1 Definitions

1.1 Misc

Let $m = \left\lceil \left(\frac{N}{102}\right)^{\frac{1}{d}} \right\rceil$, base of the counter

$MSR =$ most signifcant digit region

$\mathcal{C}_0 =$ starting value of counter

$d = \lceil \log_m \mathcal{C}_0 \rceil = \left\lfloor \frac{k}{2} \right\rfloor$, number of digits per row

$\mathcal{C}_f = m^d$, final value of the counter

$\mathcal{C}_\Delta = \mathcal{C}_f - \mathcal{C}_0$, number of rows/ times to count

$l = \lceil \log m \rceil + 2$, bits needed to encode each digit in binary, plus 2 for MSR and MSD

1.2 Determining the starting value

...therefore, let $d = \left\lfloor \frac{k}{2} \right\rfloor$, $m = \left\lceil \left(\frac{N}{102}\right)^{\frac{1}{d}} \right\rceil$, $l = \lceil \log m \rceil + 2$, $\mathcal{C}_0 = m^d - \left\lfloor \frac{N-12l-76}{12l+90} \right\rfloor$, where d is the number of digits per row of the counter, m is the base of the counter, l is the number of bits needed to encode each digit in binary plus 2 for indicating whether a digit is in the MSR and is the MSD in that region, and $\mathcal{C}_0$ is the start of the counter in decimal.

In general, the height of a digit region is $12l + 90$. There are two cases when the height is different, namely in the first and last digit regions, where the height is $12l + 91$ and $12l + 75$, respectively. Let h be the height of the construction before any filler/roof tiles are added. If we define $\mathcal{C}_\Delta$ as the number of `Counter` unit rows, then $h = (\mathcal{C}_\Delta - 1)(12l + 90) + (12l + 91) + (12l + 75)$, simplifying to $\mathcal{C}_\Delta(12l + 90) + 12l + 76$. So then the maximum height of the counter is $m^d(12l + 90) + 12l + 76$. Since our goal is to end with a rectangle of height N, we need to pick a base such that the counter can increment so many times that when it stops, it is at least N.

Lemma 1. $N \leq m^d(12l + 90) + 12l + 76$.

Proof.

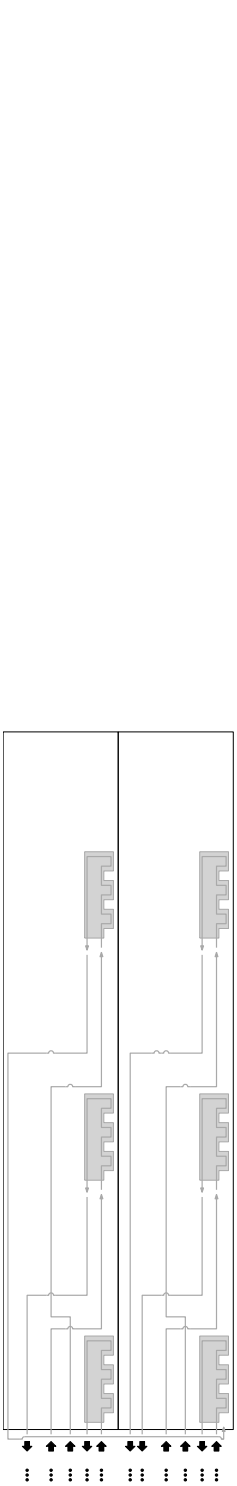
$$\begin{aligned} N = 102\left(\frac{N}{102}\right) = 102\left(\left(\frac{N}{102}\right)^{\frac{1}{d}}\right)^d &\leq 102\left\lceil\left(\frac{N}{102}\right)^{\frac{1}{d}}\right\rceil^d \\ &= 102m^d \leq 12lm^d + 90m^d \leq 12lm^d + 90m^d + 12l + 76 \\ &= m^d(12l + 90) + 12l + 76 \end{aligned}$$

□

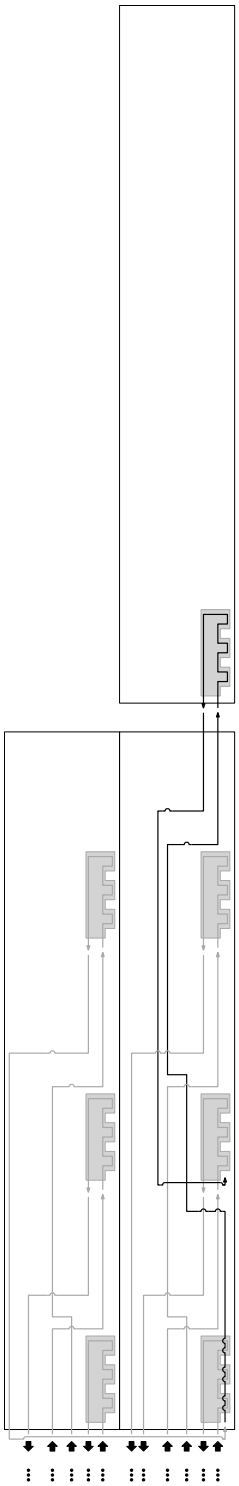
1.3 Filling in the gaps

...this means that the number of `Counter` unit rows $\mathcal{C}_\Delta$ is $m^d - \mathcal{C}_0$, where we have defined $\mathcal{C}_0$ as the starting value of the counter. To choose the best starting value, we find the value for $\mathcal{C}_\Delta$ that gets h as close to N without exceeding N. It follows from the equation $h = \mathcal{C}_\Delta(12l + 90) + 12l + 76$, that $\mathcal{C}_\Delta = \left\lfloor \frac{N-12l-76}{12l+90} \right\rfloor$. Thus, $\mathcal{C}_0 = m^d - \left\lfloor \frac{N-12l-76}{12l+90} \right\rfloor$. As a result of each digit requiring a width of 2 tiles, if k is odd, one additional tile column must be added. The number of filler tiles needed for the width is $k \mod 2$, and the number of filler tiles for the height is $N - 12l - 76 \mod 12l + 90$.

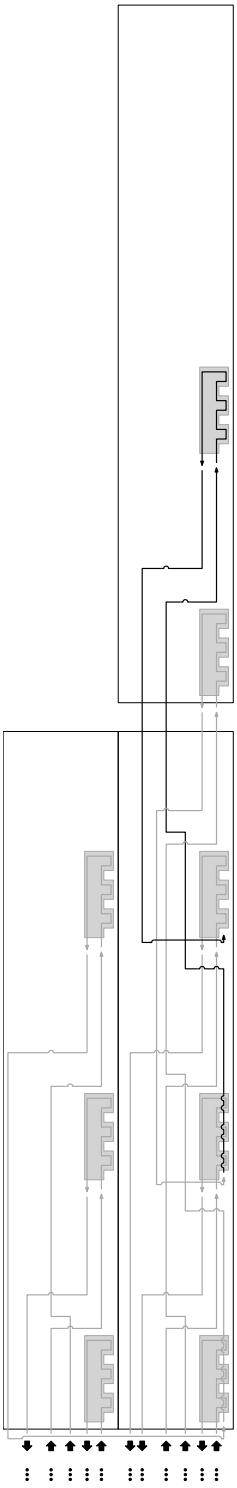
2 General counter



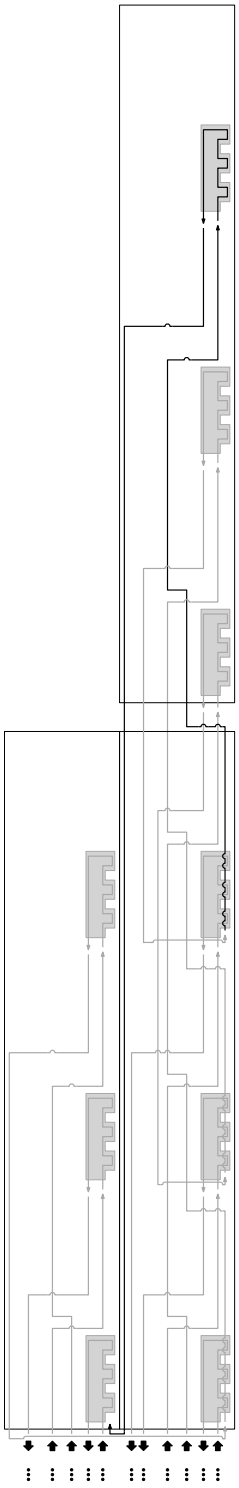
(a) A "clean" counter row, before any reading has started.



(b) Read digit 1 in the current row, write digit 1 in the next row.



(c) Read digit 2 in the current row, write digit 2 in the next row.



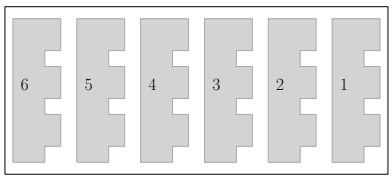
(d) Read digit 3 in the current row, write digit 3 in the next row.

