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 $\mathcal{C}_0$

...therefore, let $d = \lfloor \frac{k}{2} \rfloor$, $m = \left\lceil \left(\frac{N}{102}\right)^{\frac{1}{d}} \right\rceil$, $l = \lceil \log m \rceil + 2$, $\mathcal{C}_0 = m^d - \left\lfloor \frac{N-12l-76}{12l+90} \right\rfloor$, 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.

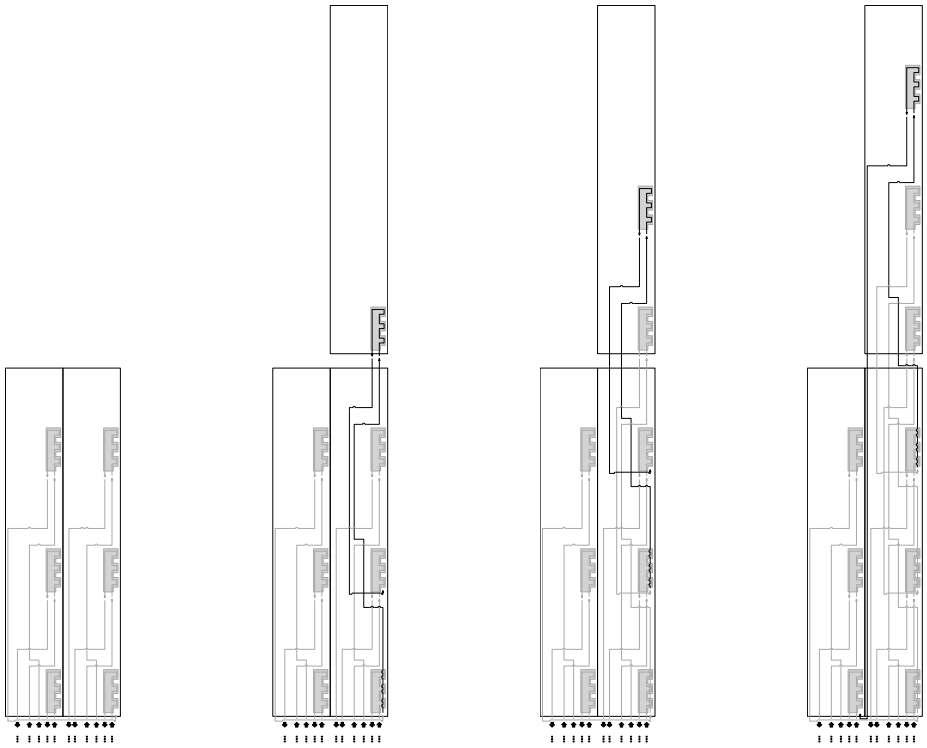
$$\begin{aligned} N = 102\left(\frac{N}{102}\right) &= 102\left(\left(\frac{N}{102}\right)^{\frac{1}{d}}\right)^d \leq 102\left\lceil\left(\frac{N}{102}\right)^{\frac{1}{d}}\right\rceil^d \\ &= 102m^d \leq 12lm^d + 90m^d \leq 12lm^d + 90m^d + 12l + 76 \\ &= m^d(12l + 90) + 12l + 76 \end{aligned}$$

□

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.

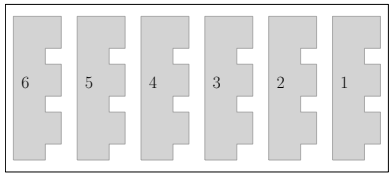
(d) Read digit 3 in the current row, write digit 3 in the next row.

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.

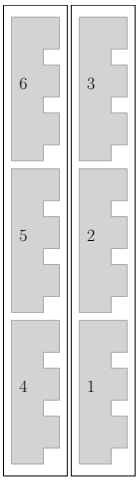
2.1 Digit region explanation (in progress)

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

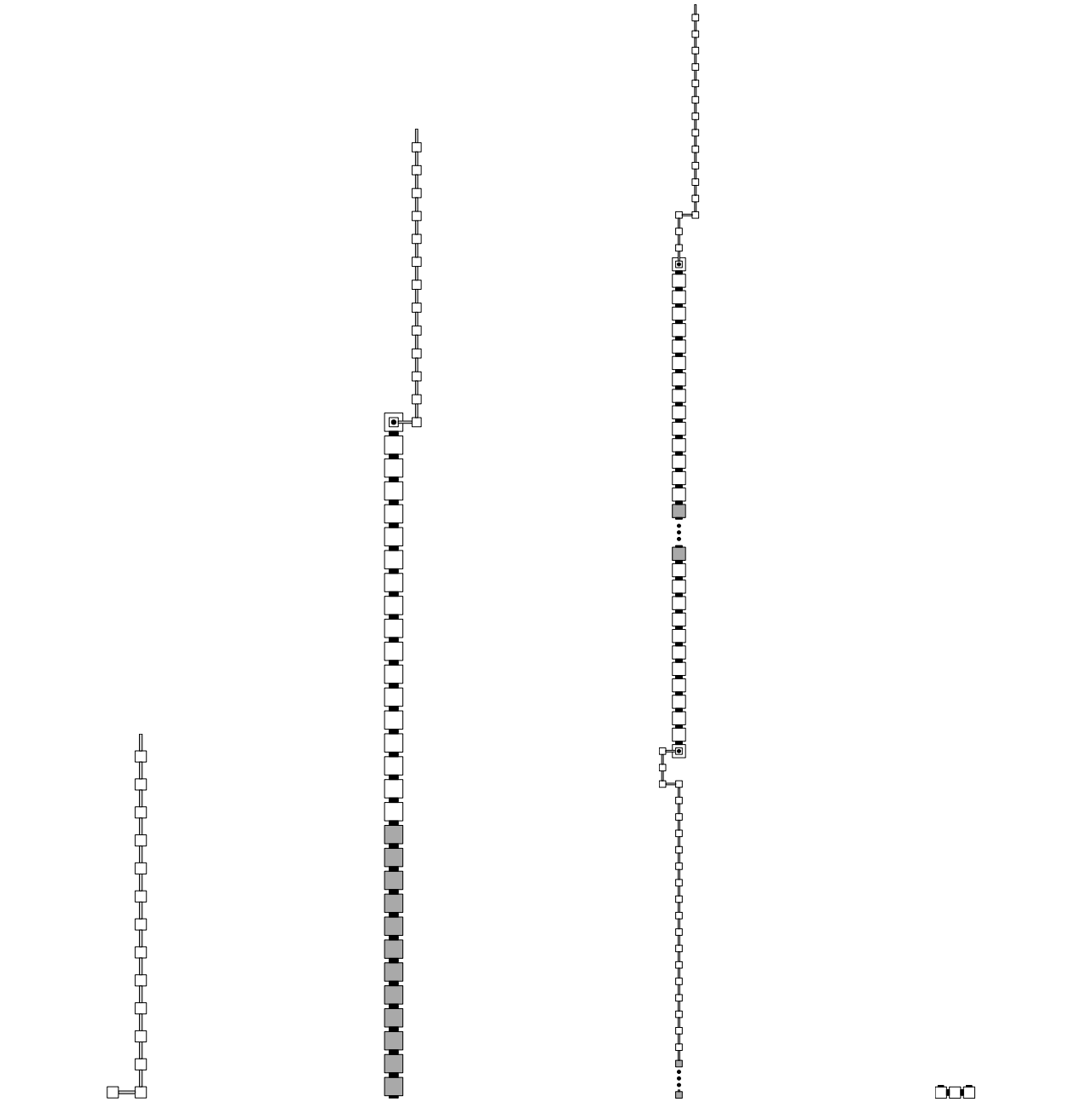
When describing a special case, i.e. "digit x – case y", whatever follows will only apply to the MSR (due to each case only affecting the MSR.)

