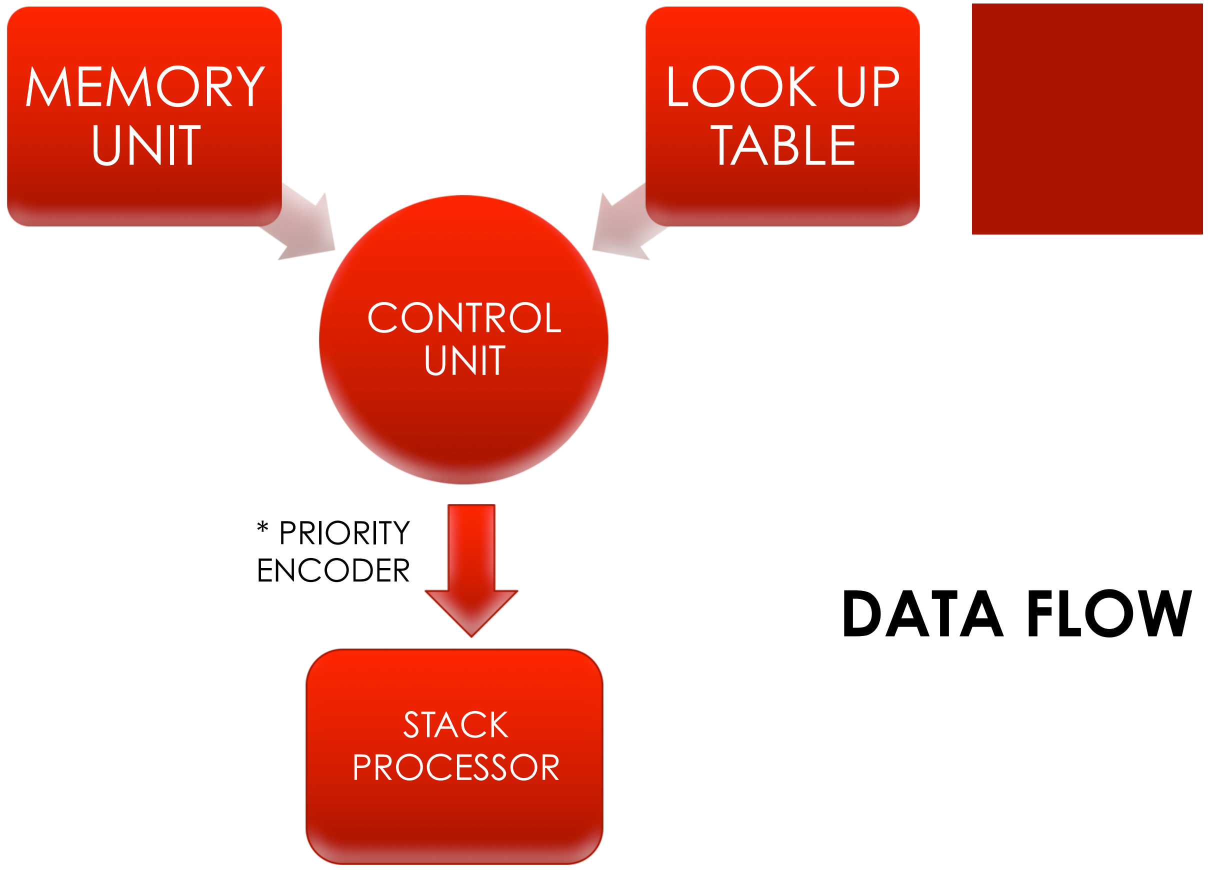
BReMICS

HARDWARE BASED IMPLEMENTATION

WORKING BLOCKS

- MEMORY UNIT (ROM)
- CONTROL UNIT
- LOOK UP TABLE
- STACK PROCESSOR
  - QUEUE
  - Q CONTROLLER
  - SCHEDULER
  - FPU

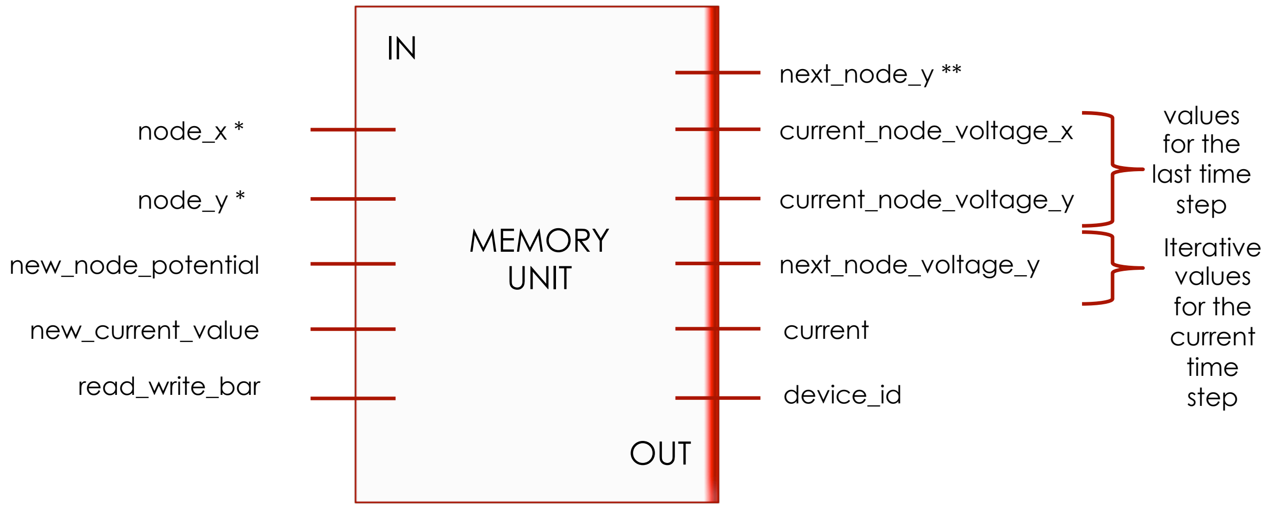


DATA FLOW

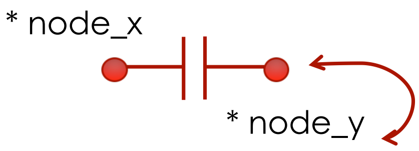
* THE DETAILS OF PRIORITY ENCODER AND ITS USE ARE GIVEN WITH THE CONTROL UNITS DESCRIPTION

MEMORY UNIT

- It receives a pair of NODE ID's from the control unit, and returns the device data, node potentials and current values.
- In addition to this, it receives updated values of voltage and current from the control unit and saves them in appropriate locations.



* While solving he KCL equation for a particular node, the node is marked as “node x”. The node adjacent to it, during the computation of its fix and coefficient s marked as “node y”.



In addition to this, the adjacency matrix is stored in the form of a sparse array.

Node voltages during the current time step and the next iterative step are stored in simple arrays and can be accessed with nodal indices assigned to each node at startup.

** As the control unit creates the nodal equation for a particular node “x”, the sparse array in the memory has a list of nodes adjacent to this node “x” and provides, along with the data corresponding to the node “y”, the ID of the next node adjacaent to node x.

Each device has a particular ID…

001	RESISTOR
010	CAPACITOR
011	INDUCTOR
100	DIODE -P
101	DIODE -N

The control unit picks up all this data from the memory and generates the fix and coefficient corresponding to the node. The memory required is significantly low because of the sparse storage.

Also, the read_write_bar pin enables the control unit to write data back to the memory

LOOK UP TABLE

- Has the expressions for 'fix' and 'coefficient'* equations of different kinds of devices (The fixes and coefficients are stored in a 2D vector)

For a resistor :

ELEMENT ID = 01;

FIX = $V_{2\,(n+1)}$ * (1/R);

COEFFICIENT = (1/R); // 1/R itself is the device value in this case.

