ASSIGNMENT NO.:

Title: Build a small compute cluster using Raspberry pi / BBB modules to implement booths multiplication algorithm.

Aim: Build a small compute cluster using Raspberry pi/BBB modules to implement Booths Multiplication algorithm

Objective: 1) Create Cluster of BBB.

2) Implement Booths Multiplication algorithm.

Theory:

Booths Multiplication Algorithm:

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeck College in Bloomsbury, London.

Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

Booth's algorithm examines adjacent pairs of bits of the N-bit multiplier Y in signed two's complement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for i running from 0 to N-1, the bits y_i and $y_{i-1}$are considered. Where these two bits are equal, the product accumulator P is left unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product.

The multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at $i = 0$; the multiplication by 2^i is then typically replaced by incremental shifting of the P accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest N bits of P.[1] There are many variations and optimizations on these details.

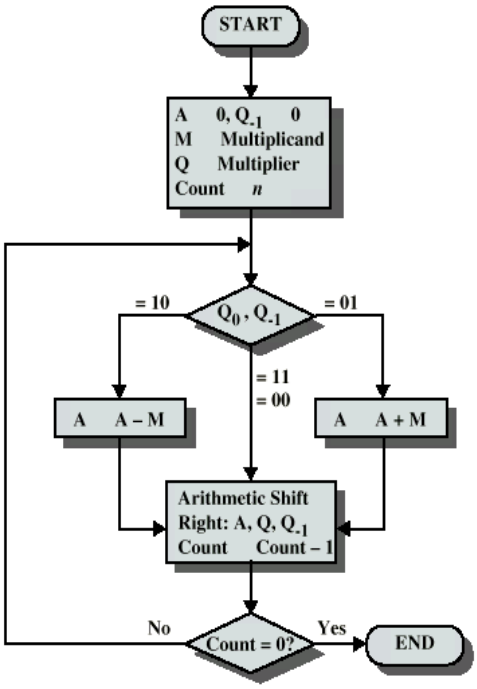
The algorithm is often described as converting strings of 1's in the multiplier to a high-order +1 and a low-order –1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

Booth's algorithm works because 99 * N = 100 * N - N, but the latter is easier to calculate (thus using fewer brain resources).

In binary, multiplication by powers of two are simply shifts, and in hardware, shifts can be essentially free (routing requires no gates) though variable shifts require either multiplexers or multiple clock cycles.

Thus instead of multiplying n * 7 as (n * 4) + (n * 2) + (n * 1) which requires 2 additions, Booth recoding allows us to implement it as (n * 8) - (n * 1) requiring one subtraction.

Booths Multiplication Flowchart:



Advantages:

Booth's Multiplication Algorithm (invented by A.D. Booth in 1951) takes advantage of the fact that the time taken for multiplication depends on the number of 1's in the multiplier. Multiplication by 1 involves addition of the multiplicand to the partial product, but multiplication by 0 does not.

Disadvantages:

The time taken for multiplication increases with the number of 1's in the multiplier.

Creating Cluster of Beagle Bone Black :

Step1. Connect BeagleBoneBlack(BBB) to your Machine(Ubuntu) via USB cable.

Step2: Connect Ethernet cable to BBB.

Step3: Open Terminal in Machine(Ubuntu) and fire following commands

Step4: sudo apt-get install minicom (only if you don't have minicom installed)

Step5: sudo minicom –s

Step6: go to serial port setup

Step7: Change device name to /dev/ttyACM0 //ACM(zero)

Step8: make option 'F-Hardware Flow Control' to =No

Step9: make option 'G-Sofware Flow Control' to =No

Step10: save as dfl

Step11: Exit

Step12: Enter username=debian

Password=temppwd

Step13: sudo su

Step14: Change hostname to beagle1 using command : hostname beagle1

Step15: su – debian

Step16: find out ip address of BBB using ifconfig –a

Step 17: perform steps 5 to 17 for each node. Put ip-host pairs of every nodes of cluster in /etc/hosts by

editing into the file /etc/hosts

Step18:save and exit wq

Step 19: generate ssh key pair using ssh-keygen

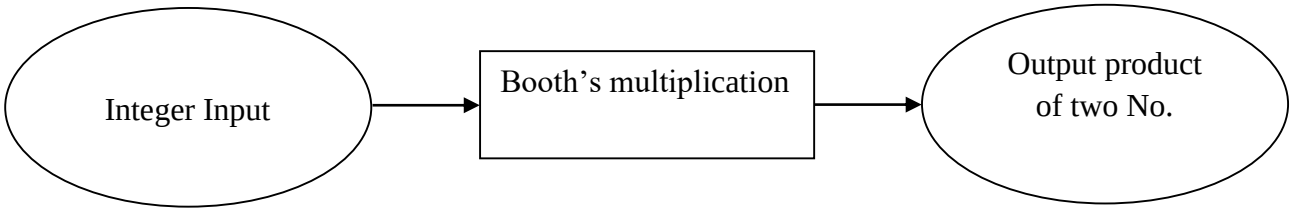
Step20: Perform steps. Now from each node in cluster share public key of each node to master by performing following command. Following command share beagle1's public key to beagle2 It will ask for password of beagle2 which is "temppwd" by default.

Step21: install MPICH2 package using apt-get install mpich2

Input: Multiplier & multiplicand

Output: Product

Mathematical Model:



Let S be the system such that,

S= {I,O,Fn,Sc,Fc}

I-Input Set

Integer Values

O-Output set

Product of the inputs

Fn-Function Set

F1 – Binary Conversion

F2 – Booths Multiplication

Sc – Success Set

Sc1 – Correct product obtained

Sc2 – Binary conversion is correct

Fc – Failure Cases

Fc1 – Binary conversion incorrect

Fc2 – multiplication incorrect

Platform: Ubuntu 16.04

Programming language: C

Conclusion: Booth's Multiplication is implemented on cluster of BBB.