2.3.1 Line Gadgets

(a) `Line_North` (b) `Line_South`

Figure 3: `Line` gadgets

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



(a) `Return_Start_A` (b) `Seed_Return_Digit_2` (c) `Seed_Return_Digit_3` (d) `Seed_Internal_Bridge`

(e) `Return_Digit_3_Bump` (f) `Seed_External_Bridge` (g) `Return_Row_Case3` (h) `Return_Row_Case1_And_Case2`

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.

while $i < d$:

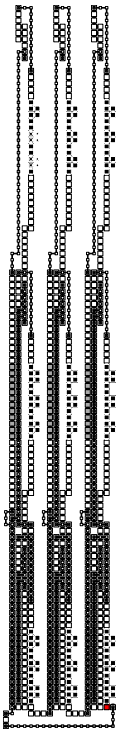
- if $i = 0$: create `SeedStart`($\langle$`SeedDigit1`$, i\rangle$)
- `Digit1` - for each $j = 0, \ldots, l$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Bit_Writer`($\langle$`SeedDigit1`$, i\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $0 \leqslant j \leqslant l$: create `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l$: `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedDigitTop`$, i\rangle$) 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`($\langle$`SeedDigitTop`$, i\rangle$, $\langle$`SeedDigitTopA`$, i\rangle$) from the micro-gadget shown in Figure 3a
  - Create `Topper`($\langle$`SeedDigitTopA`$, i\rangle$, $\langle$`SeedDigitTopB`$, i\rangle$) from the micro-gadget shown in Figure 11a
  - Create `South_Line4`l($\langle$`SeedDigitTopB`$, i\rangle$, $\langle$`SeedWarpInitializer`$, i\rangle$) from the micro-gadget shown in Figure 3b
- $i \leftarrow i + 1$
- Create `Seed_Warp_Initializer`($\langle$`SeedWarpInitializer`$, i-1\rangle$, $\langle$`SeedSecondWarp`$, i\rangle$)
- Create `Second_Warp`($\langle$`SeedSecondWarp`$, i\rangle$, $\langle$`SeedPostWarp`$, i\rangle$)
- Create `Post_Warp`($\langle$`SeedPostWarp`$, i\rangle$, $\langle$`SeedDigit2`$, i\rangle$) from the general gadget show in Figure 9b
- `Digit2`: for each $j = 0, \ldots, l$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Bit_Writer`($\langle$`SeedDigit2`$, i\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $0 \leqslant j \leqslant l$: create `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l$: `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedDigitTop2`$, i\rangle$) 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`($\langle$`SeedDigitTop`$, i\rangle$, $\langle$`SeedDigitTopA`$, i\rangle$) from the micro-gadget shown in Figure 3a
  - Create `Topper`($\langle$`SeedDigitTopA`$, i\rangle$, $\langle$`SeedDigitTopB`$, i\rangle$) from the micro-gadget shown in Figure 11a
  - Create `South_Line4`l($\langle$`SeedDigitTopB`$, i\rangle$, $\langle$`SeedReturnPath`$, i\rangle$) from the micro-gadget shown in Figure 3b
- `Return_Path`: the following statements create the gadget shown in Figure 4b

- Create `Return_Path`(⟨`SeedReturnPath`, i⟩ , ⟨`SeedReturnPathA`, i⟩) from the micro-gadget shown in Figure 4a
- Create `South_Line18`(⟨`SeedReturnPathA`, i⟩ , ⟨`SeedReturnPathB`, i⟩) from the micro-gadget shown in Figure 3b
- Create `South_Line4`l(⟨`SeedReturnPathB`, i⟩ , ⟨`SeedInternalBridge`, i⟩) from the micro-gadget shown in Figure 3b