We have two simple 1D arrays to store this data, one array for fix, the other for coefficient.

FIX[1] = { $V_{2\,(n+1)}$, 1/R, * } // array with '3' elements
COEF[1] = { 1/R } // array with a single element.

*** Instead of storing explicit values, the fix/coefficient arrays store select lines for each kind of data, i.e. device value, node potential etc, which is passed on to the control unit.

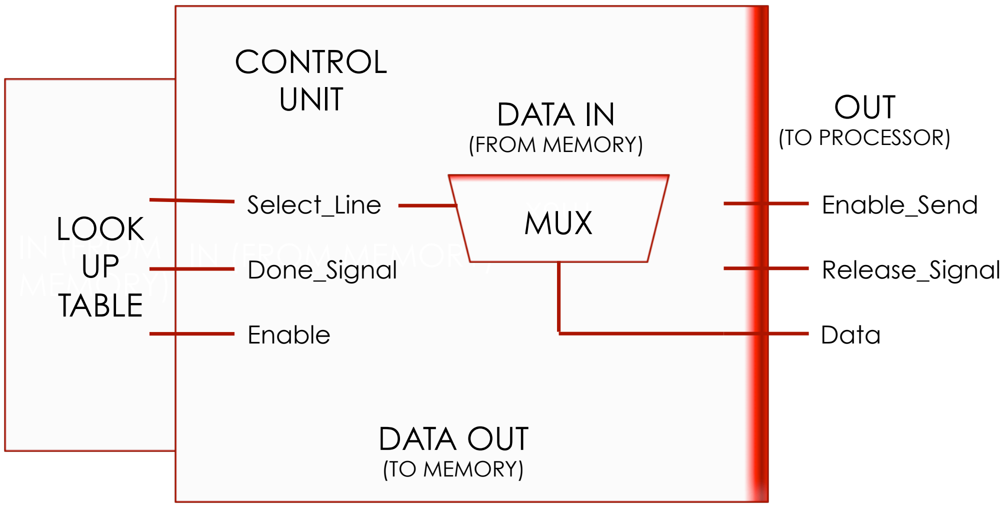
- Needs an 'ELEMENT ID' from the control unit to transfer the corresponding data

```
Input   [3:0]   ELEMENT_ID;      // 4 BIT element id ( device type )
Input           enable;          // data transfer begins at its rising edge
Output [3:0]   select_line;      //  4 BIT select line ( post fix format )
```

* Using Gauss Seidel's method to compute the potentials and currents iteratively

CONTROL UNIT

- Main function is to generate the KCL equation for each node and send it to the stack processor.
- The stack processor returns calculated values which are to be written back to the memory unit



OUTPUTS TO THE PROCESSOR

We have a single control unit operating several queues in the stack processor. So, we need to bind and release queues as and when required to the control unit.

A **priority encoder** looks for the vacant queues in the stack processor. Any free queue that is bound (i.e. currently receiving data from) the control unit is released at the rising edge of the Release_Signal. The encoder, at the same moment provides another free queue (if available) to the control unit for the computations for the next node.

