Part I
Upper bound

1 Notation for gadgets and figures

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

MSR = most signifcant digit region

$s =$ starting value of counter

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

$m^d =$ final value of the counter

$n = m^d - s$, 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

2 Parameters for the counter

...therefore, let $d = \lfloor \frac{k}{2} \rfloor$, $m = \left\lceil \left(\frac{N}{102}\right)^{\frac{1}{d}} \right\rceil$, $l = \lceil \log m \rceil + 2$, $s = m^d - \left\lfloor \frac{N-12l-94}{12l+90} \right\rfloor$, 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 s is the starting value of the counter in decimal.

Let h be the height of the construction without any additional "roof" tiles. By inspection...we see that the height of the construction without additional tiles is $12l + 94$ for the `Seed` unit, plus $12l + 90$ for each incrementation of the counter, adding a `Counter` unit row each time. If we define n as the number of `Counter` unit rows, then $h = n(12l + 90) + (12l + 94)$. So then the maximum height of the counter is $m^d(12l + 90) + 12l + 94$. 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. $N \leq m^d(12l + 90) + 12l + 94$.

Proof.

$$\begin{aligned} N &= 102\left(\frac{N}{102}\right) = 102\left(\left(\frac{N}{102}\right)^{\frac{1}{d}}\right)^d \leq 102\left\lceil\left(\frac{N}{102}\right)^{\frac{1}{d}}\right\rceil^d \\ &= 102m^d \leq 12lm^d + 90m^d \leq 12lm^d + 90m^d + 12l + 94 \\ &= m^d(12l + 90) + 12l + 94 \end{aligned}$$

□

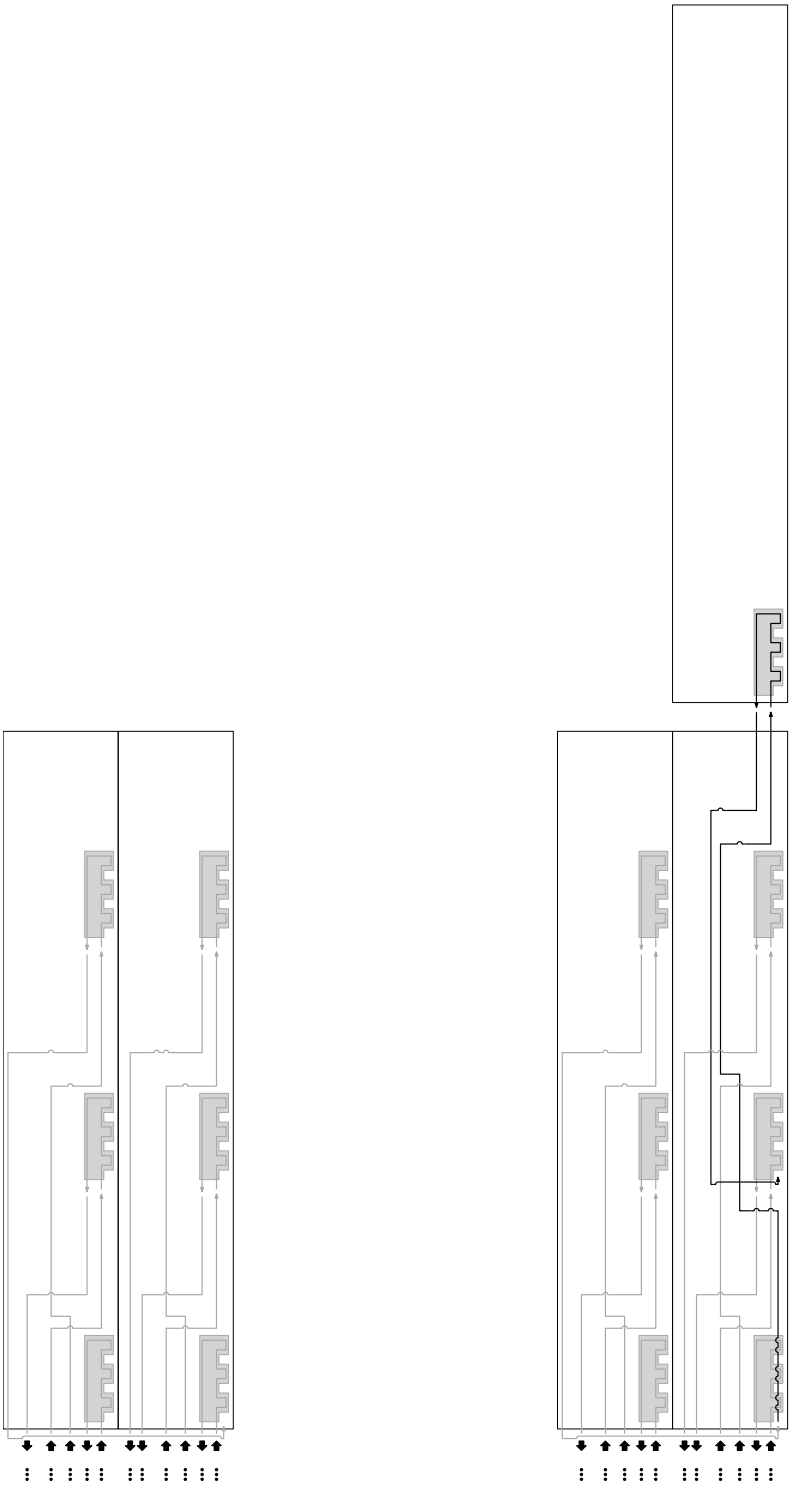
And the mininum height is $24l + 184$.