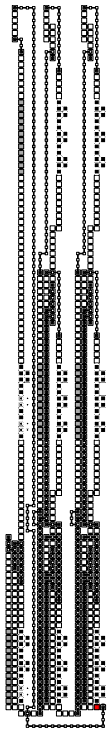
- $i \leftarrow i + 1$
- Create `Seed_Internal_Bridge`(⟨`SeedInternalBridge`, $i-1$⟩ , ⟨`SeedFirstWarp`, i⟩) from the general gadget shown in Figure 4d
- Create `First_Warp`(⟨`SeedFirstWarp`, i⟩ , ⟨`SeedWarpBridge`, i⟩)
- Create `Warp_Bridge`(⟨`SeedWarpBridge`, i⟩ , ⟨`SeedSecondWarp`, i⟩) from the general gadget shown in Figure 8a
- Create `Second_Warp`(⟨`SeedSecondWarp`, i⟩ , ⟨`SeedPostWarp`, i⟩)
- Create `Post_Warp`(⟨`SeedPostWarp`, i⟩ , ⟨`SeedDigit3`⟩) from the general gadget shown in Figure 9b
- `Digit3`: for each $j = 0, \ldots, l$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Bit_Writer`(⟨`SeedDigit3`, i⟩ , ⟨`SeedBit`, $i, j+1$⟩) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $0 \leqslant j \leqslant l$: create `Bit_Writer`(⟨`SeedBit`, i, j⟩ , ⟨`SeedBit`, $i, j+1$⟩) from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l$: create `Bit_Writer`(⟨`SeedBit`, i, j⟩ , ⟨`SeedDigitTop2`, i⟩) 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`(⟨`SeedDigitTop`, i⟩ , ⟨`SeedDigitTopA`, i⟩) from the micro-gadget shown in Figure 3a
  - Create `Topper`(⟨`SeedDigitTopA`, i⟩ , ⟨`SeedDigitTopB`, i⟩) from the micro-gadget shown in Figure 11a
  - Create `South_Line4`l(⟨`SeedDigitTopB`, i⟩ , ⟨`SeedReturnPath`, i⟩) from the micro-gadget shown in Figure 3b
- `Return_Path`: the following statements create the gadget shown in Figure 4b
  - Create `Return_Path`(⟨`SeedReturnPath`, i⟩ , ⟨`SeedReturnPathA`, i⟩) from the micro-gadget shown in Figure 4a
  - Create `South_Line3`(⟨`SeedReturnPathA`, i⟩ , ⟨`SeedReturnPathB`, i⟩) from the micro-gadget shown in Figure 3b
  - Create `South_Line15`(⟨`SeedReturnPathB`, i⟩ , ⟨`SeedReturnPathC`, i⟩) from the micro-gadget shown in Figure 3b
  - Create `South_Line4`l(⟨`SeedReturnPathC`, i⟩ , ⟨`SeedReturnPathD`, i⟩) from the micro-gadget shown in Figure 3b
  - Create `South_Line12`(⟨`SeedReturnPathD`, i⟩ , ⟨`SeedReturnPathE`, i⟩) from the micro-gadget shown in Figure 3b
  - Create `Return_Digit3_Bump`(⟨`SeedReturnPathE`, i⟩ , ⟨`SeedReturnPathF`, i⟩) from the micro-gadget shown in Figure 4e

- Create South_Line16(⟨SeedReturnPathF, i⟩ , ⟨SeedReturnPathG, i⟩) from the micro-gadget shown in Figure 3b
- Create South_Line4l(⟨SeedReturnPathG, i⟩ , ⟨SeedRegionEnd, i⟩) from the micro-gadget shown in Figure 3b

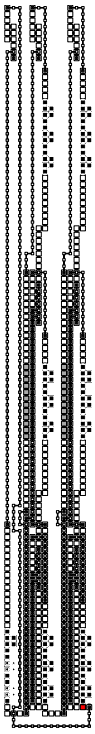
- if $i{+}1 \neq d$: create Seed_External_Bridge(⟨SeedRegionEnd, i⟩ , ⟨SeedDigit, $i+1$⟩) from the general gadget shown in Figure 4f
- $i \leftarrow i + 1$



(a) Initial value case 3



(b) Initial value case 2



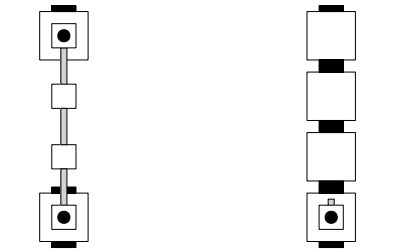
(c) Initial value case 1

2.4 Counter Unit

2.4.1 Digit readers

- For each $i = 1, 2, 3$, $j = l - 1, \ldots, 1$, $u \in \{0,1\}^j$, and op $\in$ {increment, copy}:
  - if $j = 0$: Create Bit_Reader(⟨DigitReader, i, λ, op⟩ , ⟨DigitReader, $i, 0$, op⟩ , ⟨DigitReader, $i, 1$, op⟩) from the general gadget in Figure 6
  - if $1 \leqslant j \leqslant l - 2$: Create Bit_Reader(⟨DigitReader, i, u, op⟩ , ⟨DigitReader, $i, 0u$, op⟩ , ⟨DigitReader, $i, 1u$, op⟩) from the general gadget in Figure 6

– if $j = l - 1$: Create `Bit_Reader`($\langle$`DigitReader`$, i, u,$ `op`$\rangle$,
$\langle$`PreWarp`$, i, 0u,$ `op`$\rangle$,
$\langle$`PreWarp`$, i, 1u,$ `op`$\rangle$)
from the general gadget in Figure 6



