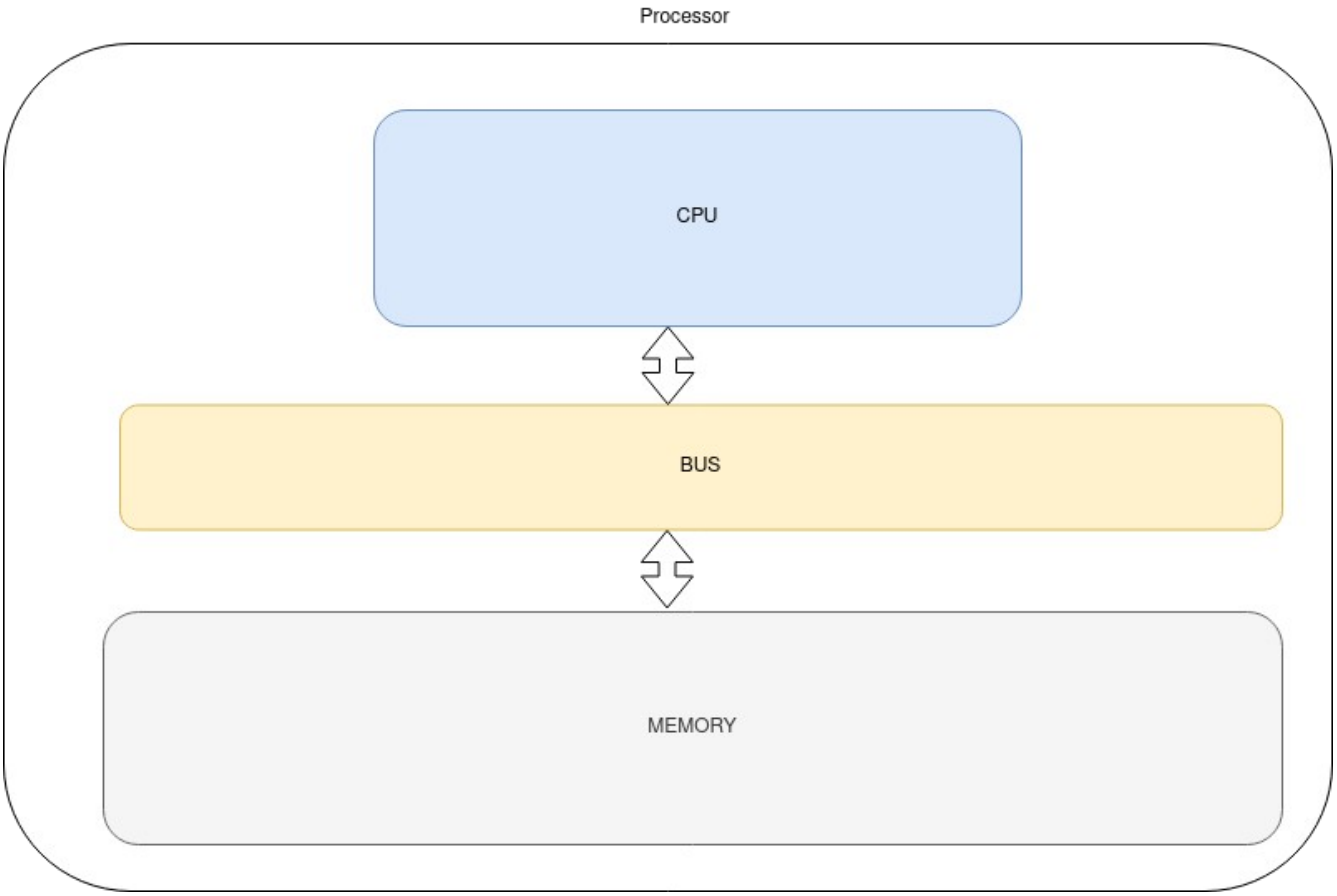
Bus based MOESI protocol

The code in this repository implements bus-based MOESI protocol. Following are components defined in the source code:



Each component is implemented using a separate class (All implementations are done using C++.).