Figure 1: This illustrates how a counter reads and writes a digit region, in a general sense. The counter starts in the rightmost digit region by reading the bottommost digit within that region. After reading digit 1 in the current row, the corresponding digit region in the next row be started in the next row. The counter writes the first digit in the next row, and then returns to the second digit in the current digit region. Once all the digits in the current digit region are read and written into the next row, the counter can then do one of the following: continue reading digits by moving on to the next digit region, cross back all the way to the right of the rectangle and start reading the next row, or halt.

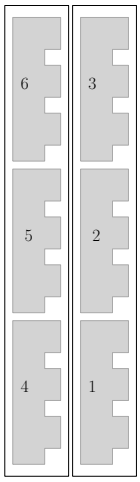
2.1 Digit region explanation (in progress)

Each logical row of the counter is made up of $\lceil \frac{d}{3} \rceil$ "digit regions". A digit region is a group of 1-3 digits, stacked vertically on top of one another. Within a digit region, the digits are sorted in order of significance, thus the top digit is the most signifcant digit, the middle digit is second most significant and the bottommost digit is the least signifcant.

The leftmost digit region is most signifcant and the rightmost is the least signifcant. The counter reads the least signifcant digit (1) in digit region 1, and continues in the current row until it detects the final digit, in the most signifcant digit region (MSR).



(a) Digits in a typical counter



(b) Digits in two digit regions, stacked vertically, minimizing the width.

Contrary to a typical counter, each counter row has an approximate height of 3 digits $\approx 12l$. The digits are stacked up to 3 before increasing the width.

2.2 Detecting the edges

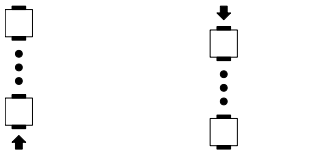
The counter must detect if a digit is in the MSR and if it's in the MSR, whether or not it is the most signifcant digit. To do this, all digits are encoded with two additional bits on the least signifcant end. If bit 0 is 1, the reader tiles know they could be reading the most signifcant digit (MSD) or in case 2, the second most signifcant digit. If bit 1 is 1, the digit currently being read is the MSD, otherwise the digit is digit 1 in case 2.

bit_1	bit_0	Meaning
0	0	digit is not in MSR
0	1	digit is in the MSR but is not the MSD
1	0	
1	1	digit is in the MSR and is MSD

2.3 Tile set

When describing a special case, i.e. "digit x – case y", whatever follows will only apply to the MSR (due to each case only affecting the MSR.)

2.3.1 Line Gadgets



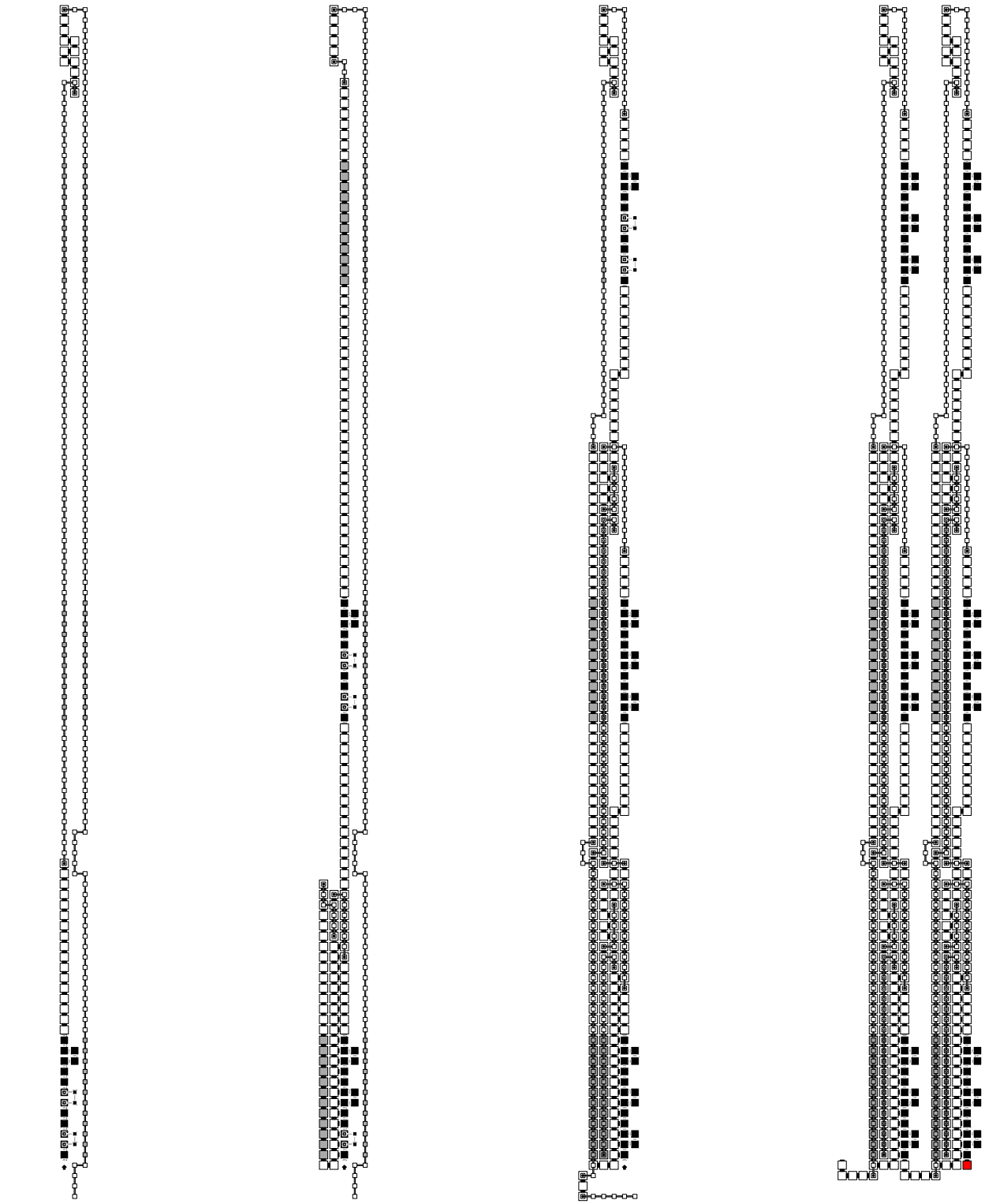
(a) `North_Line` (b) `South_Line`

Figure 3: `Line` gadgets

We will use the notation `NorthN_Line` and `SouthN_Line` where `N` corresponds to the length of a specific line gadget.

2.3.2 Initial Value (updated to assemble right to left like the other gadgets)

We begin by encoding $\mathcal{C}_0$ with the Seed unit. It has $\lceil \frac{d}{3} \rceil$ digit regions. Each digit region has three digits, except for the most significant digit region (MSR) which has $d \mod 3$ if $d \mod 3 \neq 0$, otherwise it has 3 digits.



(a) MSR case 1 (b) MSR case 2 (c) MSR case 3 (d) General digit regions

Figure 4: These figures show an example construction of the initial value, with all the possible MSR to the left. Of the three possible MSRs, of course only one would occur in a real assemble.

Note that we use i as the index of a digit in $\mathcal{C}_0$ and j as the index of a bit in a encoded digit.

- Create `Seed`($\langle$`Write`, 1, `seed`, 0, 0$\rangle$)

The idea here is to repeat these steps from $i = 0$ until i is the index of the first digit in the MSR. These steps build general non-MSR digit regions shown in Figure 4d.

- `Start`:
- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l-1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 1, `seed`, $i\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 12a.
  - Create `North_Line5`($\langle$`DigitTop`, 1, `seed`, $i\rangle$, $\langle$`DigitTopA`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 1, `seed`, $i\rangle$, $\langle$`DigitTopB`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopB`, 1, `seed`, $i\rangle$, $\langle$`ReturnPath`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3b.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 1, `seed`, $i\rangle$, $\langle$`NextRead`, 1, `seed`, $i\rangle$) (single-tile)
- $i \leftarrow i + 1$
- Create `Next_Read`($\langle$`NextRead`, 1, `seed`, $i-1\rangle$, $\langle$`SecondWarp`, 2, `seed`, $i\rangle$) (single-tile)
- Create `Second_Warp`($\langle$`SecondWarp`, 2, `seed`, $i\rangle$, $\langle$`PostWarp`, 2, `seed`, $i\rangle$)
- Create `Post_Warp`($\langle$`PostWarp`, 2, `seed`, $i\rangle$, $\langle$`Write`, 2, `seed`, $i, 0\rangle$) from the general gadget show in Figure 9b.
- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l-1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 2, `seed`, $i\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 12a.

