# Xtilemod

## Section 1 : Introduction

What is Xtilemod all about?
Xtilemod is bunch of java codes which on compilation generate a .tile file. This is a file which can be used to make DNA tile equivalent (refer to next section) in real world, which would do arithmetic computation according to input tile-set provided to it. We use a software named Xgrow (which currently runs on Linux base) to observe the output simulation of that .tiles file.

### 1.1 DNA Tiles Equivalent

Tile is a term used to describe a set of four glues which characterize a DNA 3'5' ,5'3' sticky ends which are at 4 locations and therefore characterized by a tile.

### 1.2 Basic Functionalities of Xtilemod

Xtilemod does a wonderful job in unifying a large chunk of arithmetic computations which can be simulated in real world by DNA combinations to get the computed output. This includes addition, subtraction, multiplication and division of integers. Primarily and modulus are the other two features added to it. Not only does it provide these basic arithmetic functionalities at a single platform but it also provides flexibility of choosing the type of implementation at user's part.

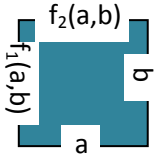


Figure 1

Figure 1 on the left shows a typical tile with single inputs. Here as you can see, the North and the west tile are the glues which correspond to the combination of input which is the East and South tile. When you open up a .tiles file you will see the below (excluding the header discussed in the next sub-section):

**tile edge matches {{N E S W}*};**

This shows that each tile can be written as a 4-Dimentional vector with each tuple containing values of glue in all direction of 2-D plane. It looks something like this :

**{1 5 0 6}[.1](yellow)**

The additional specifications shown are the concentration term in square braces and the color of the tile in Xgrow, in the curly braces. There are few things noteworthy, that is glue 0 is by default used to represent glue strength of 0.

### 1.3 Header provided by codes generated by Xtilemod

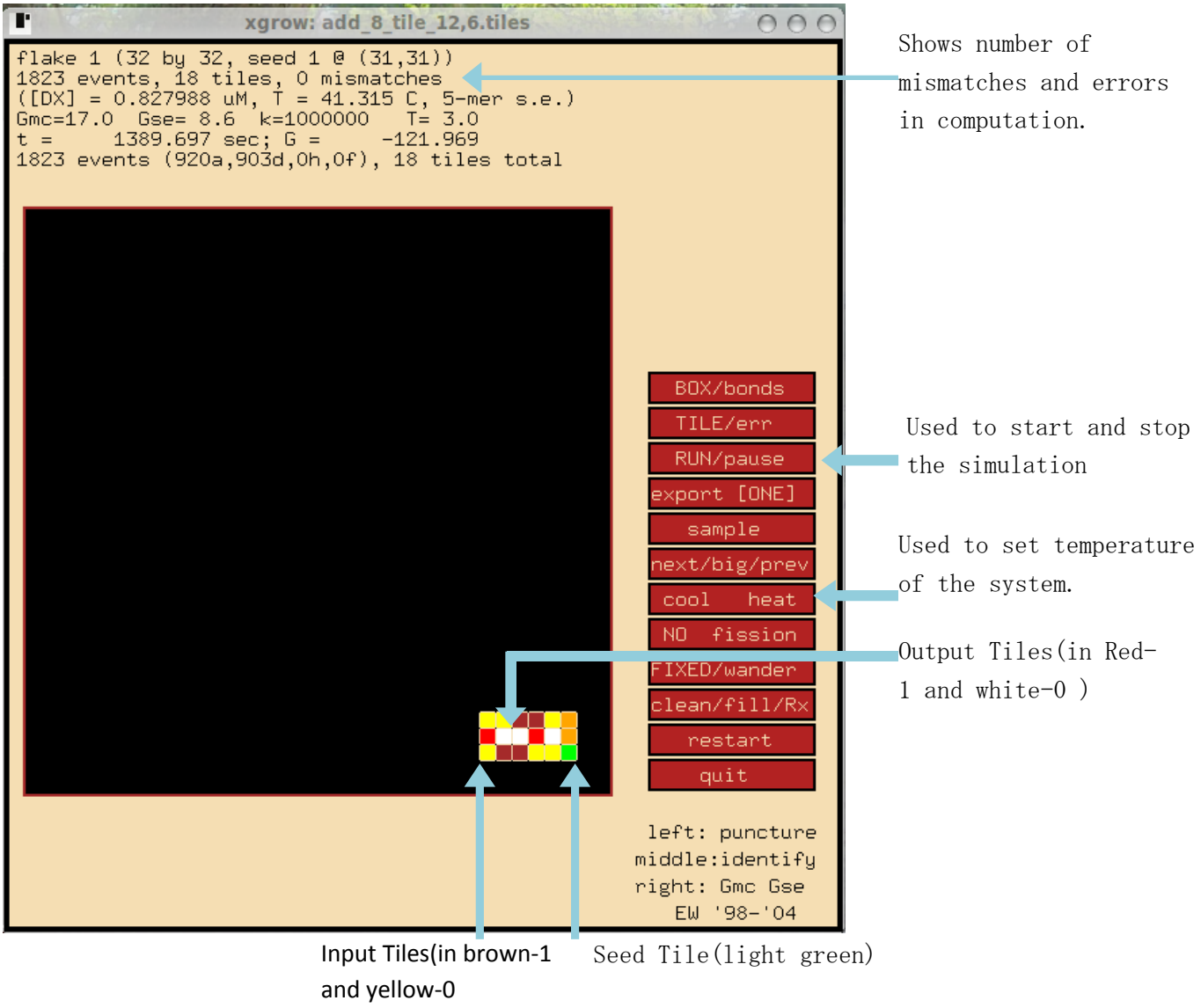
Xtilemod provides understandable header before the actual tileset in the .tiles file to mirror the functionalities of the header.

```
% add_L_var_12,6.tiles (L configuration)
% Mentor : Manish Gupta
% Author : Sandeep Vasani, Abhishek Chhajer, Jaley Dholakiya
%Colouring Pattern
%Seed Tile :    Green
%Output Tile:   Red = 1   White = 0
%Input  Tile:   Brown = 1      yellow = 0
% Addition tile set of 12,6
% The top most row shows the sum of 12, 6

%-----Inputs in Binary------

%      12       1100
%     +6        110
%--------------------------
% sum: 18       10010
```