One row of the counter might result in a final assembly that will not be tall enough but ... says that having all possible rows of the counter might result in a final assembly that is too tall. Therefore, we must start the counter at an appropriate value to get the correct height of the final assembly.

The counter can start at any whole number less than m^d and ends when it reaches 0 by rolling over $m^d - 1$. This means that the number of Counter unit rows n, is $m^d - s$, where we have defined s as the starting value of the counter. To choose the best starting value, we find the value for n that gets h close to N without exceeding N. It follows from the equation $h = n(12l + 90) + 12l + 94$, that $n = \left\lfloor \frac{N-12l-94}{12l+90} \right\rfloor$. Thus, $s = m^d - \left\lfloor \frac{N-12-94}{12l+90} \right\rfloor$.

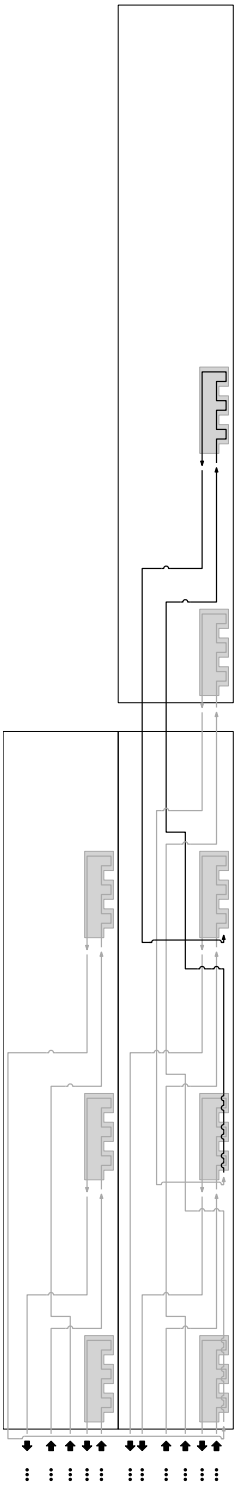
If k is an even number, no filler tiles are needed. We can assemble any rectangle that has a width k, provided that k is some even number ≥ 6. If k is any odd number ≥ 6, we use one filler tiles. ...as a result of each digit requiring a width of 2 tiles, if k is odd, one additional tile column must be added. The number of filler tiles needed for the width is $k \mod 2$, and the number of filler tiles for the height is $N - 12l - 90 \mod 12l + 90$.

3 Counting with digit regions

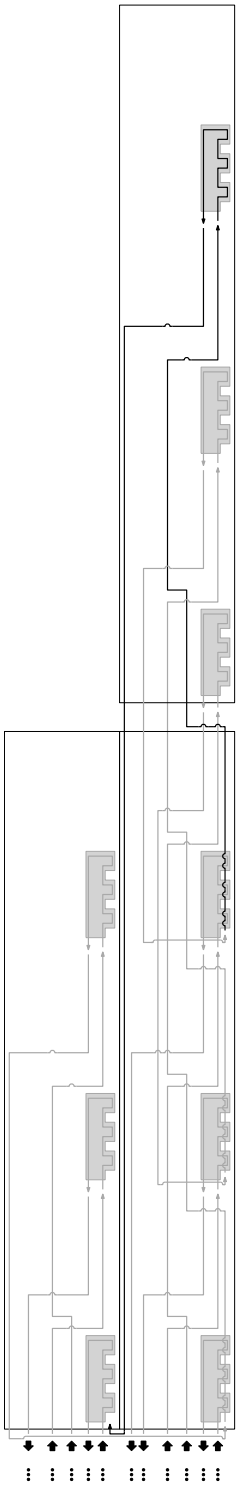


(a) A "clean" counter row, before any reading has started.