  - Create `North_Line5`($\langle$`DigitTop`, 2, `seed`, $i\rangle$, $\langle$`DigitTopA`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 2, `seed`, $i\rangle$, $\langle$`DigitTopB`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopB`, 2, `seed`, $i\rangle$, $\langle$`ReturnPath`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3b.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 2, `seed`, $i\rangle$, $\langle$`NextRead`, 2, `seed`, $i\rangle$)
  from the gadget in Figure 14a.
- $i \leftarrow i + 1$
- Create `Next_Read`($\langle$`NextRead`, 2, `seed`, $i-1\rangle$, $\langle$`FirstWarp`, 3, `seed`, $i\rangle$)
  from the general gadget shown in Figure 16f.
- Create `First_Warp`($\langle$`FirstWarp`, 3, `seed`, $i\rangle$, $\langle$`WarpBridge`, 3, `seed`, $i\rangle$)
- Create `Warp_Bridge`($\langle$`WarpBridge`, 3, `seed`, $i\rangle$, $\langle$`SecondWarp`, 3, `seed`, $i\rangle$)
  from the general gadget shown in Figure 8a.
- Create `Second_Warp`($\langle$`SecondWarp`, 3, `seed`, $i\rangle$, $\langle$`PostWarp`, 3, `seed`, $i\rangle$)
- Create `Post_Warp`($\langle$`PostWarp`, 3, `seed`, $i\rangle$, $\langle$`Write`, 3, `seed`, $i, 0\rangle$)
  from the general gadget shown in Figure 9b.
- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`Write`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`Write`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`Write`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l-1$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 3, `seed`, $i\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 12a.
  - Create `North_Line5`($\langle$`DigitTop`, 3, `seed`, $i\rangle$, $\langle$`DigitTopA`, 3, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 3, `seed`, $i\rangle$, $\langle$`DigitTopB`, 3, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopB`, 3, `seed`, $i\rangle$, $\langle$`ReturnPath`, 3, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3b.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, `seed`, $i\rangle$, $\langle$`NextRead`, 3, `seed`, $i, \rangle$)
  from the gadget in Figure 15.
- $i \leftarrow i + 1$
- Create `Next_Read`($\langle$`NextRead`, 3, `seed`, $i-1\rangle$, $\langle$`Write`, 1, `seed`, $i\rangle$)
  from the general gadget shown in Figure 16e.
- if i is not an index in the MSR, go to `start`, else go to MSR.

MSR

Case 1 – if $d - i = 1$ to create the assembly shown in 4a.

- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10b.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10b.
  - if $0 \leqslant j \leqslant l$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l - 1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 1, `seed`, $i\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 12d
  - Create `North_Line4`l($\langle$`DigitTop`, 1, `seed`, $i\rangle$, $\langle$`DigitTopA`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3a.
  - Create `North_Line4`($\langle$`DigitTopA`, 1, `seed`, $i\rangle$, $\langle$`DigitTopB`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopB`, 1, `seed`, $i\rangle$, $\langle$`DigitTopC`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopC`, 1, `seed`, $i\rangle$, $\langle$`DigitTopD`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3b.
  - Create `South_Line30`($\langle$`DigitTopD`, 1, `seed`, $i\rangle$, $\langle$`DigitTopE`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3b.
  - Create `South_Line4`l($\langle$`DigitTopE`, 1, `seed`, $i\rangle$, $\langle$`DigitTopF`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3b.
  - Create `South_Line14`($\langle$`DigitTopF`, 1, `seed`, $i\rangle$, $\langle$`DigitTopG`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3b.
  - Create `South_Line17`($\langle$`DigitTopG`, 1, `seed`, $i\rangle$, $\langle$`ReturnPath`, 1, `seed`, $i\rangle$) from the micro-gadget shown in Figure 3b.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 1, `seed`, $i\rangle$, $\langle$`NextRead`, 1, `seed`, $i\rangle$) from the general gadget shown in Figure 14d
- Create `Next_Read`($\langle$`NextRead`, 1, `seed`, $i\rangle$, $\langle$`Cross_Next_Row`, `increment`$\rangle$) from the micro-gadget shown in Figure 16c.

Case 2 – if $d - i = 2$ to create the assembly shown in 4b.

- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10b.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a.
  - if $1 < j < l - 1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l - 1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 2, `seed`, $i\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.

- `Digit_Top`: the following statements create the gadget shown in Figure 12b
  - Create `Topper`($\langle$`DigitTop`, 1, `seed`, $i\rangle$, $\langle$`DigitTopA`, 1, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 11b
  - Create `South_Line4`l($\langle$`DigitTopA`, 1, op, `msr`$\rangle$, $\langle$`ReturnPath`, 1, op, `msr`$\rangle$)
    from the micro-gadget shown in Figure 3b
- Create `Return_From_Digit`($\langle$`ReturnPath`, 1, `seed`, $i\rangle$, $\langle$`NextRead`, 1, `seed`, $i\rangle$)
  (single tile)
- $i \leftarrow i + 1$
- Create `Next_Read`($\langle$`NextRead`, 1, `seed`, $i-1\rangle$, $\langle$`SecondWarp`, 2, `seed`, $i\rangle$)
  (single tile)
- Create `Second_Warp`($\langle$`SecondWarp`, 2, `seed`, $i\rangle$, $\langle$`PostWarp`, 2, `seed`, $i\rangle$)
  (single tile)
- Create `Post_Warp`($\langle$`PostWarp`, 2, `seed`, $i\rangle$, $\langle$`Write`, 2, `seed`, $i, 0\rangle$)
  from the general gadget show in Figure 9e.
- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $1 < j < l-1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l-1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 2, `seed`, $i\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 12c
  - Create `North_Line4`l($\langle$`DigitTop`, 2, `seed`, $i\rangle$, $\langle$`DigitTopA`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 2, `seed`, $i\rangle$, $\langle$`DigitTopB`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 11c.
  - Create `South_Line4`l($\langle$`DigitTopB`, 2, `seed`, $i\rangle$, $\langle$`DigitTopC`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3b.
  - Create `South_Line30`($\langle$`DigitTopC`, 2, `seed`, $i\rangle$, $\langle$`ReturnPath`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3b.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 2, `seed`, $i\rangle$, $\langle$`NextRead`, 2, `seed`, $i\rangle$)
  from the micro-gadget shown in Figure 14d.
- Create `Next_Read`($\langle$`NextRead`, 2, `seed`$\rangle$, $\langle$`Cross_Next_Row`, `increment`$\rangle$)
  from the micro-gadget shown in Figure 16c.

Case 3 – if $d - i = 3$ to create the assembly shown in 4c.

- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.

  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`Write`, 1, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l-1$: create `Counter_Write`($\langle$`Write`, 1, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 1, `seed`, $i\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.

- `Digit_Top`: the following statements create the gadget shown in Figure 12a.
  - Create `North_Line5`($\langle$`DigitTop`, 1, `seed`, $i\rangle$, $\langle$`DigitTopA`, 1, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 1, `seed`, $i\rangle$, $\langle$`DigitTopB`, 1, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopB`, 1, `seed`, $i\rangle$, $\langle$`ReturnPath`, 1, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3b.

- $i \leftarrow i + 1$

- Create `Return_From_Digit`($\langle$`ReturnPath`, 1, `seed`, $i-1\rangle$, $\langle$`SecondWarp`, 2, `seed`, $i\rangle$)
  (single tile)

- Create `Second_Warp`($\langle$`SecondWarp`, 2, `seed`, $i\rangle$, $\langle$`PostWarp`, 2, `seed`, $i\rangle$)

- Create `Post_Warp`($\langle$`PostWarp`, 2, `seed`, $i\rangle$, $\langle$`Write`, 2, `seed`, $i, 0\rangle$)
  from the general gadget show in Figure 9b.

- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`Write`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l-1$: create `Counter_Write`($\langle$`Write`, 2, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 2, `seed`, $i\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.

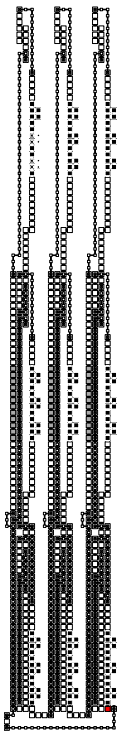
- `Digit_Top`: the following statements create the gadget shown in Figure 12a.
  - Create `North_Line5`($\langle$`DigitTop`, 2, `seed`, $i\rangle$, $\langle$`DigitTopA`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 2, `seed`, $i\rangle$, $\langle$`DigitTopB`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopB`, 2, `seed`, $i\rangle$, $\langle$`ReturnPath`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3b.

- Create `Return_From_Digit`($\langle$`ReturnPath`, 2, `seed`, $i\rangle$, $\langle$`NextRead`, 2, `seed`, $i\rangle$)
  from the gadget in Figure 14a.

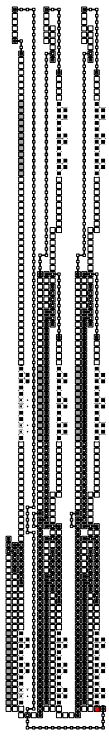
- $i \leftarrow i + 1$

- Create `Next_Read`($\langle$`NextRead`, 2, `seed`, $i-1\rangle$, $\langle$`FirstWarp`, 3, `seed`, $i\rangle$)
  from the general gadget shown in Figure 16f.

