Memory:
- 32 bit header: N, M, Count, mode, FP
- 64 x 1: for L
- 64 x 2: for two h's
- 64 x 50 x 2: for two U's
- 64 x 50 x 6: for X_w,(0:5), X_0 and the five output values…w stands for warehouse.
- 64 x 50 x 50 x 2: for A and B.
- 64 x 3: for 0.9 value (constant)
- 64 x 1: for address follower/counter.
- 64 x 1: for err.
- 3 x 1: for C
- God only knows…!

FPU: two for now..

Solver:

1- The Yellow part (initialization):
Solver checks on the load signal, and listen to its addresses.
if it happens that one of his addresses is placed at the address bus, it will read/store the data placed at the data bus, and increment its address pointer…and so on.

2-The Green part (fixed step size):
    2_1-The Purple part (regular operations):
        Solver sends 'h' to interpolator
        calculates part_1 of Xn+1 and place it at X_i, X_i = (1+hA) * X_w[c]
        wait until solver receive 'done' signal from interpolator
        calculate part two of X_i equation, X_i += h * B * U_h
then solver checks if with the 'done' signal it received an OP signal?
    if so, it places X_i into X_w[c+1] and increments c
    if not, resume…
check for any existing error during the whole operation. And if so inform CPU
if not, check did [C == count] ?
if so, send success to CPU,
if not, repeat.

3- The Red part (variable step size):
- First divide h by 2 and calculate the two steps equations and place the final output at X_w[c+1]
- Multiply 'h' by 2, and calculate the one step equation and place it at X_i
- calculate err and check if err <= L ?

-if not, calc new h and back to ‘C’, NOTE: YOU NEED TWO U’S FOR THAT REASON!

-if so, do a little placing in order to get to the exact *purple* stage.

-if olver received an OP signal, it will store the output, inc. C, and restore the old h, , NOTE: YOU NEED TWO h’S FOR THAT REASON!

-repeat..