I. 1.d Multistate Logic

A combination of n binary digits (bits), can represent 2^n unique numbers. In this case, this system represents the set $\{x \in \mathbb{Z} | 0 \leq x < 2^n\}$. In the first section, the application of a set of operations, on this system is analysed. Later, the system is expanded by a special "Undefined" token. The analysis of these 2 Systems, is then transferred to bigger combinations of bits.

A. Operations

- AND: An and Operation on 2 binary numbers, can be performed bitwise, which for any 2 n digit binary numbers, yields a n digit number as a result
- OR: Similarly to the AND operation, the OR operation can be performed bitwise, to yield a number in the range of $[0, 2^n]$
- XOR: The XOR operation of 2 bits, yields 1 if only one of the inputs is 1 and otherwise yields 0
- NOT: A NOT Operation is usually performed to incert a single bit, for a comparison of 2 numbers, the Operation is performed NOT EQUAL
- ADD: The ADD Operation, adds the 2 numbers and truncates them to the last 3 bits, which are represented by the bit sequence
- SUB: The SUB Operation, subtracts the 2nd number from the 1st and truncates them to the last 3 bits, which are represented by the bit sequence
- MUL: The MUL Operation, multiplies the 2 numbers and truncates them to the last 3 bits, which are represented by the bit sequence
- DIV: The DIV Operation, diviedes the 1st number by the 2nd and truncates them to the last 3 bits, which are represented by the bit sequence

B. Incorret Results

An incorrect result, can appear in all of the non primary Operations, namely the ADD, SUB, MUL and DIV. These can be either backtraced to an over or underflow, in the case of ADD, SUB or MUl, or are the result of integer truncation and division by 0. An overflow, occurs when the result needs more digits than are available, namely if $x \geq 2^n$. An underflow, is the opposite of the overflow and happens when the result of the calculation is smaller than the smallest representable number, in our case $x < 0$. For a DIV operation, for every operation x/y where $x \mod y \neq 0$ the result is incorrect, as the number system does not represent fractions.

Listing the Conditions for a valid result of the non primary operations on x and y in an n bits system, yields the following List.

- ADD:
  - $x + y < 2^n$
- SUB:
  - $x \geq y$
- MUL:
  - $x * y \leq 2^n$
- DIV:
  - $x \mod y = 0$
  - $y > 0$

Applying these Conditions to the given set of numbers, 28 faulty additions, 28 faulty substractions, 33 faulty multiplications and 41 faulty divisions are performed. In total $2^{3^2} * 8 = 512$ operations are performed, from which $2^{3^2} * 4 = 256$ are non primary. This results in an error rate of 25.4% for all operations or respectively an error rate of 50.8% for all non primary operations.

C. Change due to implemention of $0b111$ as "Undefined"

When implementing a specific combination of bits, as the undefined Token ("Undefined"), it has to be considered, what happens when executing operations on "Undefined". There are 2 obvious ways of dealing with this. The first one being, to consider every operation involving "Undefined" as the identity operation $x \mapsto x$. On the other hand, it can be defined that every operation involving "Undefined" results in undefined $(x, "Undefined") \mapsto "Undefined"$. For the following analysis, the second option is considerd.

Due to the implemention of "Undefined", the representable set of numbers in n bits has shrunken to $2^n - 1$, or in our case to $\{x \in \mathbb{Z} | 0 \leq x \leq 6\}$. However, if we define, that the result of a faulty operation, is "Undefined", the number of incorrect Operations, decreases by definition. While this is holds for amount of faulty operations, the amount amount of wrong numerical operations increases, as the Set of correct numerical calculations shrinks because, as amongst other things, the condition for correct results in addition shrinks to $x + y < 2^n - 1$ from $x + y < 2^n$.

D. Increasing the length of bits

When the numbersystem is increased to represent the set $\{x \in \mathbb{Z} | 0 \leq x < 2^4\}$, the number of faulty operations increases to 745, while the number of performed operations increases to $2^{4^2} * 8 = 2048$, which increases the percentage of faulty operations to 36.4%. Respectively for a 5 bit system, ther error percentage increases to 40.4%. This trend is very logical, as especially in multiplication, the number of invalid operations, increases considerably faster than the numbve rof valid operations. However the amount of valid operations in division increases with a bigger number System. This growth however is not strong enough to counteract the increase in invalid operations in multiplication.