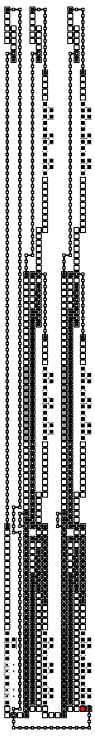
- Create `First_Warp`($\langle$`FirstWarp`, 3, `seed`, $i\rangle$, $\langle$`WarpBridge`, 3, `seed`, $i\rangle$)
- Create `Warp_Bridge`($\langle$`WarpBridge`, 3, `seed`, $i\rangle$, $\langle$`SecondWarp`, 3, `seed`, $i\rangle$)
  from the general gadget shown in Figure 8a.
- Create `Second_Warp`($\langle$`SecondWarp`, 3, `seed`, $i\rangle$, $\langle$`PostWarp`, 3, `seed`, $i\rangle$)
- Create `Post_Warp`($\langle$`PostWarp`, 3, `seed`, $i\rangle$, $\langle$`Write`, 3, `seed`, $i, 0\rangle$)
  from the general gadget shown in Figure 9b.
- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`Write`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $j = 1$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`Write`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $1 < j < l-1$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`Write`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l-1$: create `Counter_Write`($\langle$`Write`, 3, `seed`, $i, j\rangle$, $\langle$`DigitTop`, 3, `seed`, $i\rangle$)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 12a.
  - Create `North_Line5`($\langle$`DigitTop`, 3, `seed`, $i\rangle$, $\langle$`DigitTopA`, 3, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 3, `seed`, $i\rangle$, $\langle$`DigitTopB`, 3, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopB`, 3, `seed`, $i\rangle$, $\langle$`ReturnPath`, 3, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3b.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, `seed`, $i\rangle$, $\langle$`NextRead`, 3, `increment`, `msr`, `msd`$\rangle$)
  from the gadget in Figure 15.



(a) Initial value case 3



(b) Initial value case 2



(c) Initial value case 1

2.4 Counter Unit

2.4.1 Digit readers

- For each $i = 1, 2, 3$, $j = 0, \ldots, l-3$, $u \in \{0,1\}^j$, and $op \in \{\texttt{increment}, \texttt{copy}\}$:
  - if $j = 0$: create `Counter_Read`($\langle \texttt{Read}, i, \lambda, op \rangle$, $\langle \texttt{Read}, i, 0, op \rangle$, $\langle \texttt{Read}, i, 1, op \rangle$) from the general gadget in Figure 6.
  - else: create `Counter_Read`($\langle \texttt{Read}, i, u, op \rangle$, $\langle \texttt{Read}, i, 0u, op \rangle$, $\langle \texttt{Read}, i, 1u, op \rangle$) from the general gadget in Figure 6.
- For each $i = 1, 2, 3$ and each $u \in \{0,1\}^{l-2}$:
  - Create `Counter_Read`($\langle \texttt{Read}, i, u, \texttt{copy} \rangle$, $\langle \texttt{PreWarp}, i, 0u, \texttt{copy} \rangle$, $\langle \texttt{PreWarp}, i, 1u, \texttt{copy} \rangle$) from the general gadget in Figure 6.

Since the counter must only increment the current value if the result will be less than m, the `Counter_Read` gadgets that have both an `increment` signal and input size of $l-2$ must first right shift the bits 2 spots, and then for each possible value after reading one more bit, check whether that value is less than $m-1$. Basically, if the next bit read is a 0, we check if the current value + 1 is less than m. If the next bit read is a 1, we check if current value + $2m+1$ is less than m. For both cases, if the counter can increment the current value, then the `Counter_Read` gadgets output the incremented value to the `Pre_Warp` gadgets and propagate a `copy` signal. Otherwise, if the counter is unable to increment the value, it outputs a digits with all zeroes and propagates the `increment` signal to the next digit.

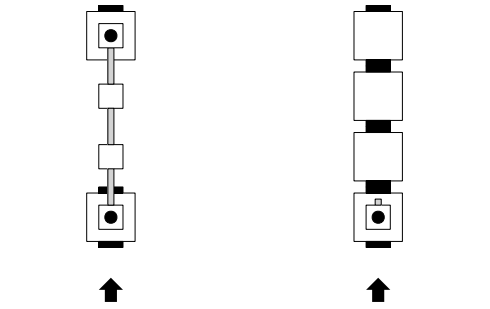
- For each $i = 1, 2, 3$ and each $u \in \{0,1\}^{l-2}$:

  Let $guess0 = 0u >> 2$, $guess1 = 1u >> 2$

if $dec(guess0) + 1 < m - 1$
then $R0 = bin(dec(guess0) + 1) + u[1] + u[0], \texttt{copy}$
else $R0 = \{0\}^l, \texttt{increment}$

if $dec(guess1) + 1 < m - 1$
then $R1 = bin(dec(guess1) + 1) + u[1] + u[0], \texttt{copy}$
else $R1 = \{0\}^l, \texttt{increment}$

– Create `Counter_Read`($\langle \texttt{Read}, i, u, \texttt{increment} \rangle, \langle \texttt{PreWarp}, i, R0 \rangle, \langle \texttt{PreWarp}, i, R1 \rangle$)
from the general gadget in Figure 6.



(a) `Counter_Read_0` (b) `Counter_Read_1`

Figure 6: The `Counter_Read` gadgets

2.4.2 Warping

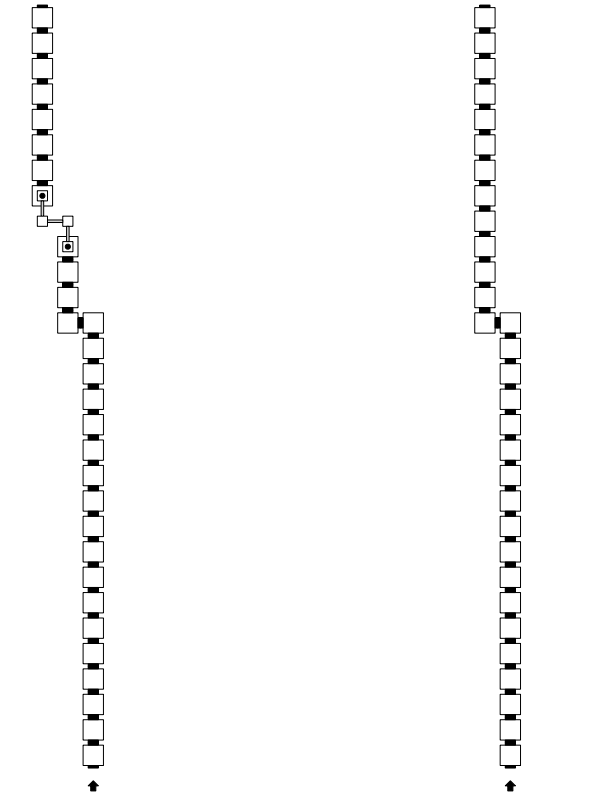
We now explain the `Warp_Unit`. A warp unit generally consists of the following 5 gadgets: `Pre_Warp`, `First_Warp`, `Warp_Bridge`, `Second_Warp`, `Post_Warp`. The job of these 5 gadgets is to transport the value read by the `Counter_Read` all the way to the digit region in the next row, so that the `Counter_Write` gadgets can write the next value in the correct locations. The `First_Warp` and `Second_Warp` gadgets are single tile gadgets that have north and south glues with identical labels. This allows the gadgets to continuously assemble until stopped by earlier parts of the assembly. These single tile gadgets also have one additional glue that will allow the next piece in the warp unit to assemble, however the assembly will also block this side of the tile all the way until the gadget can no longer continue assembling in the north direction.

- `Pre_Warp`: These gadgets use the bits read from the `Counter_Read` gadgets to translate them into a signal used to tell the counter whether to begin reading another digit in the current row, or cut across the rectangle and begin reading the first digit in the next row. This signal is used from the `Pre_Warp` gadgets through the `Digit_Top` gadgets are attached after writing the current digit.

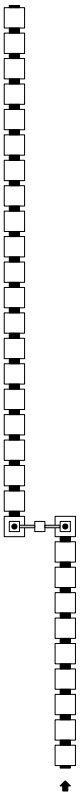
  For each $i = 1, 2, 3, u \in \{0, 1\}^l$, and each $op \in \{\texttt{increment}, \texttt{copy}\}$:

  – if u ends with 00: create `Pre_Warp`($\langle \texttt{PreWarp}, i, u, op \rangle, \langle \texttt{FirstWarp}, i, u, op \rangle$)
  from the general gadget in Figure 7a.
  – if u ends with 01: create `Pre_Warp`($\langle \texttt{PreWarp}, i, u, op \rangle, \langle \texttt{FirstWarp}, i, u, op, \texttt{msr} \rangle$)
  from the general gadget in Figure 7c.
  – if u ends with 11 and $dec(u >> 2) >= m$:
  create `Pre_Warp`($\langle \texttt{PreWarp}, i, u, op \rangle, \langle \texttt{FirstWarp}, i, u, op, \texttt{halt}, \rangle$) from the general gadget in Figure 7b if $i = 1$ (case 1), or Figure 7d if $i = 2$ (case 2), or Figure 7a if $i = 3$ (case 3).

