# EE272
# F’24 Project
# V1.0

This semesters project is to build a calculation engine for AI. The engine is connected to a wide high speed bus to provide sufficient memory bandwidth. This bus is described later in the document. The engine uses FP11 (from HW1 and HW2). An engine module contains 4 of the FPM blocks connected to one FPA block. This is referred to as SUM4 in the document. This provides an addition of 4 multiplies. 4 SUM4 blocks are connected to an FPA block. This is referred to as SUM16 in the document. This will need to be pipelined to synthesize at 1ns cycle time.

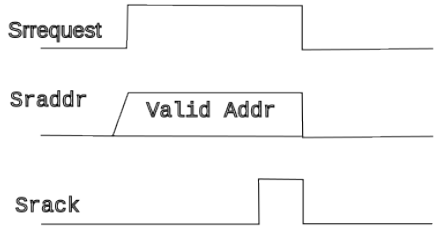
The SUM16 block has 16 11 bit A and B inputs, and a single 11 bit output. A single operation requires 16*11 (176) bits of A data, and 16*11 (176) bits of B data. The bus has a read mechanism, and a separate write mechanism. The read mechanism is a 352 bit bus. The upper 176 bits contain 16 A values. The lower 176 bits contain 16 B values.

The read bus works in burst reads of 16 X 352 bit transfers. This will create 16 outputs. These are assembled into 176 bits, and transferred on the output bus. The output bus has 176 bits of data. It is single transfers.

The busses are named in honor of SJSU, so all the signal names begin with an ‘S’. All S bus addresses reference a 352 or 176 bit word. It is not a byte address.

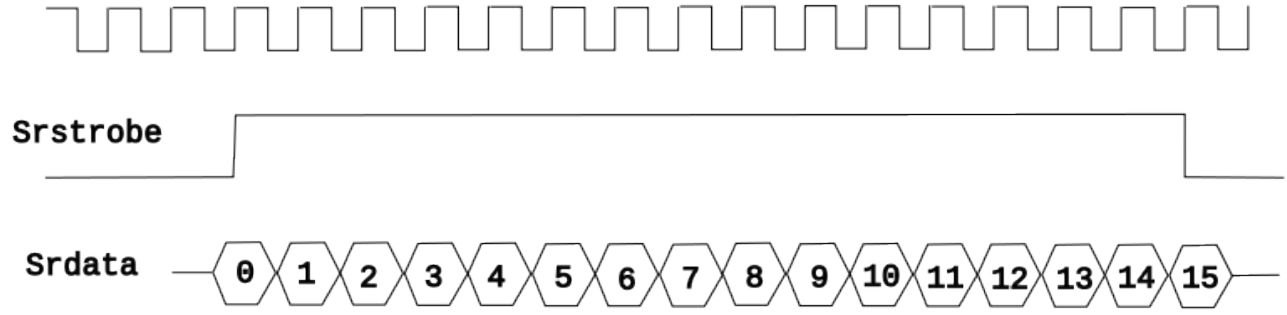
| Name | Dir | Bits | Comment |
|---|---|---|---|
| Sclock | In | 1 | Active high clock for the bus |
| Sreset | In | 1 | Active high reset for the bus hardware |
| Srrequest | Out | 1 | Bus Request |
| Srack | In | 1 | Bus acknowledge. Request accepted |
| Sraddr | Out | 48 | Read address |
| Srstrobe | In | 1 | Read data strobe |
| Srdata | In | 352 | Read data |
| Swrequest | Out | 1 | Write request |
| Swack | In | 1 | Write request accepted |
| Swaddr | Out | 48 | Write address |
| Swdata | Out | 176 | Write data |

A read operation starts with a Srrequest asserting while presenting the address. This is held on the read bus until Srack is asserted indicating the request has been accepted.



Another request for this device may not be made until the first data is returned in the 16 cycle burst. The system bus fabric (Which is part of the project design) will perform arbitration based on the Srrequest signals from each requester. (There will up to 4 of these in the system). The memory controller interface to the fabric is described later.

The bus waits for the Srstrobe signal indicating the first data is on the bus. Note the bus has no way to slow down data transfers. The Srstrobe signal is active for 16 cycles.




The write bus is like the read bus. There is no response. When a write is acknowledged, It will occur in a few cycles with no error. The address and data are latched, and do not need to be retained after the swack.

A simple register interface is provided. This interface is inspired by APB. However, there is no ready signal. All transfers are 2 cycles. The first cycle is used to decode the device, and the second cycle is used to transfer the data.

All signals on the register interface begin with the letter R for register interface.

| Name | Dir | Bits | Comment |
|---|---|---|---|
| Rclk | In | 1 | Read bus clock |
| Rreset | In | 1 | Read bus reset |
| Rwrite | In | 1 | Indicates the bus cycle type. 0=read, 1=write |
| Rxfr | In | 1 | Indicates the second cycle where the data transfer will occur |
| Raddr | In | 64 | Selects device registers. Only the lower 12 bits go to the device |
| Rwdata | In | 64 | The register write data |
| Rrdata | Out | 64 | The register read data |
| $Rdevsel_X$ | In | 1 | Indicates this device is selected. Decode of the upper 52 bits of Raddr according to the following table. |

All signals except the Rrdata are present in both cycles. The Rrdata is only present in the second cycle. The bus fabric and control is part of the test bench. The Raddr is in units of bytes for compatibility with existing software, even though the bus is in 64 bit chucks.

Each device has a unique address space. $Rdevsel_X$ is decodes as follows:

| Addr | Device ($_X$) |
|---|---|
| 0x5e00_0000_0000_0XXX | 0 |
| 0x5e00_0000_0000_1XXX | 1 |
| 0x5e00_0000_0000_2XXX | 2 |
| 0x5e00_0000_0000_3XXX | 3 |

The following Registers exist in each device:

| **Address (last 10 bits)** | **Name** | **Function** |
|---|---|---|
| 0x000 | Econtrol | Controls the engine (See below) |
| 0x008 | Efetchaddr | The starting location to fetch data from |
| 0x010 | Efetchlen | Number of 352 bit data items to process in total |
| 0x018 | Estoreaddr | Location to store the 176 bit results |

Note: the data will be fetched in 352 bit pieces. Any unneeded data will be set to zero by the software/test bench.

The Efetchlen may not be a multiple of 16. Make sure to store the last results which may only partially fill a 176 bit result word. The first result is stored in the low bits of the store location. Ensure all non stored bits are zero.

| Econtrol | | |
|---|---|---|
| Bit location | Name | Function |
| 63:4 | Reserved | Read zeros, ignore writes |
| 3:1 | Fetch Priority | A priority amount for request arbitration |
| 0 | Start | The engine can run. Clears when the engine finishes |

| Efetchaddr | | |
|---|---|---|
| Bit location | Name | Function |
| 63:48 | Reserved | Read zeros, ignore writes |
| 47:0 | Fetch starting address | The starting address to fetch data for operands. This register increments while the operation is in progress. |

| Efetchlen | | |
|---|---|---|
| Bit location | Name | Function |
| 63:16 | Reserved | Read zeros, ignore writes |
| 15:0 | Numwords | The number of 352 bit words to process |

| Estoreaddr | | |
|---|---|---|
| Bit location | Name | Function |
| 63:48 | Reserved | Read zeros, ignore writes |
| 47:0 | Store starting address | The starting address to store 176 bit results in memory. Increments while processing. |