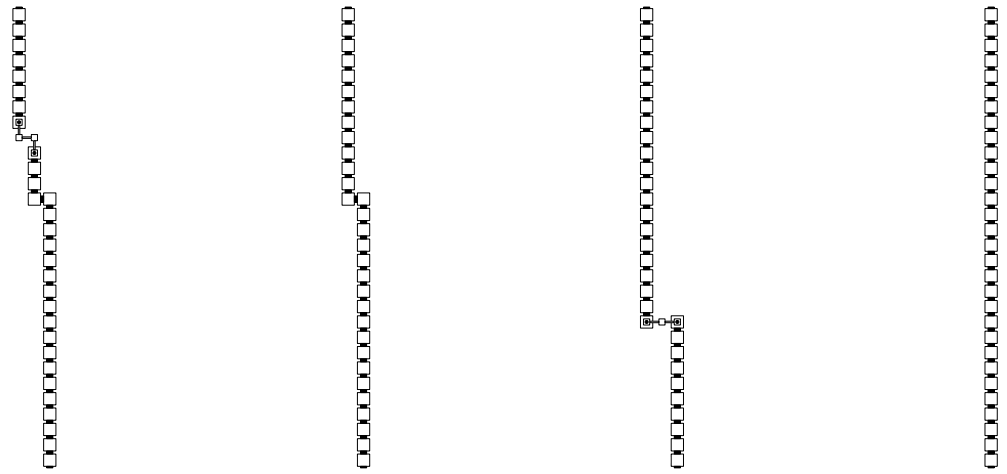
(a) `Read_0` (b) `Read_1`

Figure 6: `Read` gadgets

2.4.2 Warping

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

- `Pre_Warp`: These gadgets take the bits read from the `Bit_Reader` gadgets and convert it into a signal used until the `Digit_Top` gadgets are attached after writing the current digit. The signal started by this gadget is 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.
  - if u ends with 00: Create `Pre_Warp`($\langle$`PreWarp`$, i, u,$ `op`$\rangle$, $\langle$`FirstWarp`$, i, u,$ `op`$\rangle$)
    from the general gadget in Figure 7a
  - if u ends with 01: Create `Pre_Warp`($\langle$`PreWarp`$, i, u,$ `op`$\rangle$, $\langle$`FirstWarp`$, i, u,$ `op`$,$ `msr`$\rangle$)
    from the general gadget in Figure 7c
  - if u ends with 11: Create `Pre_Warp`($\langle$`PreWarp`$, i, u,$ `op`$\rangle$, $\langle$`FirstWarp`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$) from the general gadget in Figure 7b if $i = 1$ (case 1), or Figure 7d. $i = 2$ (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: `Pre_Warp` gadgets

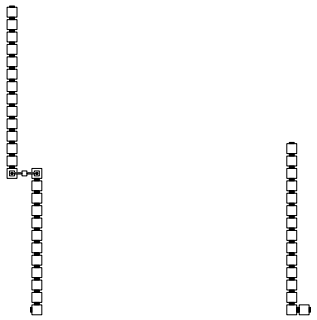
- `First_Warp`: A `First_Warp` connects to a `Warp_Bridge` gadget in all cases except when it's assembling in the MSR and it is digit 1 in case 1 or 2, in which the `First_Warp` gadget attaches directly to a `Post_Warp`.
  - Create `First_Warp`($\langle \texttt{FirstWarp}, i, u, \texttt{op} \rangle$, $\langle \texttt{FirstWarp}, i, u, \texttt{op} \rangle$, $\langle \texttt{WarpBridge}, i, u, \texttt{op} \rangle$)
  - (digit 1, case 2): Create `First_Warp`($\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr} \rangle$, $\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr} \rangle$, $\langle \texttt{PostWarp}, i, u, \texttt{op}, \texttt{msr} \rangle$)
  - (digit 1, case 1): Create `First_Warp`($\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{PostWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$)
  - (digit 2, case 2): Create `First_Warp`($\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{WarpBridge}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$)
  - (digit 3, case 3): Create `First_Warp`($\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{FirstWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{WarpBridge}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd} \rangle$)

- `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`.

– if u ends with 00:
Create `Warp_Bridge`($\langle$`WarpBridge`$, i, u,$ `op`$\rangle$, $\langle$`SecondWarp`$, i, u,$ `op`$\rangle$)
from the general gadget in Figure 8a

– if u ends with 11 and i is 2:
Create `Warp_Bridge`($\langle$`WarpBridge`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$, $\langle$`SecondWarp`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$)
from the general gadget in Figure 8b

– if u ends with 11 and i is 3:
Create `Warp_Bridge`($\langle$`WarpBridge`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$, $\langle$`SecondWarp`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$)
from the general gadget in Figure 8a



(a) General (b) Digit 2 – Case 2

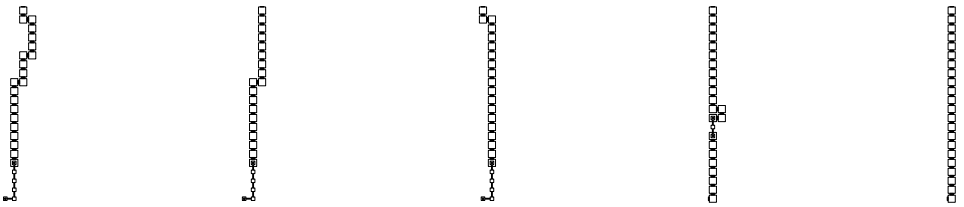
Figure 8: `Warp_Bridge` gadgets

- `Second_Warp`:
  - Create `Second_Warp`($\langle$`SecondWarp`$, i, u,$ `op`$\rangle$,
    $\langle$`SecondWarp`$, i, u,$ `op`$\rangle$,
    $\langle$`PostWarp`$, i, u,$ `op`$\rangle$)
  - Create `Second_Warp`($\langle$`SecondWarp`$, i, u,$ `op`$,$ `msr`$\rangle$,
    $\langle$`SecondWarp`$, i, u,$ `op`$,$ `msr`$\rangle$,
    $\langle$`PostWarp`$, i, u,$ `op`$,$ `msr`$\rangle$)
  - Create `Second_Warp`($\langle$`SecondWarp`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$,
    $\langle$`SecondWarp`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$,
    $\langle$`PostWarp`$, i, u,$ `op`$,$ `msr`$,$ `msd`$\rangle$)

- `Post_Warp`:
  - if u ends with 00:
    Create `Post_Warp`($\langle$`PostWarp`$, i, u,$ `op`$\rangle$, $\langle$`DigitWriter`$, i, u,$ `op`$\rangle$)
    Depending on i the gadget created in this step will differ: If i is 1 use from the general gadget in Figure 9a otherwise (i is 2 or 3) use from the general gadget in Figure 9b.