– if u ends with 11: create $\texttt{Pre\_Warp}(\langle \texttt{PreWarp}, i, u, op\rangle, \langle \texttt{FirstWarp}, i, u, op, \texttt{msr}, \texttt{msd}\rangle)$ from the general gadget in Figure 7b if $i = 1$ (case 1), or Figure 7d if $i = 2$ (case 2), or Figure 7a if $i = 3$ (case 3).



(a) General (b) Digit 1 – Case 1



(c) Digit 1 – Case 2

(d) Digit 2 – Case 2

Figure 7: The `Pre_Warp` gadgets

- `First_Warp`: A `First_Warp` connects to a `Warp_Bridge` gadget in all cases except when it's assembling in the MSR and it is digit 1 in case 1 or 2, in which the `First_Warp` gadget attaches directly to a `Post_Warp` gadget.

  For each $u \in \{0,1\}^l$, and each $op \in \{\texttt{increment}, \texttt{copy}\}$:

  - For each $i = 1, 2, 3$: create
    $\texttt{First\_Warp}(\ \langle \texttt{FirstWarp}, i, u, op\rangle\ , \langle \texttt{FirstWarp}, i, u, op\rangle\ , \langle \texttt{WarpBridge}, i, u, op\rangle\ )$

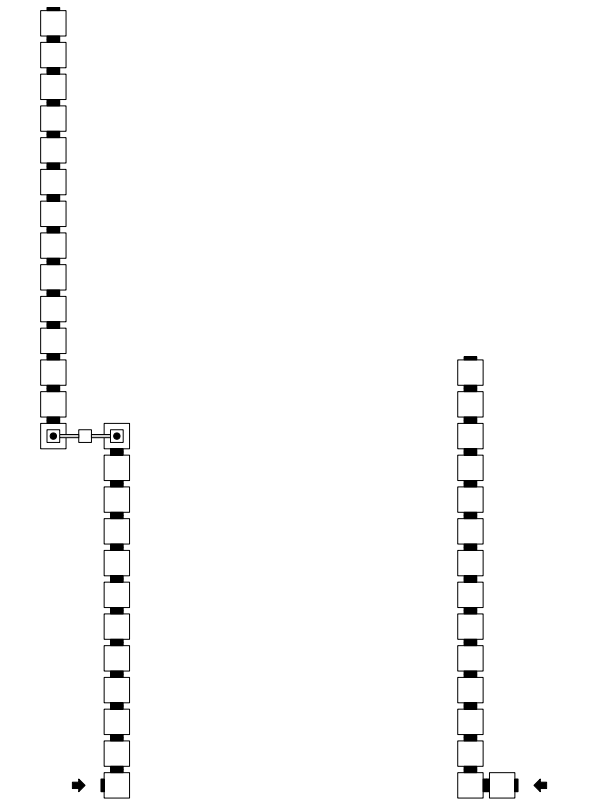
  - Create $\texttt{First\_Warp}(\ \langle \texttt{FirstWarp}, 1, u, op, \texttt{msr}\rangle\ ,$
    $\langle \texttt{FirstWarp}, 1, u, op, \texttt{msr}\rangle\ ,$
    $\langle \texttt{PostWarp}, 1, u, op, \texttt{msr}\rangle\ )$

  - Create $\texttt{First\_Warp}(\ \langle \texttt{FirstWarp}, 1, u, op, \texttt{msr}, \texttt{msd}\rangle\ ,$
    $\langle \texttt{FirstWarp}, 1, u, op, \texttt{msr}, \texttt{msd}\rangle\ ,$
    $\langle \texttt{PostWarp}, 1, u, op, \texttt{msr}, \texttt{msd}\rangle\ )$

  - Create $\texttt{First\_Warp}(\ \langle \texttt{FirstWarp}, 2, u, op, \texttt{msr}, \texttt{msd}\rangle\ ,$
    $\langle \texttt{FirstWarp}, 2, u, op, \texttt{msr}, \texttt{msd}\rangle\ ,$
    $\langle \texttt{WarpBridge}, 2, u, op, \texttt{msr}, \texttt{msd}\rangle\ )$

– Create `First_Warp`($\langle$`FirstWarp`, 3, u, op, `msr`, `msd`$\rangle$,
$\langle$`FirstWarp`, 3, u, op, `msr`, `msd`$\rangle$,
$\langle$`WarpBridge`, 3, u, op, `msr`, `msd`$\rangle$)

- `Warp_Bridge`: a `Warp_Bridge` gadget binds the last tile of the `First_Warp` gadgets to the first tile of the `Second_Warp` gadgets. For digit 1 in cases 1 and 2, the `Warp_Bridge` is omitted from the `Warp_Unit`.
  For each $u \in \{0,1\}^l$, and each $op \in \{$`increment`, `copy`$\}$:
  – For each $i = 1, 2, 3$: create `Warp_Bridge`($\langle$`WarpBridge`, i, u, $op\rangle$, $\langle$`SecondWarp`, i, u, $op\rangle$) from the general gadget in Figure 8a.
  – Create `Warp_Bridge`($\langle$`WarpBridge`, 2, u, op, `msr`, `msd`$\rangle$, $\langle$`SecondWarp`, 2, u, op, `msr`, `msd`$\rangle$) from the general gadget in Figure 8b.
  – Create `Warp_Bridge`($\langle$`WarpBridge`, 3, u, op, `msr`, `msd`$\rangle$, $\langle$`SecondWarp`, 3, u, op, `msr`, `msd`$\rangle$) from the general gadget in Figure 8a.



(a) General (b) Digit 2 – Case 2

Figure 8: The `Warp_Bridge` gadgets.

- `Second_Warp`:

  For each $u \in \{0,1\}^l$, and each $op \in \{\texttt{increment}, \texttt{copy}\}$:

  - For each $i = 1, 2, 3$: Create `Second_Warp`($\langle \texttt{SecondWarp}, i, u, op \rangle$,
    $\langle \texttt{SecondWarp}, i, u, op \rangle$,
    $\langle \texttt{PostWarp}, i, u, op \rangle$)

  - Create `Second_Warp`($\langle \texttt{SecondWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{SecondWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{PostWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$)

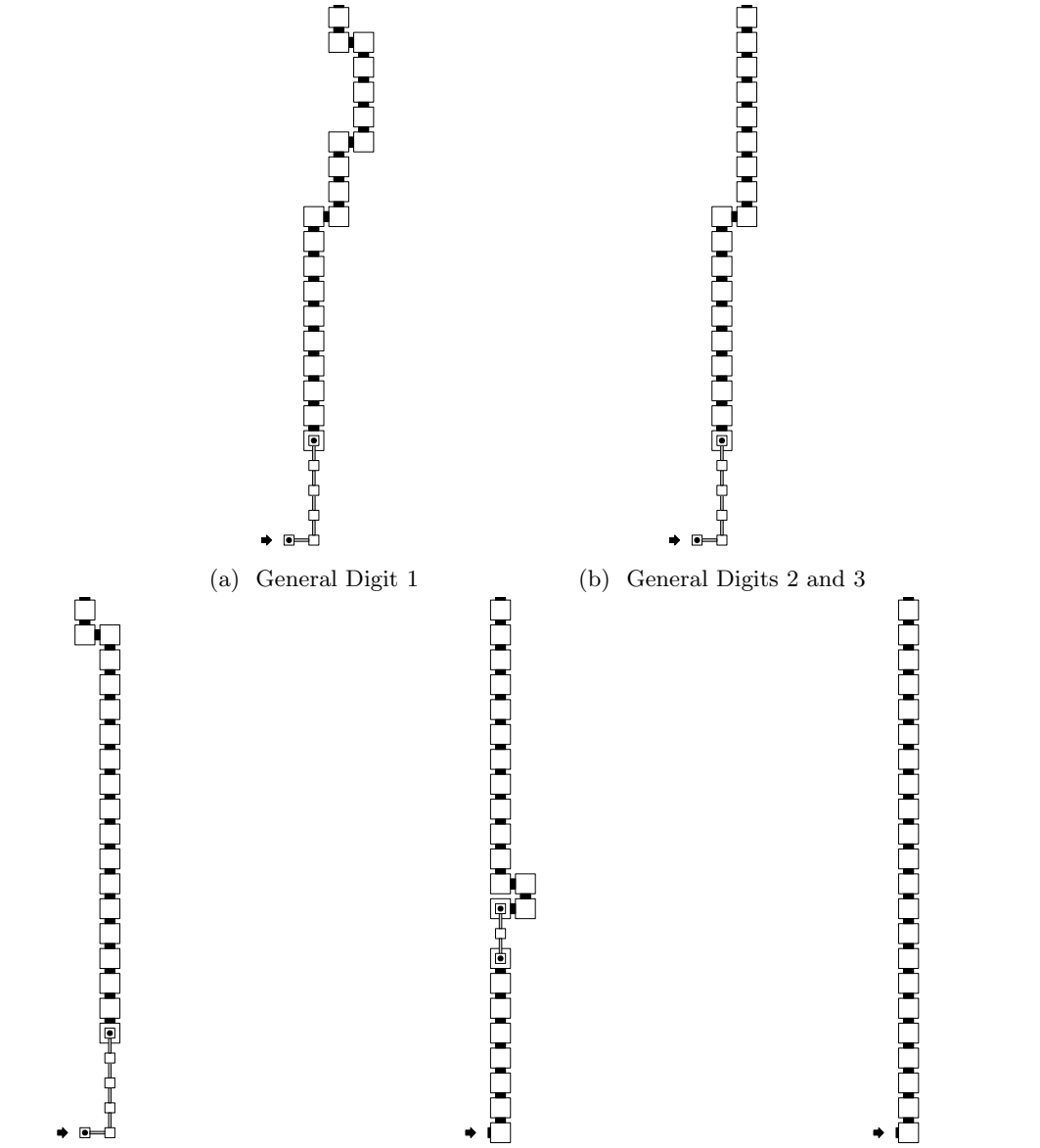
  - Create `Second_Warp`($\langle \texttt{SecondWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{SecondWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{PostWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$)

- `Post_Warp`:

  - For each $i = 1, 2, 3$: create
    `Post_Warp`($\langle \texttt{PostWarp}, i, u, op \rangle$, $\langle \texttt{Write}, i, u, op \rangle$)
    from the general gadget shown in Figure 9a if $i = 1$, or Figure 9b if $i = 2$ or $i = 3$.