(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.

3.1 Digit regions

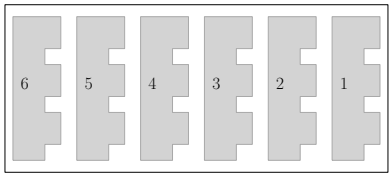
Each logical row of the counter is made up of $\lceil \frac{d}{3} \rceil$ "digit regions". A digit region is a group of 1-3 digits, stacked vertically on top of one another. Within a digit region, the digits are sorted in order of significance, thus the top digit is the most signifcant digit, the middle 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 digit 1 in the rightmost digit region, then writes digit $1'$ in the digit region directly north in the next row. After writing this first digit, the counter returns back to the current row and reads digit 2. Upon reading digit 2 in the current row, it then writes digit $2'$ in the next row and returns to the current row. Finally, in the current row, it reads digit 3, the last digit remaining, and writes digit $3'$ in the next row. Now that all three digits in this digit region have been evaluated, the counter will repeat this process for all remaining digit regions in the current row.

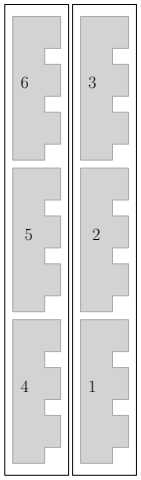
In the last digit region, which we will refer to as the MSR (most significant region), the region may or may not contain all 3 digits like the earlier regions. Because of this, there are three cases we must account for: case 1 if the MSR only has one digit; case 2 if the MSR only has two digits; or case 3 if the MSR has three digits.

In case 1, the MSR adds an additional 2 tiles to the overall width of the rectangle. In case 2, the MSR adds an additional 4 tiles to the overall width of the rectangle. And, similar to all regular digit regions with 3 digits, in case 3 the MSR adds an additional 6 tiles to the overall width of the rectangle. As a result of these cases, we're able to utilize these different to fine tune the width of the rectangle such that we only need one filler tile in the event that k is odd.

Figure 2: Digits in a typical counter vs. digits separated into digit regions.



(a) Digits in a typical counter.



(b) Digits in two digit regions, stacked vertically. Digits 1-3 are in digit region 1, and digits 4-6 are in digit region 2.

Contrary to a typical counter, each counter row has an approximate height of 3 digits $\approx 12l$. The digits are stacked up to 3, once a digit region is full, we add another digit region to the left, and repeat.

3.2 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.

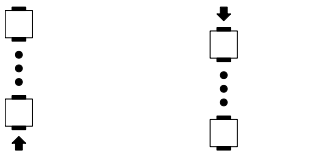
bit_1	bit_0	Meaning
0	0	digit is not in MSR
0	1	digit is in the MSR but is not the MSD
1	0	
1	1	digit is in the MSR and is MSD

4 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.)

4.1 Gadgets

4.1.1 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.

4.1.2 `Counter_Read`

- For each $i = 1, 2, 3$, $j = 0, \ldots, l-2$, $u \in \{0,1\}^j$, and $op \in \{\texttt{increment}, \texttt{copy}, \texttt{halt}\}$:
  - if $j = 0$: create `Counter_Read`($\langle \texttt{CounterRead}, i, \lambda, op \rangle$, $\langle \texttt{CounterRead}, i, 0, op \rangle$, $\langle \texttt{CounterRead}, i, 1, op \rangle$) from the general gadget in Figure 4.
  - else: create `Counter_Read`($\langle \texttt{CounterRead}, i, u, op \rangle$, $\langle \texttt{CounterRead}, i, 0u, op \rangle$, $\langle \texttt{CounterRead}, i, 1u, op \rangle$) from the general gadget in Figure 4.
- For each $i = 1, 2, 3$ and each $u \in \{0,1\}^{l-1}$:
  - Create `Counter_Read`($\langle \texttt{CounterRead}, i, u, \texttt{copy} \rangle$, $\langle \texttt{PreWarp}, i, 0u, \texttt{copy} \rangle$, $\langle \texttt{PreWarp}, i, 1u, \texttt{copy} \rangle$) from the general gadget in Figure 4.