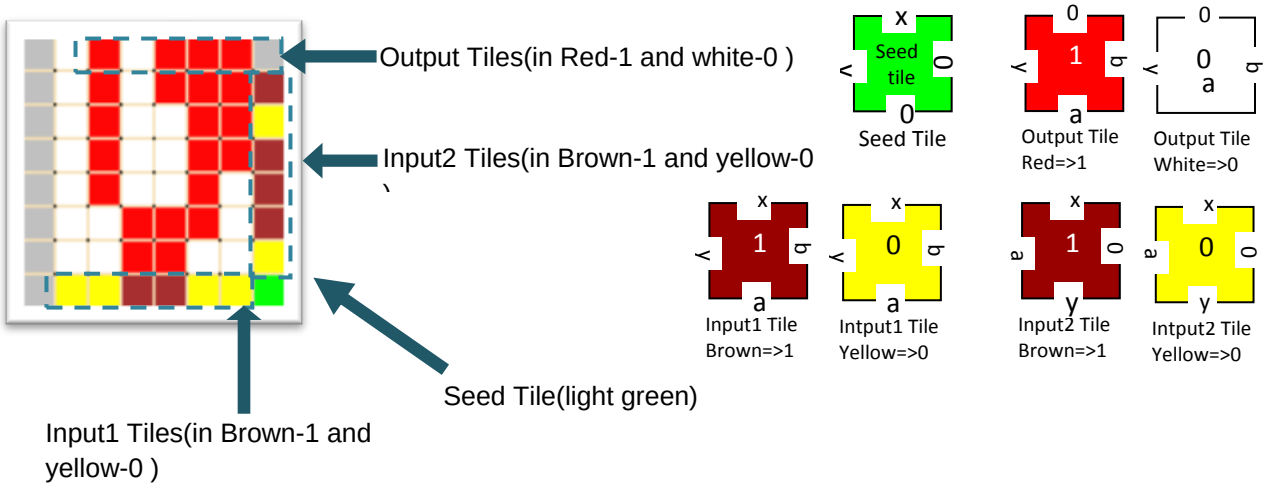
The first non-whitespace line represents type of implementation. This in followed by the name of mentor and author. Next 4 lines specify the coloring pattern of input, output and seed tile which helps in conceiving the output from the xgrow simulator. It is followed by name and type of arithmetic operation along with the inputs. The remaining lines in the header, shows expected answer and the inputs in binary as well as in decimal. Note that the code does run only for integers.

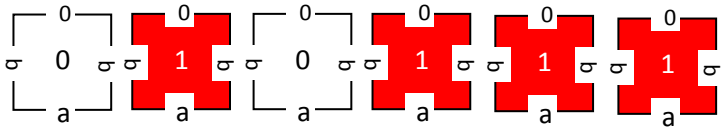



## Section 2: Analyzing the output

This section will give you an insight of how to infer the computational result from the output display shown by xgrow. Color of the input and output tiles are referred in both .tiles file of same and Read me provided. In general the output is generally inferred from the north of rectangular tile structure. The exceptions to it are 8 tile additions and division.