– if u ends with 01:
Create $\texttt{Post\_Warp}(\langle \texttt{PostWarp}, i, u, \texttt{op}, \texttt{msr}\rangle, \langle \texttt{DigitWriter}, i, u, \texttt{op}, \texttt{msr}\rangle)$
from the general gadget in Figure 9d.

– if u ends with 11:
Create $\texttt{Post\_Warp}(\langle \texttt{PostWarp}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{DigitWriter}, i, u, \texttt{op}, \texttt{msr}, \texttt{msd}\rangle)$
Depending on the number of digits in the MSR, the gadget created in this step will differ. If i is 1 (case 1) use the general gadget in Figure 9c. If i is 2 (case 2) use the general gadget in Figure 9e. If i is 3 (case 3) use the general gadget in Figure 9b.



(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: `Post_Warp` gadgets

2.4.3 Digit writers

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

– Create Bit_Writer(⟨DigitWriter, i, 0, op, msr, msd⟩ , ⟨DigitTop, i, op, msr, msd⟩) from the general gadget in Figure 10a

– Create Bit_Writer(⟨DigitWriter, i, 1, op, msr, msd⟩ , ⟨DigitTop, i, op, msr, msd⟩) from the general gadget in Figure 10b

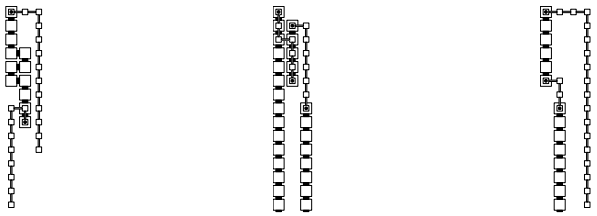
(a) Digit_Write_0 (b) Digit_Write_1

Figure 10: Digit_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.

If we examine the topper shown in Figure 11a,



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

Figure 11: Topper micro-gadgets

For each op ∈ {increment, copy}

- Digit 1 (general): the following statements create the gadget shown in Figure 12a
  - Create North_Line5(⟨DigitTop, 1, op⟩ , ⟨DigitTop_1_A, op⟩) from the micro-gadget shown in Figure 3a
  - Create Topper(⟨DigitTop_1_A, op⟩ , ⟨DigitTop_1_B, op⟩) from the micro-gadget shown in Figure 11a
  - Create South_Line4l(⟨DigitTop_1_B, op⟩ , ⟨ReturnD1ReadD2, op⟩) from the micro-gadget shown in Figure 3b

- Digit 1 (MSR): the following statements create the gadget shown in Figure 12d
  - Create Topper(⟨DigitTop, 1, op, msr⟩ , ⟨DigitTop_1_MSR_A, op⟩) from the micro-gadget shown in Figure 11b

– Create South_Line4l(⟨DigitTop_1_MSR_A, op, ⟩ , ⟨ReturnD1ReadD2 − Case2, op⟩) from the micro-gadget shown in Figure 3b