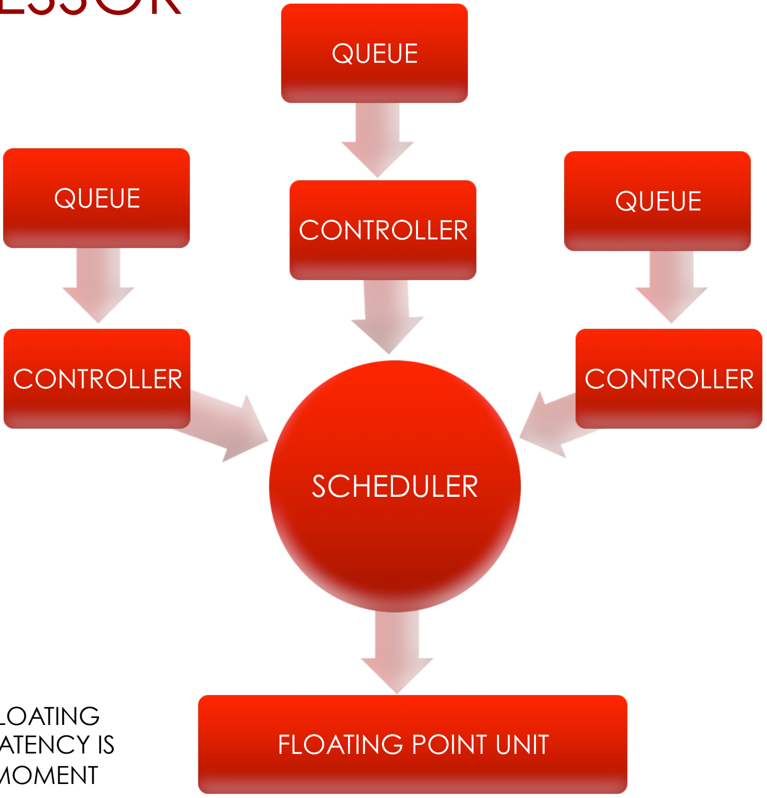
Queue begins receiving data at the Enable_Send signal.

The **"data"** similarly consists of an "18 BIT" number and a **2 bit "TAG",** this tag lets the processor identify whether the number is an operator, operand, Wait_Signal or an NR_Tag. This tag is generated in the Control Unit itself.

NOTE THAT THE RELEASE_SIGNAL ONLY IMPLIES THAT CONTROL UNIT HAS SENT ALL THE DATA TO THE QUEUE. SOME PART OF THIS DATA MIGHT STILL BE THERE IN THE QUEUE, WAITING TO BE SENT TO THE ALU. THUS, THE QUEUE MIGHT OR MIGHT NOT BE FREE

PROCESSOR

AT MOST "1" OF THE QUEUES IS CONNECTED TO THE CONTROL UNIT AT A TIME. THIS QUEUE RECIEVES DATA CONTINUOUSLY FROM THE CU.

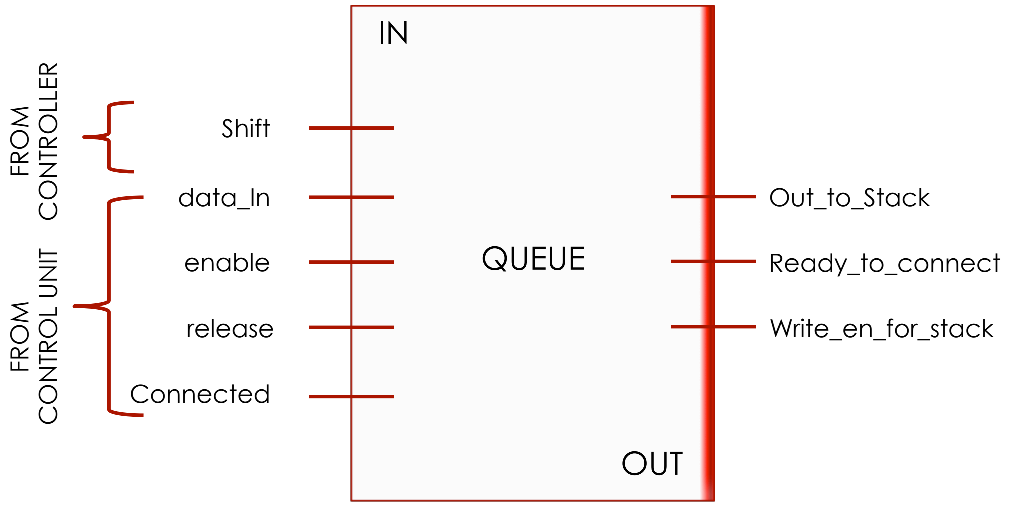


CONTROLLER IS THE BASIC STACK STRUCTURE THAT DOES THE POST-FIX CALCULATIONS

PIPELINED FLOATING POINT UNIT. LATENCY IS "4" AT THE MOMENT

QUEUE

- The Queue is a basic (FIFO) shift register which takes data continuously from the control unit when connected to it and shifts the first most data out whenever **shift** is high.
  - The 4 signals have specific functions for data input.
  - Shift is supplied by the controller of the queue and controls the data output.

