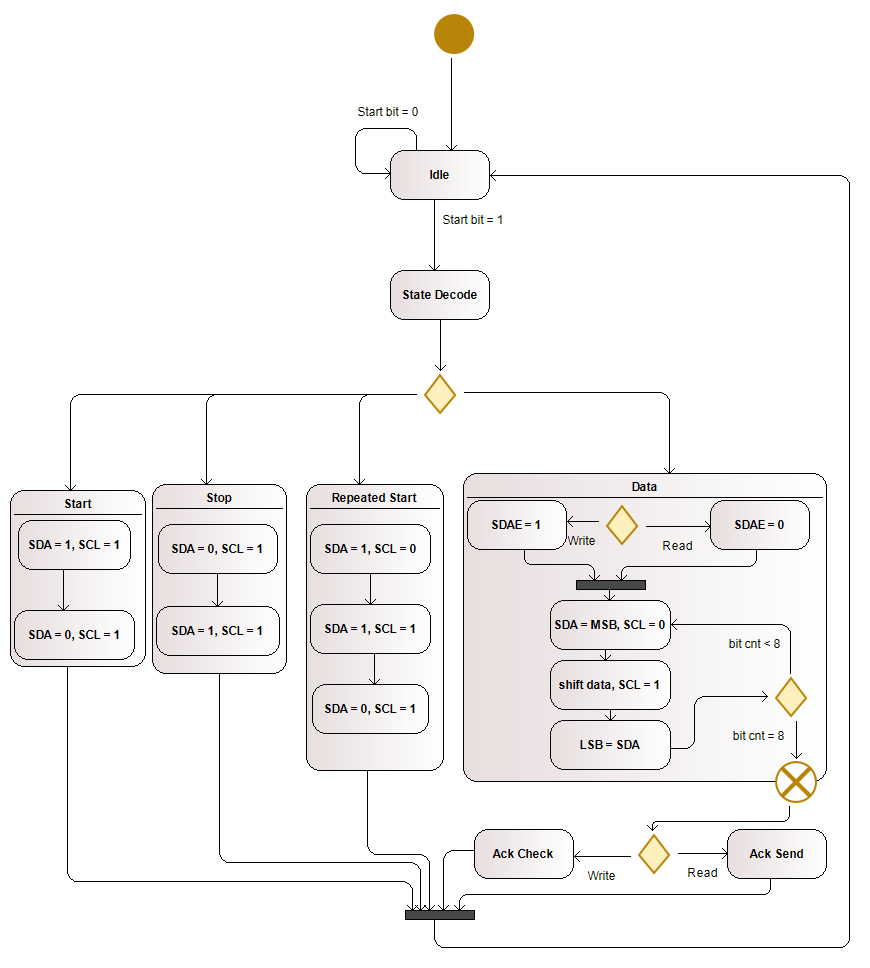
Shift register stores 8 bits and shifts the MSB onto the I2C bus every I2C clock cycle or shifts the bit on the I2C bus into the LSB every I2C clock cycle.

## Clocking



clk\_i2c\_write and clk\_i2c\_read will be combined into a single signal where the falling edge will trigger the read logic. This is to reduce the signal count and I guess to maybe reduce readability.

## I2C Event State Machine



Transmit and Receive can write to SDA from the MSB before clock high and read from SDA to the LSB during clock high. These can happen during every transaction. The result is that during transmit, the reading circuitry will record the data just exported back into the data register, basically a circular register.