For eg: Processor component is implemented in code: src/processor/

1) CPU

CPU consist of eight cores (implemented using Core class). Each core has its own cache and a set of queues.

List of queues in Core:

a) Instruction queue: This queue contains instruction for the code of a particular id. At the beginning of simulation, all the instruction from an input file and loaded into respective cores. The instructions are implemented in round-robin manner starting from Core 0 and moving to higher id.

List of valid instructions:

> Read: This instruction reads the address and core send CoreRead message to bus.

> Wrire: Core send CoreRead to bus which is followed by invalidation request.

b) Core to Bus queue: This queue is used by core to send message to bus (eg: CoreRead, CoreDataResponse). List of valid message in core to bus queue:

> CoreRead : This message is sent by Core if there is a read instruction to be executed.

> MemWriteBack: This message is sent when a dirty cache line needs to be evicted.

> InvalidateReq: This message is sent when there core sends invalidation request to bus when there is write instruction to be executed.

c) Core to Bus response queue: This queue is used to send response to bus. List of valid messages:

> CoreDataResponse: This message is sent in response to bus data request.

> CoreInvalidateAck: When bus send invalidate request to a core, it makes its cache line in INVALID state and send acknowledgment to bus using CoreInvalidateAck.

d) Bus to core queue: This queue has messages sent by bus. List of valid messages:

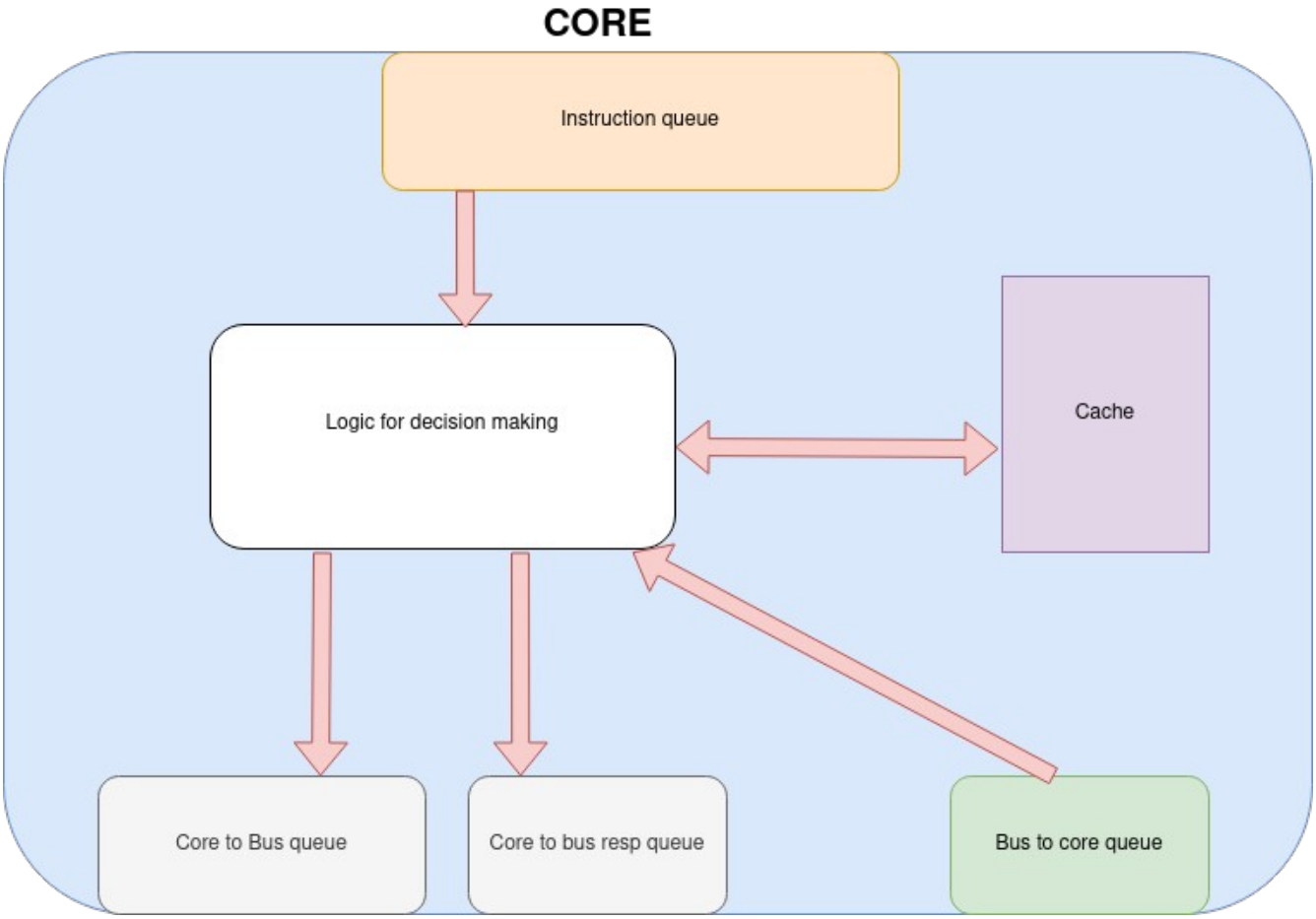
> MemWriteAck: This is sent in response to MemWriteBack when memory has been updated with latest data.

