memory_stark

draft

PADDING	ADDR	CLK	OP	VALUE	DIFF_ADDR	DIFF_ADDR_INV	DIFF_CLK
0	100	0	SB	5	0	0	0
0	100	1	LB	5	0	0	1
0	100	4	SB	10	0	0	3
0	100	5	LB	10	0	0	1
0	200	2	SB	15	100	3504881373188771021	0
0	200	3	LB	15	0	0	1
1	200	3	LB	15	0	0	0
1	200	3	LB	15	0	0	0

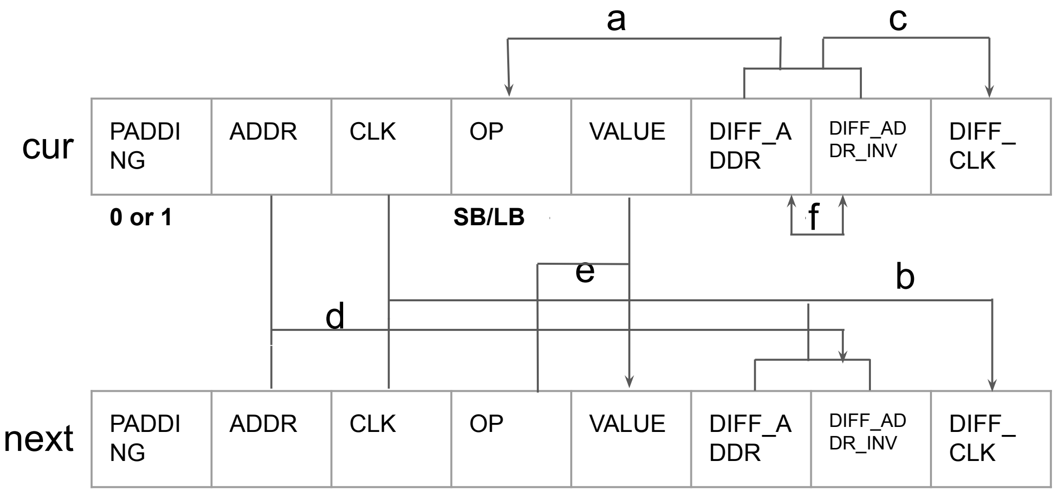
Constraints

VM

Sorted by addresses and then the clock

ADDR	CLK
100	0
100	1
100	4
100	5
200	2
200	3

STARK



cur	PADDING	ADDR	CLK	OP	VALUE	DIFF_ADDR	DIFF_ADDR_INV	DIFF_CLK
next	PADDING	ADDR	CLK	OP	VALUE	DIFF_ADDR	DIFF_ADDR_INV	DIFF_CLK

new_addr = diff_addr*diff_addr_inv

a) if new_addr: op === sb

b) if new_addr == 0: diff_clk_next <== clk_next - clk_cur

c) if new_addr == 1: diff_clk === 0

d) diff_addr_next <== addr_next - addr_cur

e) if op_next == lb: value_next === value_cur

f) (new_addr - 1)*diff_addr===0

(new_addr - 1)*diff_addr_inv===0

todo) range check: diff_addr is a u32

diff_clk < run time?

Some notes

LB inst:

Addr_next === addr_cur

Value_next === value_cur

When address changed:

OP must be SB

Calculate diff_addr

In the previous design, there is no way to make the following constraint:
if New_addr==1, then Diff_addr>0

New_addr	Diff_addr
0	0
1	100
0	0

We can make the constraints using the current design:

(diff_addr*diff_addr_inv - 1)*diff_addr===0
(diff_addr*diff_addr_inv - 1)*diff_addr_inv===0