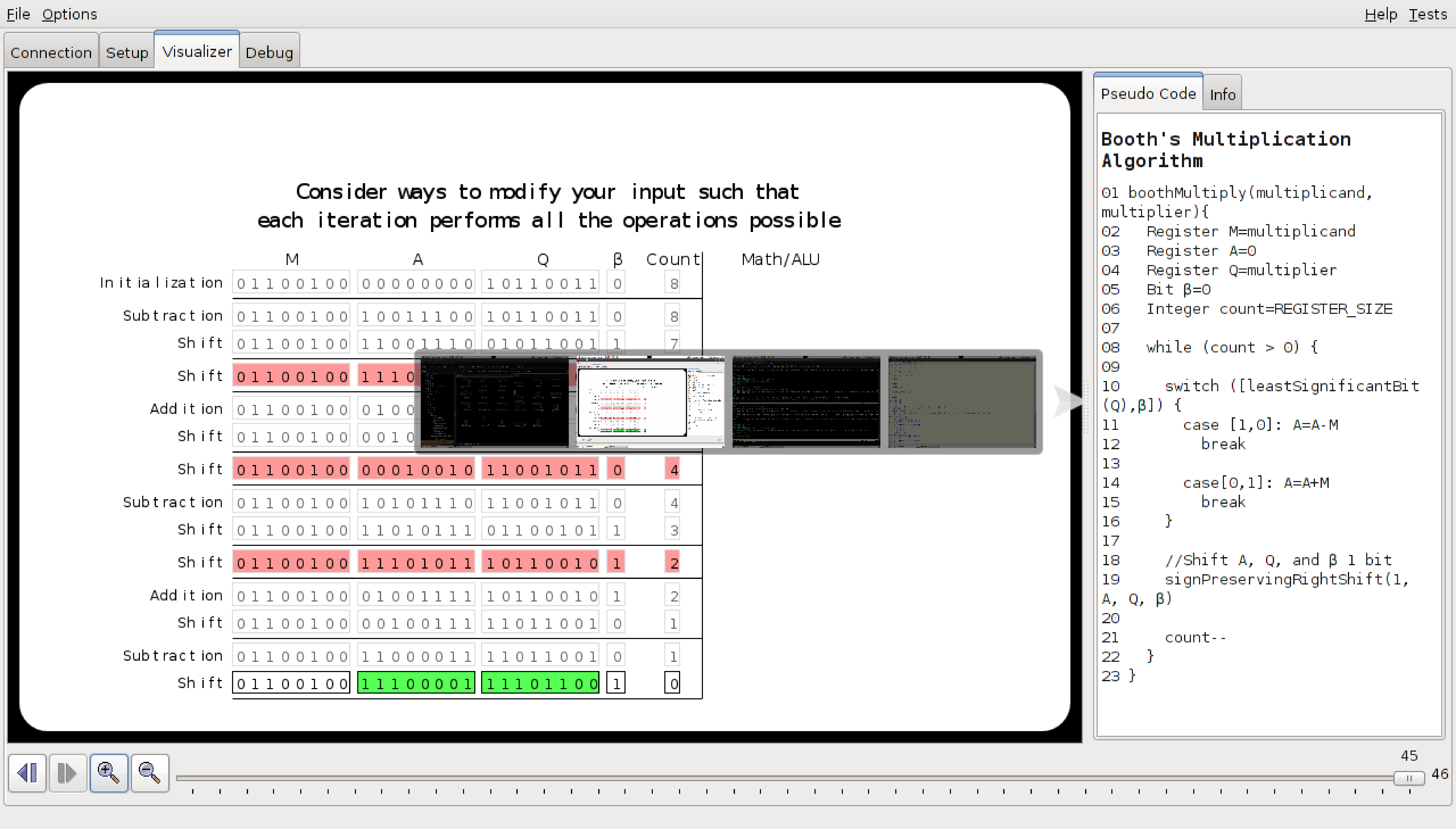

Consider ways to modify your input such that
each iteration performs all the operations possible

	M	A	Q	β	Count	Math/ALU
Initialization	01100100	00000000	10110011	0	8	
Subtraction	01100100	10011100	10110011	0	8	
Shift	01100100	11001110	01011001	1	7	
Shift	01100100	1110				
Addition	01100100	0100				
Shift	01100100	0010				
Shift	01100100	00010010	11001011	0	4	
Subtraction	01100100	10101110	11001011	0	4	
Shift	01100100	11010111	01100101	1	3	
Shift	01100100	11101011	10110010	1	2	
Addition	01100100	01001111	10110010	1	2	
Shift	01100100	00100111	11011001	0	1	
Subtraction	01100100	11000011	11011001	0	1	
Shift	01100100	11100001	11101100	1	0	

Booth's Multiplication Algorithm

```
boothMultiply(multiplicand, multiplier){
  Register M=multiplicand
  Register A=0
  Register Q=multiplier
  Bit β=0
  Integer count=REGISTER_SIZE

  while (count > 0) {

    switch ([leastSignificantBit(Q),β]) {
      case [1,0]: A=A-M
        break

      case[0,1]: A=A+M
        break
    }

    //Shift A, Q, and β 1 bit
    signPreservingRightShift(1, A, Q, β)

    count--
  }
}
```