## 2.1 Default Tile Set Inference:

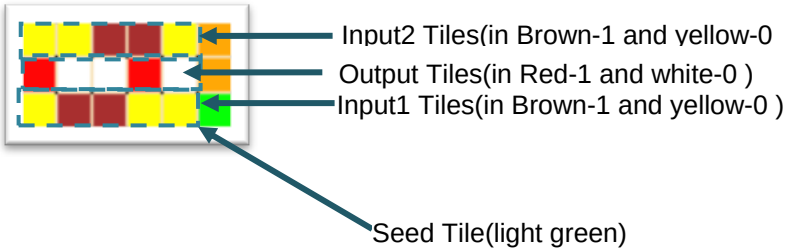


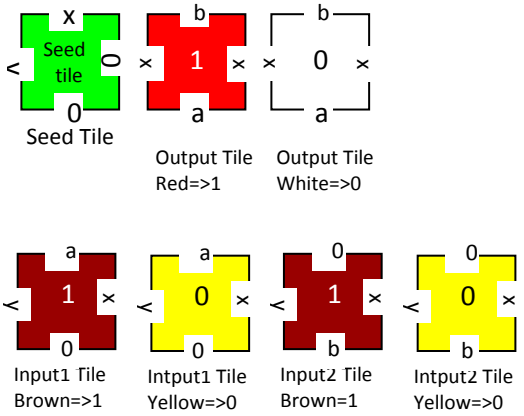



In this case the output will be :

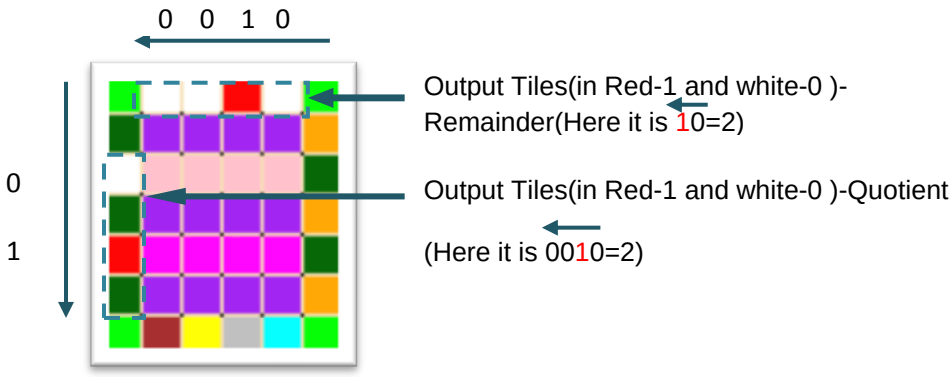
**Output** : 0 1 0 1 1 1

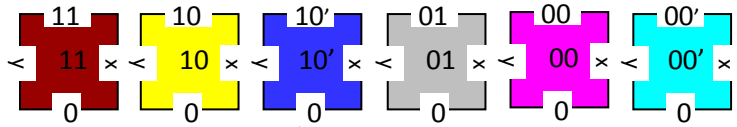
# 8 tile Addition



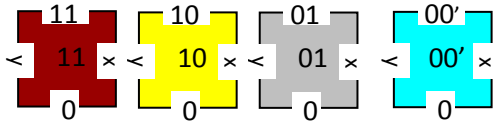

In 8 tile type addition the major difference is that the output is a




All Input Tile Set

Current Input Tile Set




11 10 01 00’