# Instruction RAM

## Ports

Table : Instruction RAM Ports

|  |  |  |  |
| --- | --- | --- | --- |
| Port | Size | Dir | Function |
| Adr\_to\_mem | 6 | In | While seq\_enable = 0 or seq\_finished = 1, this address determines the active RAM location. |
| Mem\_instr\_sel | 1 | In | 0 : Data portion of RAM location is selected  1 : Instruction portion of RAM location is selected |
| Bus\_w\_en | 1 | In | This flag determines the meaning of mem\_done.  0 : Bus interaction intends to read from RAM  1 : Bus interaction intends to write to RAM |
| Mem\_op\_req | 1 | In | This signals to RAM that bus signals are stable and RAM process should begin |
| Mem\_done | 1 | Out | This signal indicates that the RAM operation indicated at the time mem\_op\_req is set to 1 is complete. This flag is reset upon mem\_op\_req is set to 0  If bus\_w\_en = 0  0 : mem\_to\_bus is not stable  1 : mem\_to\_bus is stable and represents the value requested by adr\_to\_mem and mem\_instr\_sel  If bus\_w\_en = 1  0 : bus\_to\_mem has not been written to RAM location  1 : bus\_to\_mem has been written to RAM location defined by adr\_to\_mem and mem\_instr\_sel |
| bus\_to\_mem | 8 | In | Data or instruction to be written to RAM location defined by adr\_to\_mem and mem\_instr\_sel |
| Mem\_to\_bus | 8 | Out | Data or instruction read from RAM location defined by adr\_to\_mem and mem\_instr\_sel |
| Seq\_enable | 1 | in | This signals to the sequence logic to begin iterating through RAM locations and performing the instructions stored there |
| Seq\_finished | 1 | Out | This signal indicates when the sequence operation begun by seq\_enable has finished. This flag is reset upon seq\_enable being set to 0. |
| Seq\_cnt | 6 | Out | This indicates which RAM location is being operated on while the sequence logic is running. |
| I2c\_initiate | 1 | in | Passthrough, overridden while seq\_enable = 1 and seq\_finished = 0 |
| I2c\_instruct | 3 | In | Passthrough, overridden while seq\_enable = 1 and seq\_finished = 0 |
| I2c\_clk\_div\_in | 16 | in | passthrough |
| I2c\_bus\_busy | 1 | out | passthrough |
| I2c\_int | 1 | out | Passthrough, |
| I2c\_status\_out | 2 | out | passthrough |
| I2c\_data\_in | 8 | in | Passthrough, overridden while seq\_enable = 1 and seq\_finished = 0 |
| I2c\_data\_out | 8 | out | Passthrough |
| SDAI | 1 | In | Passthrough |
| SDAO | 1 | Out | Passthrough |
| SDAE | 1 | Out | Passthrough |
| SCLI | 1 | In | Passthrough |
| SCLO | 1 | Out | Passthrough |
| SCLE | 1 | Out | passthrough |

## uSRAM

Within Smartfusion2 devices each uSRAM block contains 1,152 bits. This is configurable to a variety of depth and width combinations but this design will use 128 locations of 8 bits each. These locations will be used in groups of 2 where even RAM locations (LSB = 0) will store data and odd RAM locations (LSB = 1) will store the instruction.

Table : RAM Address Description

|  |  |
| --- | --- |
| Bits | Address |
| 6 – 1 | RAM location as set by adr\_to\_mem or seq\_count |
| 0 | 0 : Data  1 : Instruction |