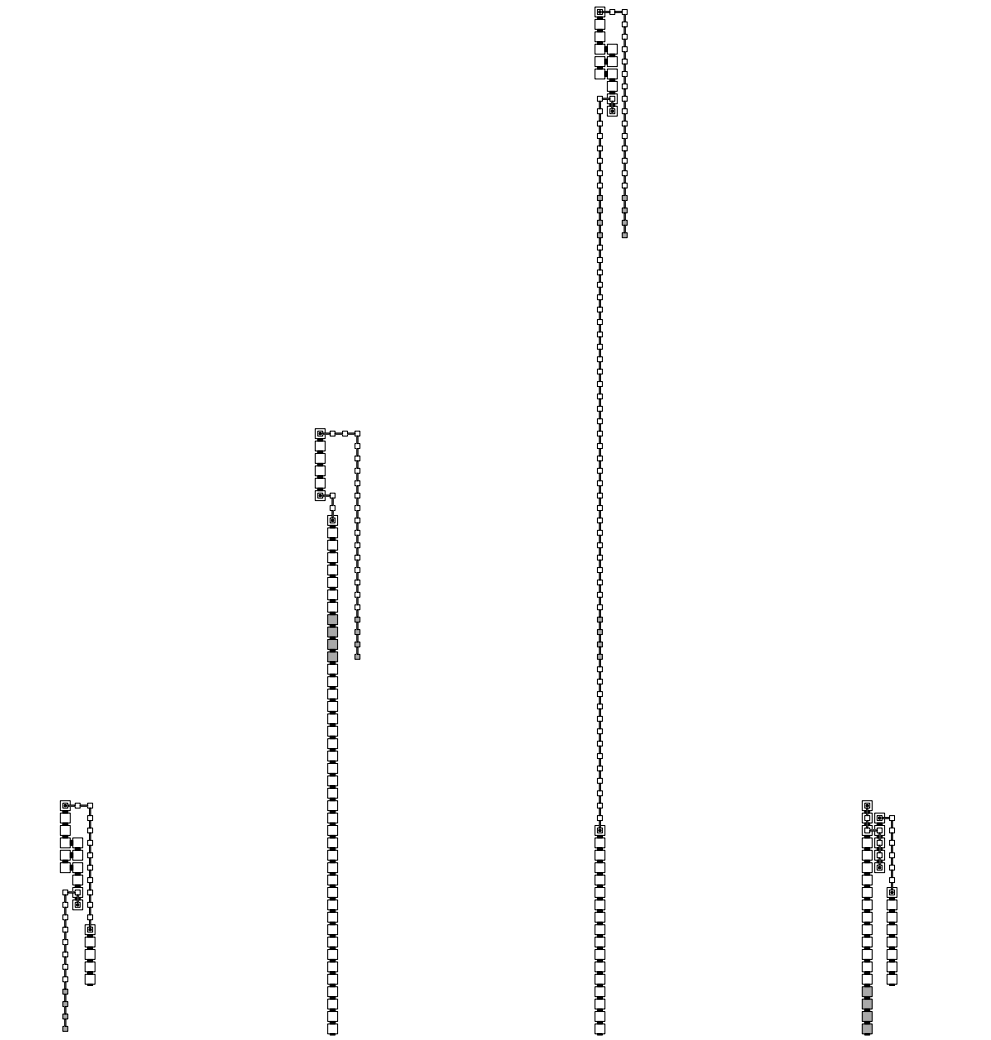
- Digit 1 (MSD): the following statements create the gadget shown in Figure 12c
  - Create North_Line4l(⟨DigitTop, 1, op, msr, msd⟩ , ⟨DigitTop_1_MSD_A, op⟩) from the micro-gadget shown in Figure 3a
  - Create North_Line4(⟨DigitTop_1_MSD_A, op⟩ , ⟨DigitTop_1_MSD_B, op⟩) from the micro-gadget shown in Figure 3a
  - Create Topper(⟨DigitTop_1_MSD_B, op⟩ , ⟨DigitTop_1_MSD_C, op⟩) from the micro-gadget shown in Figure 11a
  - Create South_Line4l(⟨DigitTop_1_MSD_C, op⟩ , ⟨DigitTop_1_MSD_D, op⟩) from the micro-gadget shown in Figure 3b
  - Create South_Line30(⟨DigitTop_1_MSD_D, op⟩ , ⟨DigitTop_1_MSD_E, op⟩) from the micro-gadget shown in Figure 3b
  - Create South_Line4l(⟨DigitTop_1_MSD_E, op⟩ , ⟨DigitTop_1_MSD_F, op⟩) from the micro-gadget shown in Figure 3b
  - Create South_Line14(⟨DigitTop_1_MSD_F, op⟩ , ⟨DigitTop_1_MSD_G, op⟩) from the micro-gadget shown in Figure 3b
  - Create South_Line17(⟨DigitTop_1_MSD_G, op⟩ , ⟨ReturnD1ReadNextRow, op⟩) 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⟩ , ⟨DigitTop_2_A, op⟩) from the micro-gadget shown in Figure 3a
  - Create Topper(⟨DigitTop_2_A, op⟩ , ⟨DigitTop_2_B, op⟩) from the micro-gadget shown in Figure 11a
  - Create South_Line4l(⟨DigitTop_2_B, op⟩ , ⟨ReturnD2ReadD3, op⟩) from the micro-gadget shown in Figure 3b

- Digit 2 (MSD): the following statements create the gadget shown in Figure 12b
  - Create North_Line4l(⟨DigitTop, 2, op, msr, msd⟩ , ⟨DigitTop_2_MSD_A, op⟩) from the micro-gadget shown in Figure 3a
  - Create Topper(⟨DigitTop_2_MSD_A, op⟩ , ⟨DigitTop_2_MSD_B, op⟩) from the micro-gadget shown in Figure 11c
  - Create South_Line4l(⟨DigitTop_2_MSD_B, op⟩ , ⟨DigitTop_2_MSD_C, op⟩) from the micro-gadget shown in Figure 3b
  - Create South_Line30(⟨DigitTop_2_MSD_C, op⟩ , ⟨ReturnD2ReadNextRow, op⟩) 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`⟩ , ⟨`DigitTop_3_A`, `op`⟩) from the micro-gadget shown in Figure 3a
  - Create `Topper`(⟨`DigitTop_3_A`, `op`⟩ , ⟨`DigitTop_3_B`, `op`⟩) from the micro-gadget shown in Figure 11a
  - Create `South_Line4`*l*(⟨`DigitTop_3_B`, `op`⟩ , ⟨`ReturnD3ReadD1`, `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`⟩ , ⟨`DigitTop_3_MSD_A`, `op`⟩) from the micro-gadget shown in Figure 3a
  - Create `Topper`(⟨`DigitTop_3_MSD_A`, `op`⟩ , ⟨`DigitTop_3_MSD_B`, `op`⟩) from the micro-gadget shown in Figure 11a
  - Create `South_Line4`*l*(⟨`DigitTop_3_MSD_B`, `op`⟩ , ⟨`ReturnD3ReadNextRow`, `op`⟩) from the micro-gadget shown in Figure 3b



