Circom Exercises for ACM India Summer School 2024
Indian Institute of Technology Bombay
Instructor: Saravanan Vijayakumaran

Date: June 8, 2024

1. Write circom code for an `IsEqual` component (subcircuit) that has two inputs and one output. The output is 1 when the inputs are equal and 0 otherwise. How many R1CS constraints does your `IsEqual` component require?

2. Write circom code for a `Mux4` component (subcircuit) that has six inputs `x0`, `x1`, `x2`, `x3`, `s0`, `s1`, and one output `out`. The inputs `s0` and `s1` must be forced to take boolean values. If `s0 + 2s1 = i`, then the output `out` should equal `xi`. How many R1CS constraints does your `Mux4` component require?

3. Consider a `MultiAnd` circom component that has n inputs `in[n]` and one output `out`. The value of n will be a fixed constant greater than 2. The inputs must be forced to take boolean values. The output `out` should be equal to the AND of **all** the inputs. Write circom code for the `MultiAnd` component that uses only $n + 2$ R1CS constraints.

4. In the Bitcoin protocol, the timestamp in the a block header is required to be greater than or equal to the median of the timestamps in the previous 11 block headers. Timestamps are 32-bit unsigned integers measuring the number of seconds since Jan 1st, 1970 at UTC.

   Write circom code for a `MedianVerify` component that has two inputs: a 11-element array `ts[11]` and a signal `med`. The component should verify that `med` is equal to the median of the values in `ts`. It should also verify that both `med` and the values in `ts` fit in 32 bits.

   Note that the `MedianVerify` component **does not calculate** the value of `med`. It merely verifies that a given value of `med` is equal to the median.

   Hint: Use the strategy in the following pseudocode inspired by the implementation in ZeroSync.[1] In the pseudocode, `sign` computes the sign of an integer, returning 0, +1, or −1. `abs_value` computes the absolute value. `assert` checks that its argument is true; if it is false the `assert` fails.

```
let n_median_occurrences = 0
let signs_diff = 0

for i in range(11):
    let delta_sign = sign(ts[i] - med)

    if delta_sign == 0:
      n_median_occurrences += 1
    else
      signs_diff += delta_sign

let absolute_signs_diff = abs_value(signs_diff)

assert(absolute_signs_diff + 1 <= n_median_occurrences)
```

[1] https://github.com/ZeroSync/ZeroSync/blob/main/src/block/compute_median.cairo