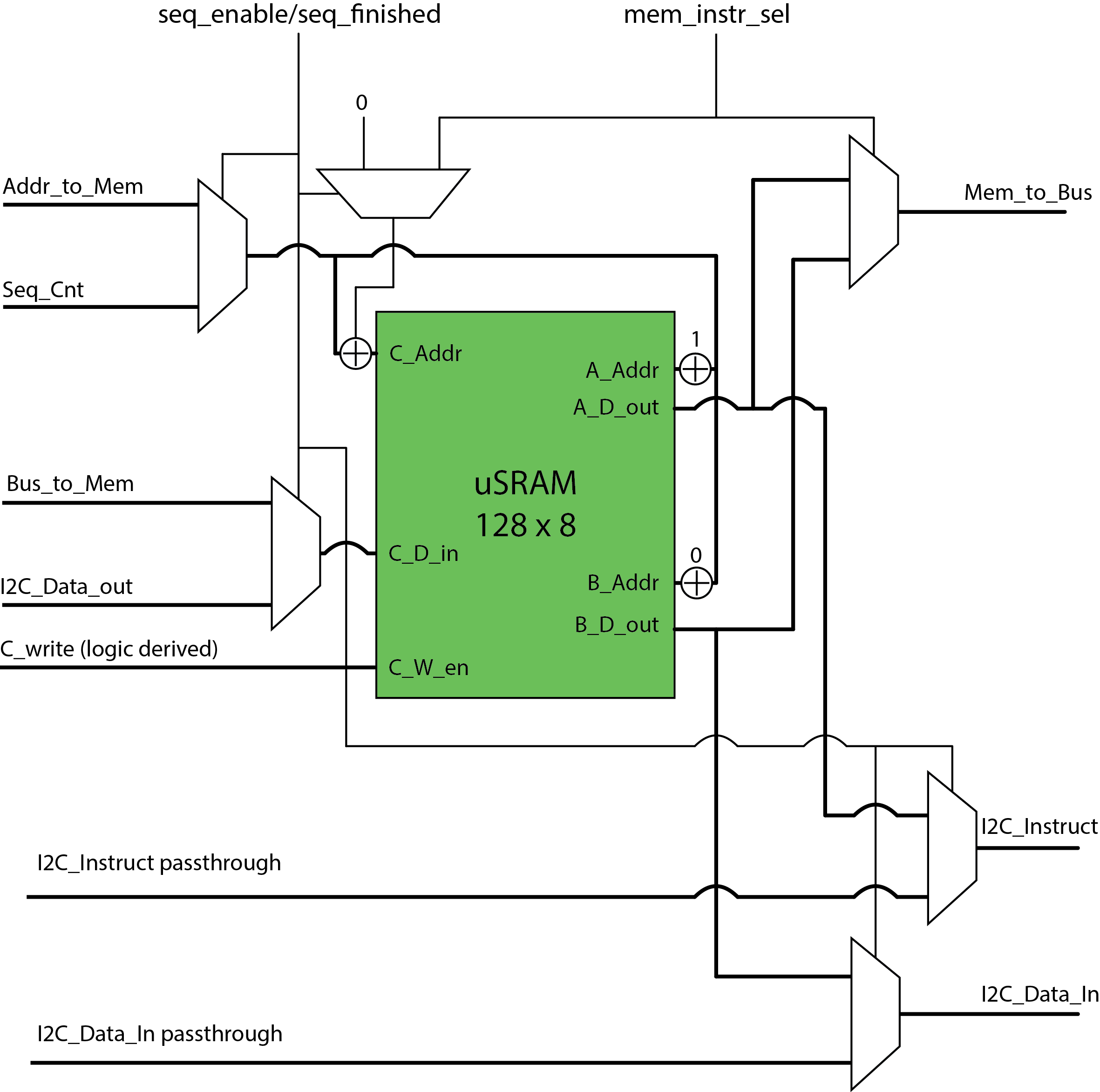
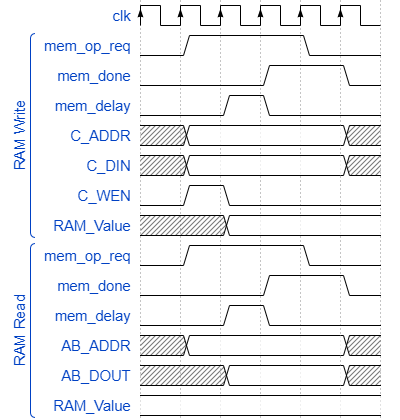


Figure : uSRAM wiring Diagram



Conditions for writing to the C port:

If instruction RAM is in “sequence mode”

# APB Interface

## Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Port | Addr | R/W | Size | Function |
| ctrl | 0x00 | W | 8 | Control I2C |
| status | 0x01 | R | 8 | Status of I2C core, including sequential instruction status |
| Clk\_0 | 0x02 | W | 8 | Lower half of I2C clock prescaler. Bits [7-0] |
| Clk\_1 | 0x03 | W | 8 | Upper half of I2C clock prescaler. Bits [15-8] |
| Data\_in | 0x04 | W | 8 | Byte to be written to the I2C bus. Never overwritten by the I2C Core |
| Data\_out | 0x05 | R | 8 | Last byte read from the I2C bus. Updated each read and write transaction |
| Instr\_Seq\_data | 0x8X | R/W | 8 | Contains data or I2C instruction |
| Instr\_Seq\_cmd | 0xCX | W | 2 | Instructs the instruction sequence on how to interpret the data bits. |