> CacheInvalidate: This is sent by bus to invalidate the cache line.

> BusInvalidateAck: This is sent by bus to the core which sent invalidate message to bus after all other cores have invalidated their valid cache line.

> BusDataResponse: This message provide data to requesting core. The data can come from either memory or other core which has cache line in modified and owned state.

> BusReadReq: This is sent by bus to a core whose cache line is in modified/owned state. The core responds by sending CoreDataResponse



Core diagram

2) Bus

The bus is responsible for responding to the request sent by core. Basic components of bus are as follows:

a) BusInfo: A data structure which keeps track of cores having the cache line and store cache state in each of core.

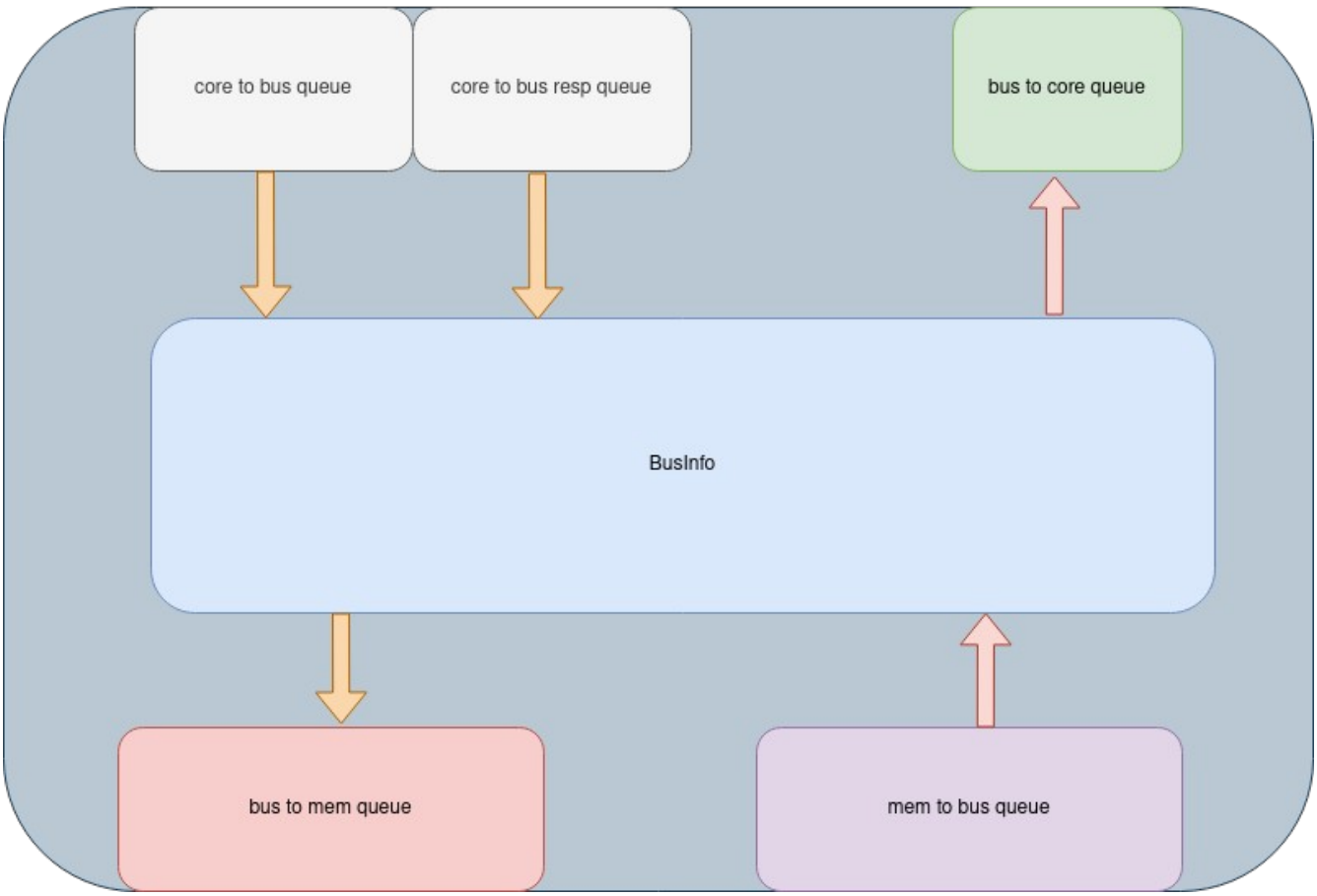
b) core to bus queue: Described in CPU section

c) core to bus resp queue: Described in CPU section

d) bus to core queue: Described in CPU section

e) bus to mem queue: This queue has messages sent by bus to memory which can be either readData or MemWriteBack.

f) mem to bus queue: This queue has messages sent by memory to bus which can be data or MemWriteAck


BUS
core to bus queue
core to bus resp queue
bus to core queue
BusInfo
bus to mem queue
mem to bus queue