(a) General (b) Digit 2 – Case 2 (c) Digit 1 – Case 1 (d) Digit 1 – Case 2

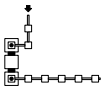
Figure 12: `Digit_Top` gadgets

2.4.5 Return paths between digits in the same row

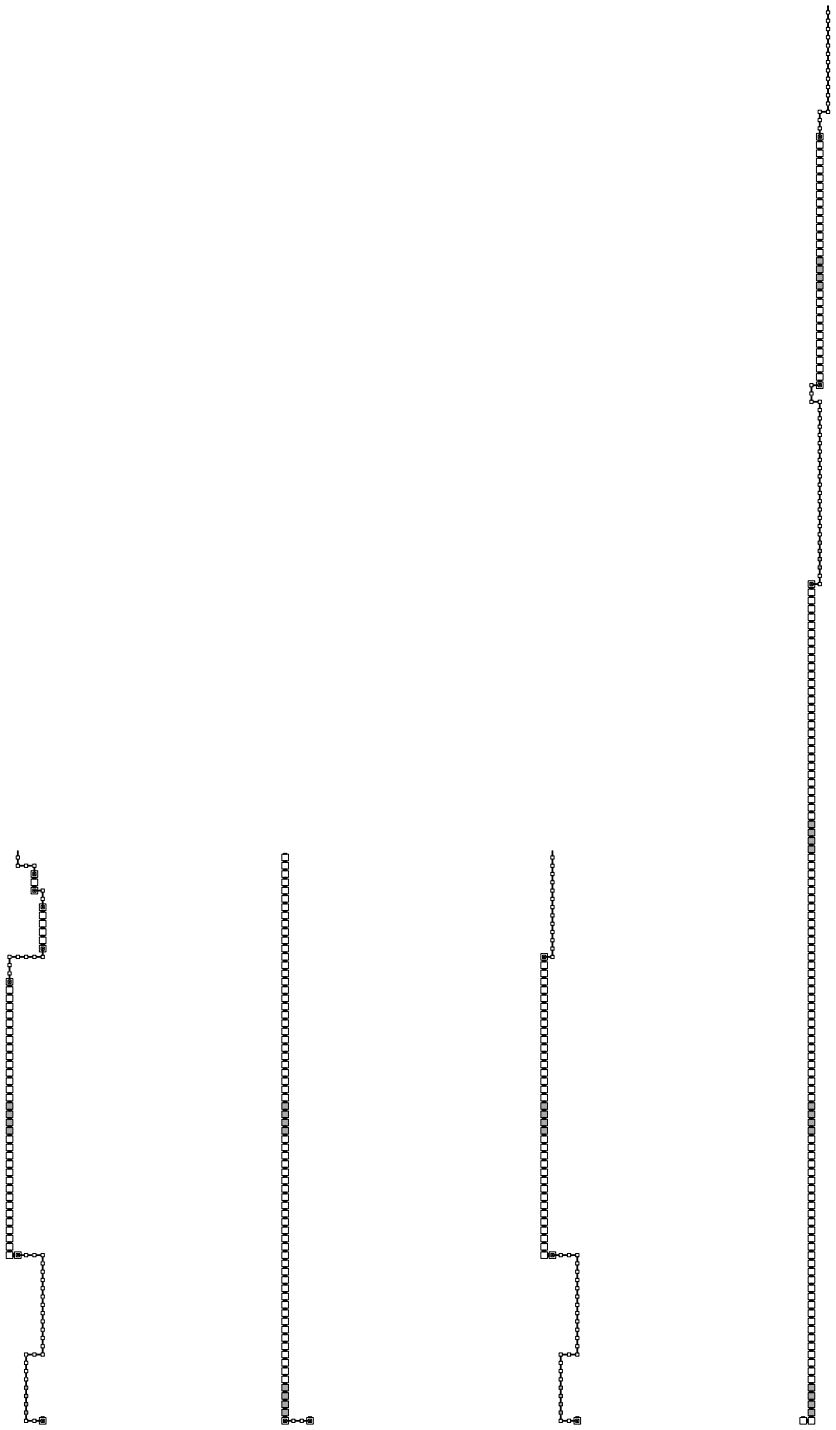
The gadgets of this class hold a increment/copy signal and the regional index of the next digit to read. The height of these gadgets is dependent on l. These gadgets are used so that upon writing a digit, the counter is able to move back down to the next digit in the current row, and continue reading.

(a) `Cross_To_Next_Row`

(b) `Exit_Row_From_Digit_1_Or_2`



(c) `Exit_Row_From_Digit_3`



(a) Return digit 1 read digit 2

(b) Return digit 1 read digit 2 – Case 2

(c) Return digit 2 read digit 3

(d) Return digit 3 read digit 1

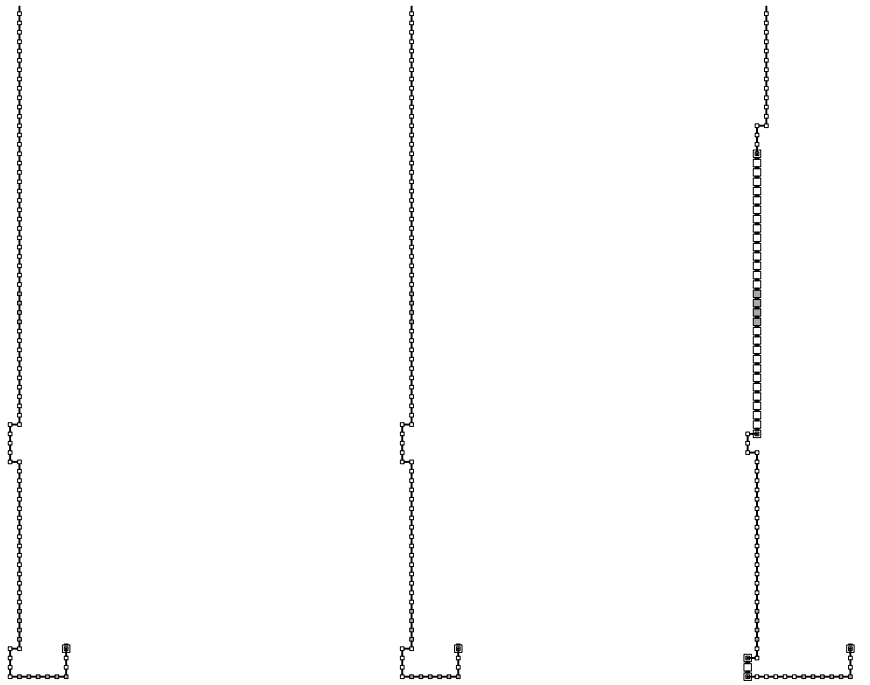
Figure 14: `Return_From_Digit_Read_Digit` gadgets. These gadgets assemble north to south, starting on the south side of a digit top.