  - Create `Post_Warp`($\langle \texttt{PostWarp}, 1, u, op, \texttt{msr} \rangle$, $\langle \texttt{Write}, 1, u, op, \texttt{msr} \rangle$)
    from the general gadget in Figure 9d.

  - For each $i = 1, 2, 3$: create
    `Post_Warp`($\langle \texttt{PostWarp}, i, u, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{Write}, i, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the general gadget shown in Figure 9c if $i = 1$, or Figure 9e if $i = 2$, or Figure 9b if $i = 3$.



(a) General Digit 1 (b) General Digits 2 and 3

(c) Digit 1 – Case 1 (d) Digit 1 – Case 2 (e) Digit 2 – Case 2

Figure 9: The `Post_Warp` gadgets

2.4.3 Counter write

- For each $i = 1, 2, 3$, $j = l - 1, \ldots, 1$, $u \in \{0,1\}^j$, and each $op \in \{\texttt{increment}, \texttt{copy}\}$:
  - Create `Counter_Write`($\langle \texttt{Write}, i, u0, op \rangle$, $\langle \texttt{Write}, i, u, op \rangle$)
    from the general gadget in Figure 10a
  - Create `Counter_Write`($\langle \texttt{Write}, i, u1, op \rangle$, $\langle \texttt{Write}, i, u, op \rangle$)
    from the general gadget in Figure 10b
  - Create `Counter_Write`($\langle \texttt{Write}, 1, u0, op, \texttt{msr} \rangle$, $\langle \texttt{Write}, 1, u, op, \texttt{msr} \rangle$)
    from the general gadget in Figure 10a

- Create `Counter_Write`($\langle$`Write`, 1, $u1$, op, `msr`$\rangle$, $\langle$`Write`, 1, u, op, `msr`$\rangle$)
  from the general gadget in Figure 10b
- Create `Counter_Write`($\langle$`Write`, i, $u0$, op, `msr`, `msd`$\rangle$, $\langle$`Write`, i, u, op, `msr`, `msd`$\rangle$)
  from the general gadget in Figure 10a
- Create `Counter_Write`($\langle$`Write`, i, $u1$, op, `msr`, `msd`$\rangle$, $\langle$`Write`, i, u, op, `msr`, `msd`$\rangle$)
  from the general gadget in Figure 10b

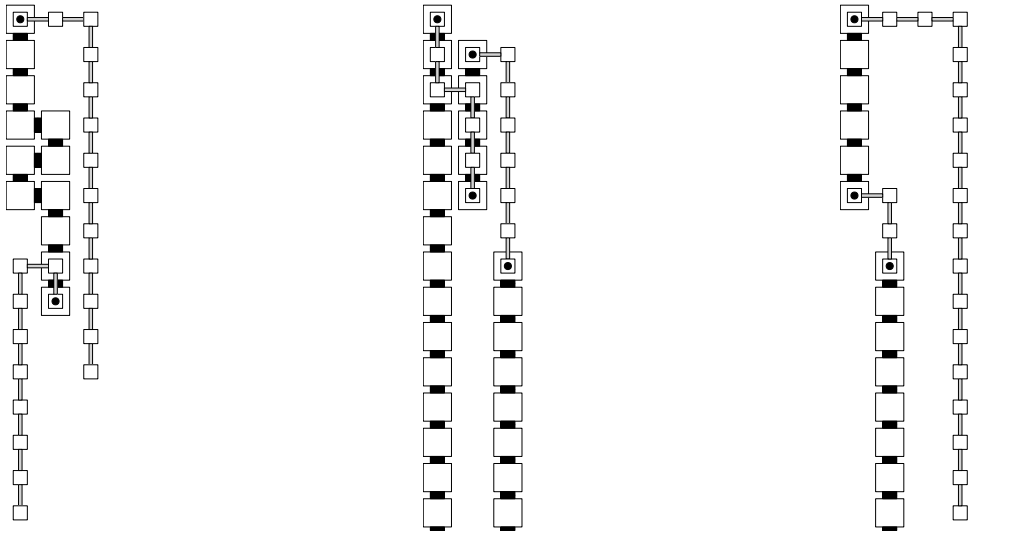
- For each $i = 1, 2, 3$ and each $op \in \{$`increment`, `copy`$\}$:
  - Create `Counter_Write`($\langle$`Write`, i, 0, $op\rangle$, $\langle$`DigitTop`, i, $op\rangle$)
    from the general gadget in Figure 10a
  - Create `Counter_Write`($\langle$`Write`, i, 1, $op\rangle$, $\langle$`DigitTop`, i, $op\rangle$)
    from the general gadget in Figure 10b
  - Create `Counter_Write`($\langle$`Write`, 1, 0, op, `msr`$\rangle$, $\langle$`DigitTop`, 1, op, `msr`$\rangle$)
    from the general gadget in Figure 10a
  - Create `Counter_Write`($\langle$`Write`, 1, 1, op, `msr`$\rangle$, $\langle$`DigitTop`, 1, op, `msr`$\rangle$)
    from the general gadget in Figure 10b
  - Create `Counter_Write`($\langle$`Write`, i, 0, op, `msr`, `msd`$\rangle$, $\langle$`DigitTop`, i, op, `msr`, `msd`$\rangle$)
    from the general gadget in Figure 10a
  - Create `Counter_Write`($\langle$`Write`, i, 1, op, `msr`, `msd`$\rangle$, $\langle$`DigitTop`, i, op, `msr`, `msd`$\rangle$)
    from the general gadget in Figure 10b

(a) `Counter_Write_0` (b) `Counter_Write_1`

Figure 10: The `Counter_Write` gadgets

2.4.4 Digit tops

The `Digit_Top` gadgets have special geometry designed so that `First_Warp` and `Second_Warp` tiles are allowed to "wake up", and complete their warping journey. Each digit has some type of `Digit_Top` gadget, however, depending on the digit region and index of a specific digit, the exact digit top will differ.



(a) General topper (b) Case 1 – topper (c) Case 2 – topper

Figure 11: Topper micro-gadgets

For each $op \in \{\texttt{increment}, \texttt{copy}\}$

- Digit 1 (general): the following statements create the gadget shown in Figure 12a
  - Create `North_Line5`($\langle$`DigitTop`$, 1, op\rangle$, $\langle$`DigitTopA`$, 1, op\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`$, 1, op\rangle$, $\langle$`DigitTopB`$, 1, op\rangle$)
    from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l($\langle$`DigitTopB`$, 1, op\rangle$, $\langle$`ReturnPath`$, 1, op\rangle$)
    from the micro-gadget shown in Figure 3b.

- Digit 1 (MSR): the following statements create the gadget shown in Figure 12b
  - Create `Topper`($\langle$`DigitTop`$, 1, op,$ `msr`$\rangle$, $\langle$`DigitTopA`$, 1, op,$ `msr`$\rangle$)
    from the micro-gadget shown in Figure 11b.
  - Create `South_Line4`l($\langle$`DigitTopA`$, 1, op,$ `msr`$\rangle$, $\langle$`ReturnPath`$, 1, op,$ `msr`$\rangle$)
    from the micro-gadget shown in Figure 3b.

- Digit 1 (MSD): the following statements create the gadget shown in Figure 12d
  - Create `North_Line4`l($\langle$`DigitTop`$, 1, op,$ `msr`$,$ `msd`$\rangle$, $\langle$`DigitTopA`$, 1, op,$ `msr`$,$ `msd`$\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `North_Line4`($\langle$`DigitTopA`$, 1, op,$ `msr`$,$ `msd`$\rangle$, $\langle$`DigitTopB`$, 1, op,$ `msr`$,$ `msd`$\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopB`$, 1, op,$ `msr`$,$ `msd`$\rangle$, $\langle$`DigitTopC`$, 1, op,$ `msr`$,$ `msd`$\rangle$)
    from the micro-gadget shown in Figure 11a.