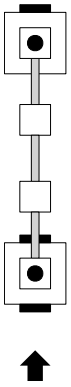
  Since the counter must only increment the current value if the result will be less than m, the `Counter_Read` gadgets that have both an `increment` signal and input size of $l-2$ must first right

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 + $2^{\log(m)-1}$ + 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 output a `copy` signal. Otherwise, if the counter is unable to increment the value, it outputs signal in which the bits of the digit is all zeroes and it will propagate the `increment` signal to the next digit.

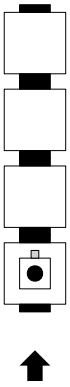
Algorithm 1 Incrementing and halting

function ReadMostSignicantBit( )
  $guess0 \leftarrow 0u >> 2$.
  $guess1 \leftarrow 1u >> 2$.
  if $convertToDecimal(guess0) + 1 < m - 1$ **then**
    $out0 \leftarrow \langle \texttt{PreWarp}, i, convertToBinary(convertToDecimal(guess0) + 1) + u[1] + [0], \texttt{copy} \rangle$.
  else
    if u ends with "11" **then**
      $out0 \leftarrow \langle \texttt{PreWarp}, i, repeat(\text{“0”}, m) + 11, \texttt{halt} \rangle$.
    else
      $out0 \leftarrow \langle \texttt{PreWarp}, i, repeat(\text{“0”}, m) + u[1] + u[0], \texttt{increment} \rangle$.
  if $convertToDecimal(guess1) + 1 < m - 1$ **then**
    $out1 \leftarrow \langle \texttt{PreWarp}, i, convertToBinary(convertToDecimal(guess1) + 1) + u[1] + [0], \texttt{copy} \rangle$.
  else
    if u ends with "11" **then**
      $out1 \leftarrow \langle \texttt{PreWarp}, i, repeat(\text{“0”}, m) + 11, \texttt{halt} \rangle$.
    else
      $out1 \leftarrow \langle \texttt{PreWarp}, i, repeat(\text{“0”}, m) + u[1] + u[0], \texttt{increment} \rangle$.

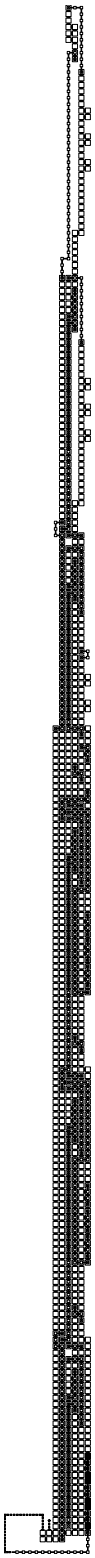
– Create `Counter_Read`($\langle \texttt{CounterRead}, i, u, \texttt{increment} \rangle, out0, out1$)
from the general gadget in Figure 4.



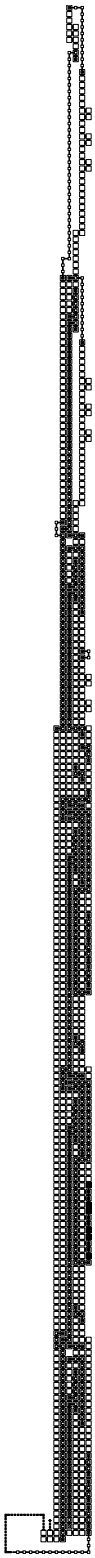
(a) `Counter_Read_0`



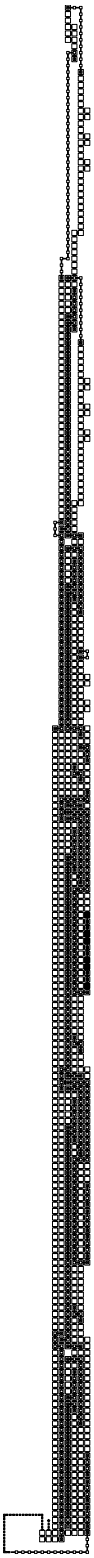
(b) `Counter_Read_1`



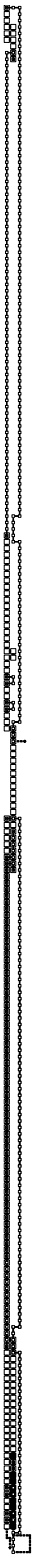
(c) Digit 1 - general overview.



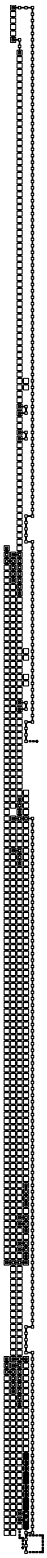
(d) Digit 2 - general overview.



