CONTEXT

Take this simple multiplication circuit. We can run nargo prove p --print-acir

just to print the ACIR opcodes, which are shown here:

BLACKBOX::RANGE [(*1, num_bits: 32)] [ ] BLACKBOX::RANGE [(_2, num_bits: 32)] [ ] BLACKBOX::RANGE [(_3, num_bits: 32)] [ ] EXPR [(1, _1, _2) (-1, _4) 0] DIR::QUOTIENT (out :* %EXPR [(1, *4, _4) 0]%, (_6, %EXPR [2^{32}]%), _5) *BLACKBOX::RANGE [(_5, num_bits: 32)] [ ] BLACKBOX::RANGE [(_6, num_bits: 96)] [ ] EXPR [(-1, _4, _4) (-1, _14) 0] EXPR [(1, _5) (2^{32}, _6) (1, _14) 0] EXPR [(1, _5, _5) (-1, _7) 0] DIR::QUOTIENT (out :* %EXPR [(1, _7, _7) 0]%, (_9, %EXPR [2^{32}]%), _8) BLACKBOX::RANGE [(_8, num_bits: 32)] [ ] BLACKBOX::RANGE [(_9, num_bits: 96)] [ ] EXPR [(-1, _7, _7) (-1, _16) 0] EXPR [(1, _8) (2^{32}, _9) (1, _16) 0] EXPR [(-1, _3) (1, _8) (-1, _10) 0] DIR::INVERT (_10, out: _11) EXPR [(1, _10, _11) (-1, _12) 0] EXPR [(1, _10, _12) (-1, _10) 0] EXPR [(1, _12) 0]

We are attempting to translate these expressions into a standard plonk gate. What is already clear:

- The first element (q

) in a tuple represents the selector, and the “_

” prefixed values are witness indexes.

- (q, _l, _r)

represents a multiplicative term, and (q, _x)

represents a linear combination

QUESTIONS

1. When q = 1

, this denotes a selector toggled on, right?

1. Does q = -1

denote a selector toggled off, such that the selector should be witnessed as 0? If not, how is q = -1

interpreted?

1. Infrequently, we see q = 2^{32}

. How is this value interpreted as a selector?