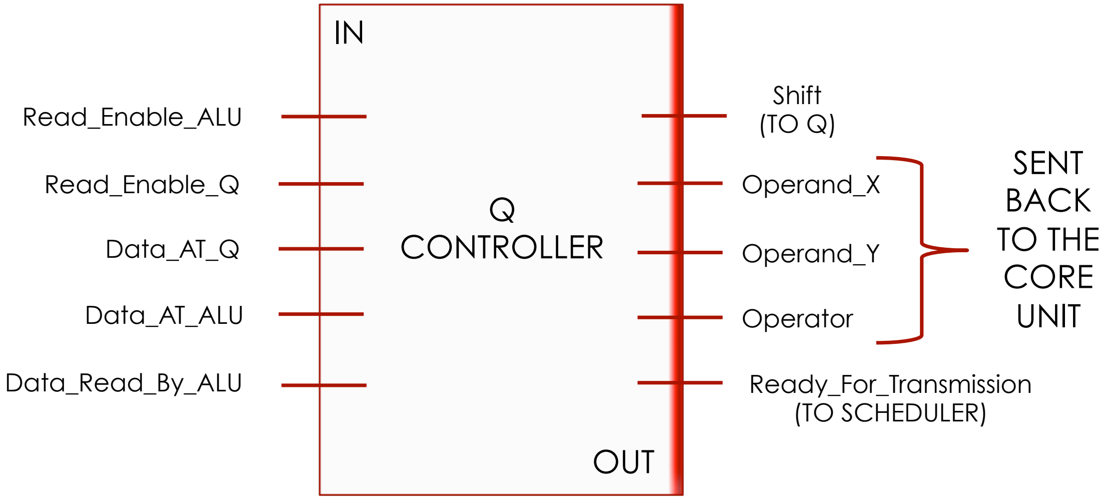


- The data at **“data_in”** is read by the queue when ENABLE is HIGH.
- **Release** tells the queue that no more data is coming in from the control unit. Whatever data remains must be shifted into controller.
- **Connected** turns low when all the data from the queue is shifted into the control unit and the next iterative value of node potential has been calculated.
- **Shift** turns low when the controller is not ready to receive data from the queue because of a pending calculation.

Queue has a pointer representing the position at which data has to be entered currently. Shifting happens as in an ordinary shift register.

Q CONTROLLER

- It reads data both from the ALU and its "Q". Data is read at the rising edge of respective enable signals.
- The data is stored in the form of a **"FILO stack".**
  - A pointer keeps track of the current write point.
  - Whenever an "operator" hits the pointer, two positions below it are checked, if both these positions are ready with operands, then the values are sent to the core and ready_for_Transmission is set to HIGH.
  - Operands might not be ready when the vaule at that location is yet to be received from the ALU.



SCHEDULER

- Scheduler has a slightly different version of the usual ROUND ROBIN method.
- Besides the usual shift registers used in the round robin method, we also incorporated a priority encoder.
  So, we now have a priority encoder in which the index corresponding to the highest priority keeps changing with every clock pulse.
- This way, the scheduler always gives a ready signal if any of the controllers is ready with an operation and at the same time, none of the queues is starved.

INPUT
READY SIGNALS FROM Q CONTROLLERS

OUTPUT
INDEX OF THE CONTROLLER THAT WILL FEED THE FPU

FLOATING POINT UNIT

FPU is not ready yet. Just for testing purpose we had coded a simple fpu which does operations in a single cycle and gives the output 3 cycles later.

Report By :
Archit Gupta
Nehal Bhandari