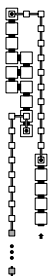
- Create South_Line4l(⟨DigitTopC, 1, op, msr, msd⟩ , ⟨DigitTopD, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3b.
- Create South_Line30(⟨DigitTopD, 1, op, msr, msd⟩ , ⟨DigitTopE, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3b.
- Create South_Line4l(⟨DigitTopE, 1, op, msr, msd⟩ , ⟨DigitTopF, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3b.
- Create South_Line14(⟨DigitTopF, 1, op, msr, msd⟩ , ⟨DigitTopG, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3b.
- Create South_Line17(⟨DigitTopG, 1, op, msr, msd⟩ , ⟨ReturnPath, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3b.

- Digit 2 (general): the following statements create the gadget shown in Figure 12a
  - Create North_Line5(⟨DigitTop, 2, op⟩ , ⟨DigitTopA2, op⟩)
    from the micro-gadget shown in Figure 3a.
  - Create Topper(⟨DigitTopA2, op⟩ , ⟨DigitTopB2, op⟩)
    from the micro-gadget shown in Figure 11a.
  - Create South_Line4l(⟨DigitTopB2, op⟩ , ⟨ReturnPath, 2, op⟩)
    from the micro-gadget shown in Figure 3b.

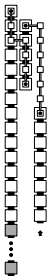
- Digit 2 (MSD): the following statements create the gadget shown in Figure 12c
  - Create North_Line4l(⟨DigitTop, 2, op, msr, msd⟩ , ⟨DigitTopA, 2, op, msr, msd⟩)
    from the micro-gadget shown in Figure 3a.
  - Create Topper(⟨DigitTopA, 2, op, msr, msd⟩ , ⟨DigitTopB, 2, op, msr, msd⟩)
    from the micro-gadget shown in Figure 11c.
  - Create South_Line4l(⟨DigitTopB, 2, op, msr, msd⟩ , ⟨DigitTopC, 2, op, msr, msd⟩)
    from the micro-gadget shown in Figure 3b.
  - Create South_Line30(⟨DigitTopC, 2, op, msr, msd⟩ , ⟨ReturnPath, 2, op, msr, msd⟩)
    from the micro-gadget shown in Figure 3b.

- Digit 3 (general): the following statements create the gadget from Figure 12a
  - Create North_Line5(⟨DigitTop, 3, op⟩ , ⟨DigitTopA, 3, op⟩)
    from the micro-gadget shown in Figure 3a.
  - Create Topper(⟨DigitTopA, 3, op⟩ , ⟨DigitTopB, 3, op⟩)
    from the micro-gadget shown in Figure 11a.
  - Create South_Line4l(⟨DigitTopB, 3, op⟩ , ⟨ReturnPath, 3, op⟩)
    from the micro-gadget shown in Figure 3b.

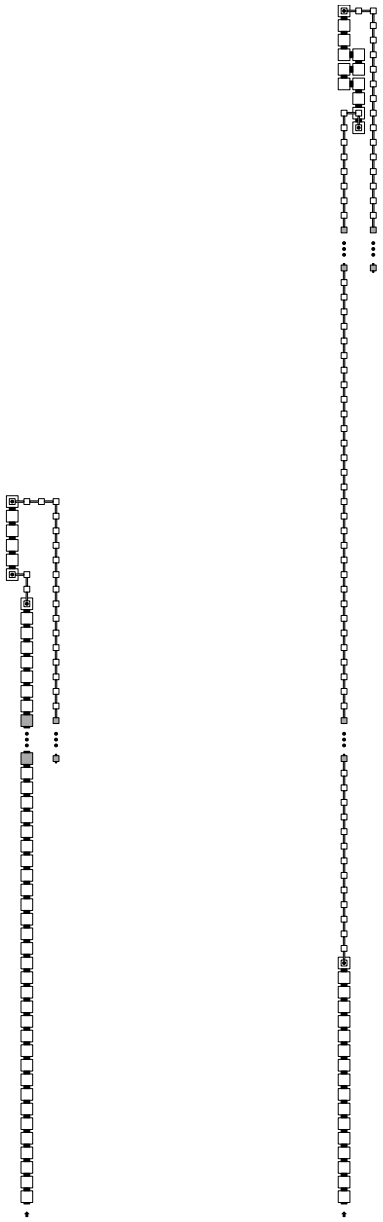
- Digit 3 (MSD): the following statements create the gadget from Figure 12a
  - Create `North_Line5`(⟨`DigitTop`, 3, *op*, `msr`, `msd`⟩ , ⟨`DigitTopA`, 3, *op*, `msr`, `msd`⟩) from the micro-gadget shown in Figure 3a.
  - Create `Topper`(⟨`DigitTopA`, 3, *op*, `msr`, `msd`⟩ , ⟨`DigitTopB`, 3, *op*, `msr`, `msd`⟩) from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`*l*(⟨`DigitTopB`, 3, *op*, `msr`, `msd`⟩ , ⟨`ReturnPath`, 3, *op*, `msr`, `msd`⟩) from the micro-gadget shown in Figure 3b.



(a) General

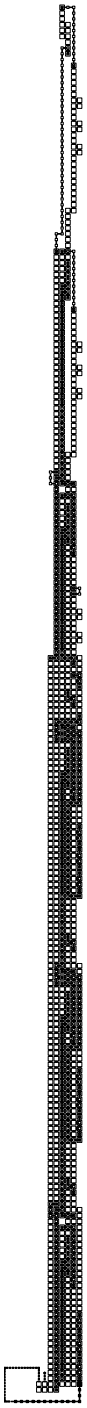


(b) Digit 1 – Case 2



(c) Digit 2 – Case 2 (d) Digit 1 – Case 1

Figure 12: `Digit_Top` gadgets

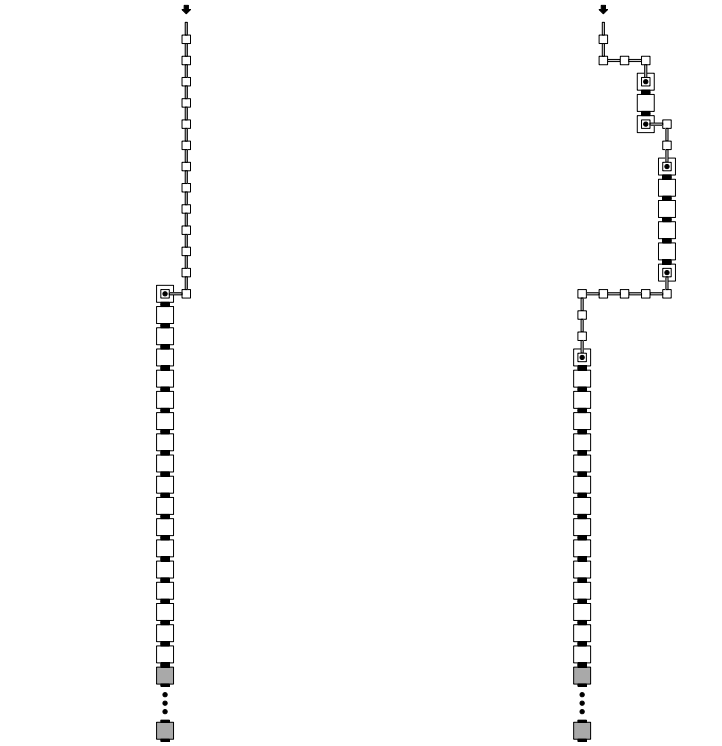


(a) Crosser overview

2.4.5 Return paths

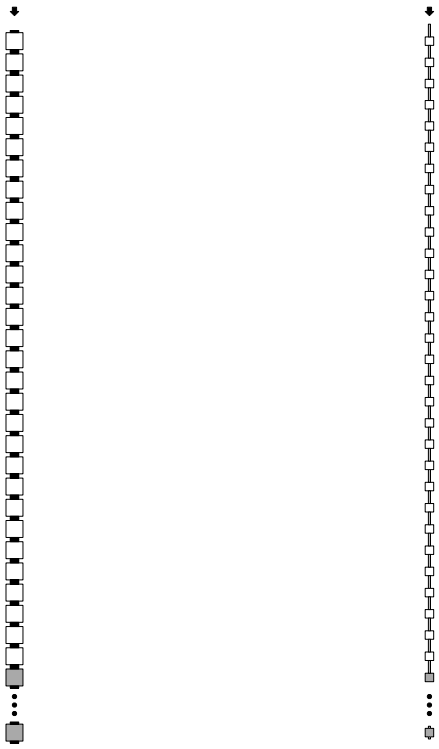
In this section, we explain the gadgets used after a digit and its `Digit_Top` gadget have assembled. These are the return paths, the purpose of these gadgets is to route the counter to the next place it needs to be, which could be the next digit, a new row, etc.

In general, a `Return_Path` gadget is comprised of two micro-gadgets, the first being a `Return_From_Digit` micro-gadget, and the second being a `Next_Read` micro-gadget.