Bus diagram

3) Memory

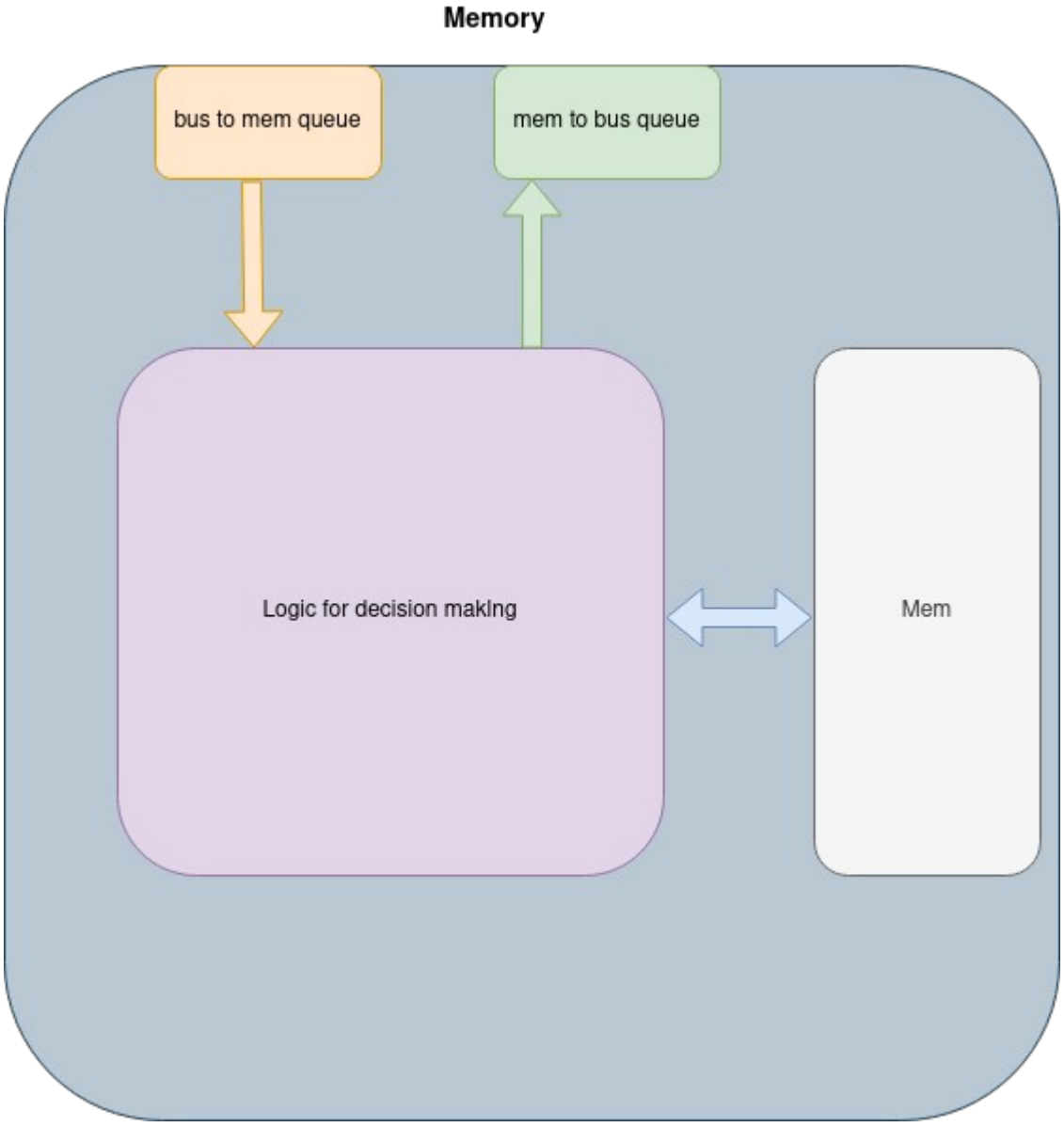
Memory consist of mem data structure which stores data for address from 0 to 63. Apart from that it has bus to mem queue and mem to bus queue as discussed in Bus section.

Valid bus to mem messages:

a) MemRead
b) MemWriteBack

Valid mem to bus messages:

a) MemWriteAck
b) MemData



Memory diagram

Relationship between different transactions

core_to_bus_tr : Transaction type for core to bus transaction.

bus_to_core_tr: Transaction type for bus to core transaction.

mem_to_bus_tr: Transaction type for memory to bus transactions

bus_to_mem_tr: Transaction type for bus to memory transactions

a) core_to_bus_tr:: CoreRead expects bus_to_core_tr :: BusDataResponse in reponse

b) core_to_bus_tr :: MemWriteBack expects bus_to_core_tr::MemWriteAck in response

c) core_to_bus_tr :: InvalidateReq expects bus_to_core_tr :: BusInvalidateAck in response to requesting core and bus_to_core_tr:: CacheInvalidate is sent to all other cores having corresponding cacheline.

d) bus_to_core_tr :: BusReadReq expects core_to_bus_tr :: CoreDataResponse in response

e) bus_to_core_tr :: CacheInvalidate expects core_to_bus_tr :: CoreInvalidateAck in response.