### Control Register

|  |  |
| --- | --- |
| Bit | Function |
| 7 – 5 | Unused |
| 4 | Initiate instruction sequence by setting to 1  Bit must be cleared to clear interrupt if triggered by sequence. |
| 3 - 1 | I2C instruction, ignored during instruction sequence  000 : NOP, No change to SDA or SCL but Interrupt will fire after 1 I2C clock  001 : Start, if I2C bus is idle, Start event will be sent.  010 : Stop, if status shows this I2C is in control, Stop signal will be sent.  011 : Repeated Start, if status shows this I2C is in control, RStart signal will be sent.  100 : Data will be written to the I2C bus, SDA Enable is enabled.  101 : Data will be read from the I2C bus, SDA Enable is disabled. |
| 0 | I2C initiate instruction, ignored during instruction sequence  Rising edge (0 to 1) triggers the instruction to start. |

### Status Register

|  |  |
| --- | --- |
| Bit | Function |
| 7 - 2 | Status of Instruction Sequence  Indicates the instruction running if bits [1-0] indicate busy  6 bits allow up to 64 instructions to be neatly tracked, if sequence is longer, value may have overflowed |
| 1 - 0 | I2C Core status  00 : Idle  01 : Busy doing action according to instruction bits  10 : Interrupt active, waiting for next command, holding I2C bus  11 : Interrupt active with ACK failed, waiting for next command, holding I2C bus |

### Clock Prescaler Registers

|  |  |
| --- | --- |
| Bit | Function |
| 15 - 8 | Address:  0x02 |
| 7 - 0 | Address:  0x03 |

### Data In Register

### Data Out Register

## Instruction Sequence

### Registers

Additional registers are available to provide offloaded I2C instructions.

#### Address

0x80 start. Configurable.

|  |  |
| --- | --- |
| Bit | Function |
| 7 | Selects Automated Sequence registers |
| 6 | 0 : register data, bits 7 - 0  1 : register command, bits 9 - 8 |
| 5 - 0 | Automated Sequence Address |

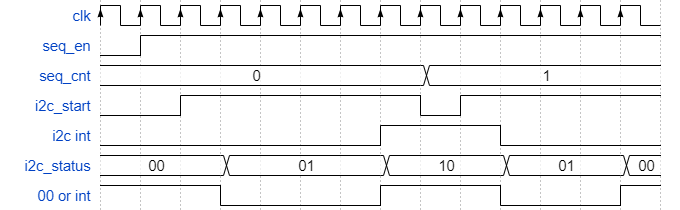
#### Data

Each Universal Register may contain an I2C slave address, an I2C slave register address, or I2C slave register data.

|  |  |  |
| --- | --- | --- |
| Bit | R/W | Function |
| 9 - 8 | W | 00 : No Operation, used to identify unused registers.  01 : bits 7-0 identify special conditions such as START, STOP, etc.  10 : data in bits 7-0 are to be written to the I2C bus  11 : data in bits 7-0 are to be read from the I2C bus |
| 7 - 0 | W | Bits to be read or written to the I2C bus where 7 is the MSB  If bit 9 = 1:  0b00000001 = START  0b00000010 = STOP  0b00000011 = REPEATED START |

e.g. The LiteOn LTR-329ALS-01 Optical Sensor has 4 data registers that must be read in a certain order to receive the complete data for its 2 optical channels. This sequence consists of 12 bytes that must be read and written to the I2C bus and is likely performed in the same order many times while the system is running. Automated Polling Registers allow this sequence to be stored within the I2C Core and offloaded from the system controller. An interrupt will be triggered to indicate when the sequence is complete and the data may be read or updated.

### Situations



#### Read to Sequence from I2C

Must occur after the sequence instruction has been completed by the I2C Core

Must occur while the I2C data\_out is stable