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

- Create $\texttt{Return\_From\_Digit1\_Read\_Digit2}(\langle \texttt{ReturnD1ReadD2}, \texttt{op}\rangle, \langle \texttt{DigitReader}, 2, \lambda, \texttt{op}\rangle)$ from the general gadget in Figure 14a
- Create $\texttt{Return\_From\_Digit1\_Read\_Digit2\_Case2}(\langle \texttt{ReturnD1ReadD2} - \texttt{Case2}, \texttt{op}\rangle, \langle \texttt{DigitReader}, 2, \lambda, \texttt{op}\rangle)$ from the general gadget in Figure 14b
- Create $\texttt{Return\_From\_Digit2\_Read\_Digit3}(\langle \texttt{ReturnD2ReadD3}, \texttt{op}\rangle, \langle \texttt{DigitReader}, 3, \lambda, \texttt{op}\rangle)$ from the general gadget in Figure 14c
- Create $\texttt{Return\_From\_Digit3\_Read\_Digit1}(\langle \texttt{ReturnD3ReadD1}, \texttt{op}\rangle, \langle \texttt{DigitReader}, 1, \lambda, \texttt{op}\rangle)$ from the general gadget in Figure 14d

2.4.6 Return paths between the MSD and LSD in different rows

The gadgets of this class hold a increment/copy signal. The height of these gadgets is dependent on l and the width is dependent of k. These gadgets are used to begin reading the first digit in the following row, once the MSD has been read in the current row.



(a) Case – 1 (b) Case – 2 (c) Case – 3

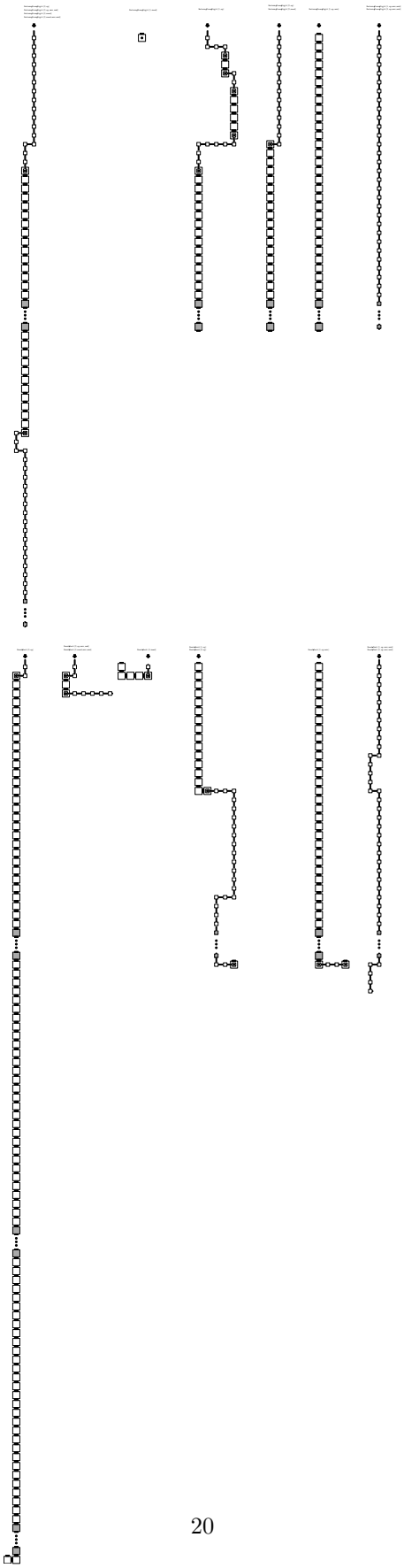
Figure 15: `Return_From_Digit_Read_Next_Row` gadgets. All of these gadgets assemble north to south. The vertical gray lines tiles have a height that depends on l and the horizontal gray lines depend on k. (cases 1 and 2 are geometrically equivalent)

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

- Create `Return_From_Digit1_Read_Next_Row`($\langle$`ReturnD1ReadNextRow`, `op`$\rangle$, $\langle$`DigitReader`, $1, \lambda$, `op`$\rangle$) from the general gadget in Figure 15a
- Create `Return_From_Digit2_Read_Next_Row`($\langle$`ReturnD2ReadNextRow`, `op`$\rangle$, $\langle$`DigitReader`, $1, \lambda$, `op`$\rangle$) from the general gadget in Figure 15b
- Create `Return_From_Digit3_Read_Next_Row`($\langle$`ReturnD3ReadNextRow`, `op`$\rangle$, $\langle$`DigitReader`, $1, \lambda$, `op`$\rangle$) from the general gadget in Figure 15c

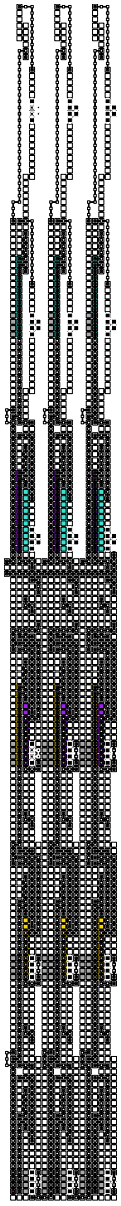
2.4.7 Return paths

In this section, we explain the gadgets used after writing a return path...

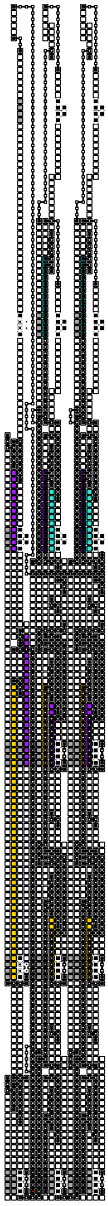


(a)

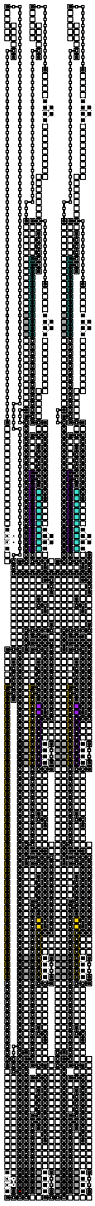
2.5 Overviews



(a) Full overview case 3



(b) Full overview case 2



(c) Full overview case 1