(a) General and initial value digit 2

(b) General digit 1



(c) Digit 1 - case 2 (d) MSD - Digit 1 and 2

Figure 14: The `Return_From_Digit` gadgets for digits 1 and 2

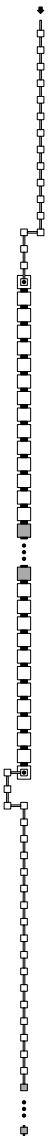


Figure 15: The `Return_From_Digit` gadget for digit 3.

The first part of the return paths is the `Return_From_Digit` micro-gadget.
For each $op \in \{\texttt{increment}, \texttt{copy}\}$:

- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 1, op \rangle$, $\langle \texttt{NextRead}, 1, op \rangle$)
  from the micro-gadget shown in Figure 14b.
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 1, op, \texttt{msr} \rangle$, $\langle \texttt{NextRead}, 1, op, \texttt{msr} \rangle$)
  from the micro-gadget shown in Figure 14c
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 1, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{NextRead}, 1, op, \texttt{msr}, \texttt{msd} \rangle$)
  from the micro-gadget shown in Figure 14d.
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 2, op \rangle$, $\langle \texttt{NextRead}, 2, op \rangle$)
  from the micro-gadget shown in Figure 14a.

- Create `Return_From_Digit`($\langle$`ReturnPath`, 2, *op*, `msr`, `msd`$\rangle$, $\langle$`NextRead`, 2, *op*, `msr`, `msd`$\rangle$)
  from the micro-gadget shown in Figure 14d.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, *op*$\rangle$, $\langle$`NextRead`, 3, *op*$\rangle$)
  from the micro-gadget shown in Figure 15.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, *op*, `msr`, `msd`$\rangle$, $\langle$`NextRead`, 3, *op*, `msr`, `msd`$\rangle$)
  from the micro-gadget shown in Figure 15.

The second part of the return gadgets is the `Next_Read` micro-gadget. These gadgets output a blank `Counter_Write` signal if the counter should read the preceding digit in the current row.

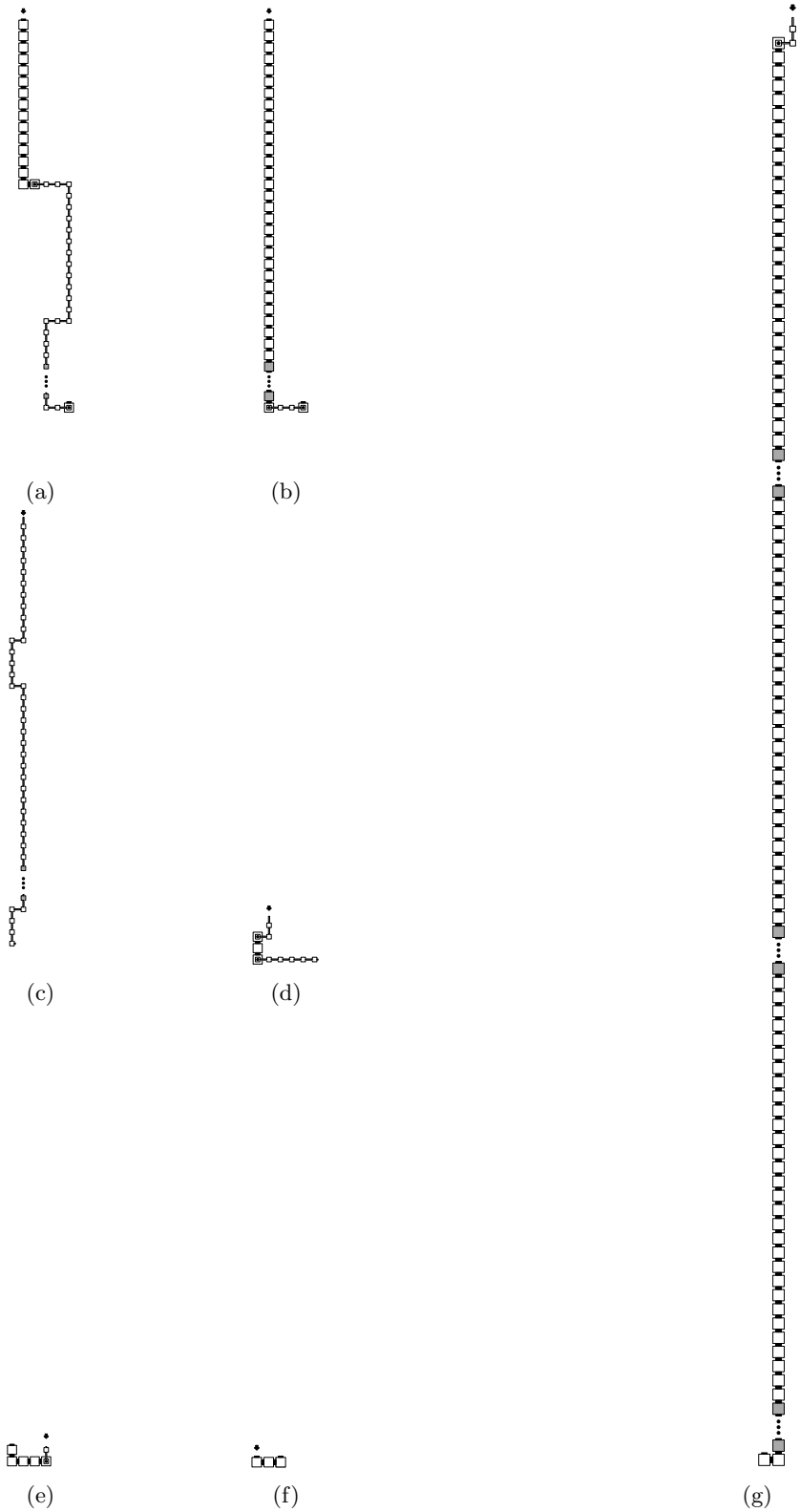


Figure 16: The `Next_Read` gadgets

For each $op \in \{\texttt{increment}, \texttt{copy}\}$:

- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 1, op\rangle, \langle \texttt{Read}, 2, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 16a.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 1, op, \texttt{msr}\rangle, \langle \texttt{Read}, 2, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 16b.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 1, op, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{Cross\_Next\_Row}, op\rangle)$
  from the micro-gadget shown in Figure 16c.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 2, op\rangle, \langle \texttt{Read}, 3, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 16a.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 2, op, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{Cross\_Next\_Row}, op\rangle)$
  from the micro-gadget shown in Figure 16c.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 3, op\rangle, \langle \texttt{Read}, 1, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 16g.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 3, op, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{Cross\_Next\_Row}, op\rangle)$
  from the micro-gadget shown in Figure 16d.

2.4.6 Cross over gadget

The idea this gadget is to assemble after reading the MSD, and route the counter to back to the start of the next row, in position for the counter to begin reading the first digit. The number of tiles shaded in darker grey is $6 \cdot \left\lfloor \frac{d}{3} \right\rfloor$.

For each $op \in \{\texttt{increment}, \texttt{copy}\}$:

- Create $\texttt{Crosser}(\langle \texttt{Cross\_Next\_Row}, op\rangle, \langle \texttt{Read}, 1, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 17.

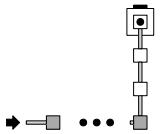
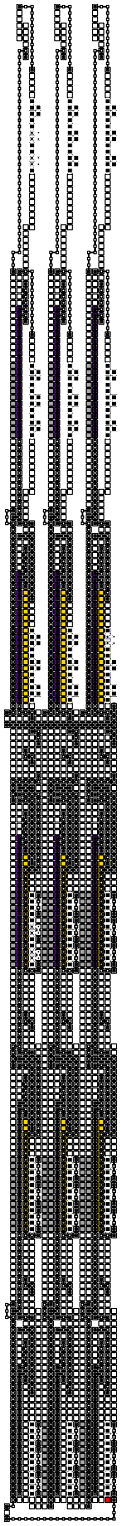
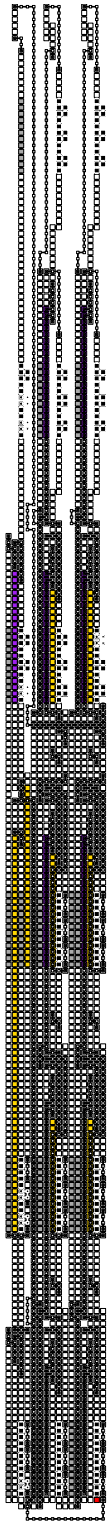


Figure 17: The `Crosser` gadget.

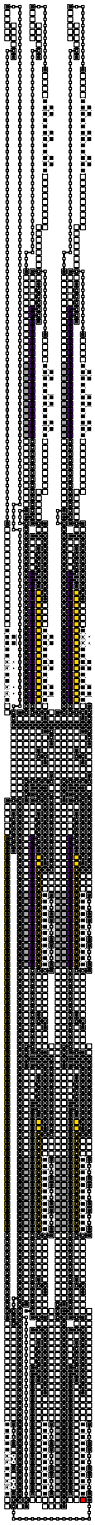
2.5 Overviews



(a) Full overview case 3



(b) Full overview case 2



(c) Full overview case 1