Dividend : 1100 = 12

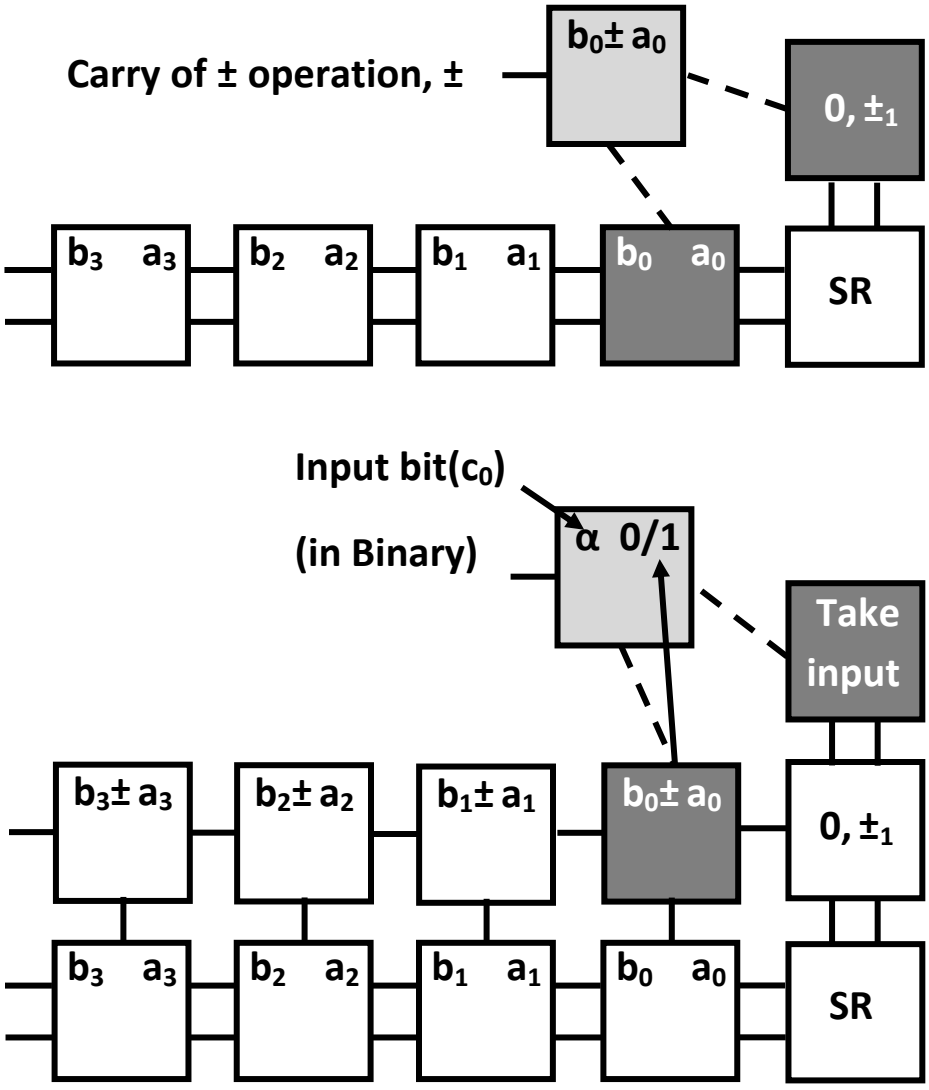
11 10 01 00’

Divisor :101 (0’ is for padding and aligning)

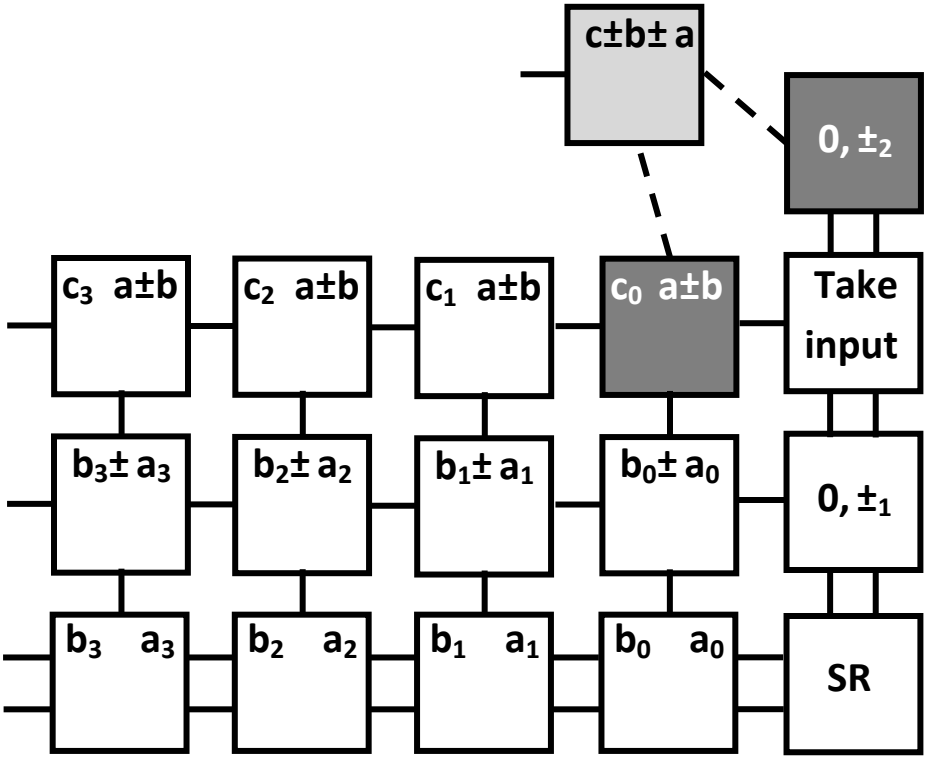
In division the quotient is on the left side of the rectangular tile structure while the remainder is on top or North of the tile structure.

## Add Subract Modulo Adder

(Note : $\pm_i$ is symbol used to denote addition/subtraction of $i^{th}$ and $(i-1)^{th}$ input integers. Output is denoted by $\mathbf{a\pm_1 b\pm_2 c\pm_3 d\pm_4}$ **- - - -** )




**Output after tiling c, $a\pm_1 b$**




**Output at end of this tile addition : $a \pm_1 b \pm_2 c$**