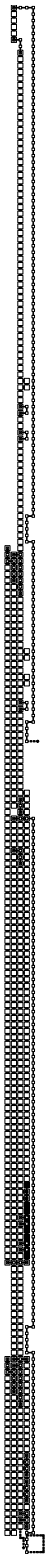
(e) Digit 3 - general overview.



(f) Digit 1 - case 1 overview.



(g) Digit 1 - case 2 overview.



(h) Digit 2 - case 2 overview.

(i) Digit 3 - case 3 overview.

Figure 4: The `Counter_Read` gadgets.

4.1.3 Warp_Unit

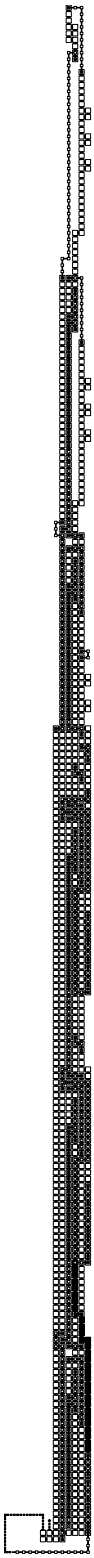
We now explain the Warp_Unit. A warp unit generally consists of the following 5 gadgets: Pre_Warp, First_Warp, Warp_Bridge, Second_Warp, Post_Warp. The job of these 5 gadgets is to transport the value read by the Counter_Read all the way to the digit region in the next row, so that the Counter_Write 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 decode the least significant two bits passed in from the read from the Counter_Read gadgets, into a special signal that indicates whether a gadget is currently in the MSR and if it is the MSD. Here is how the signal is decoded: if the least significant bit is 1, add a marker to indicate the current gadgets are in the MSR; if the second least significant bit is also 1, add a marker on the glues to indicate that the gadget is also part of the MSD.

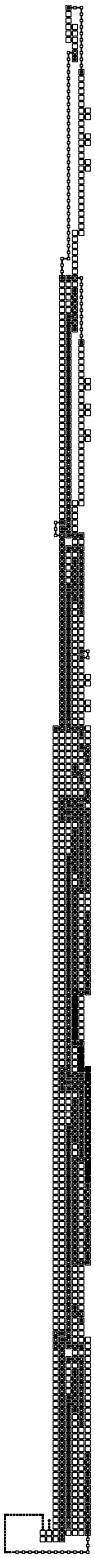
  For each $i = 1, 2, 3, u \in \{0,1\}^l$, and each $op \in \{\texttt{increment}, \texttt{copy}, \texttt{halt}\}$:

  - if u ends with 00: create $\texttt{Pre\_Warp}(\langle \texttt{PreWarp}, i, u, op\rangle, \langle \texttt{FirstWarp}, i, u, op\rangle)$ from the general gadget in Figure 5a.
  - if u ends with 01: create $\texttt{Pre\_Warp}(\langle \texttt{PreWarp}, 1, u, op\rangle, \langle \texttt{FirstWarp}, 1, u, op, \texttt{msr}\rangle)$ from the general gadget in Figure 5e.
  - 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 5g if $i = 1$ (case 1), or Figure 5i if $i = 2$ (case 2), or Figure 5a if $i = 3$ (case 3).

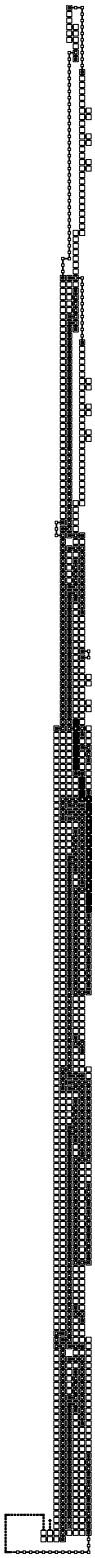
(a) Digits 1, 2, & 3 - general, Digit 3 - case 3



(b) Digit 1 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

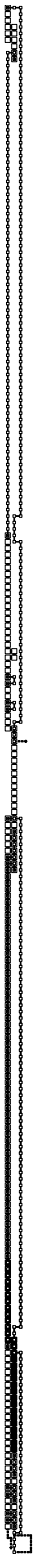


(c) Digit 2 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

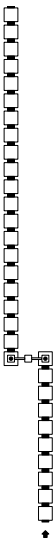


(d) Digit 3 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

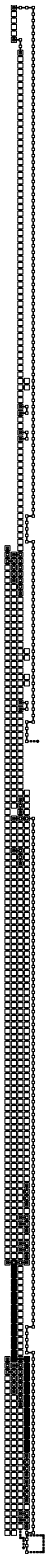
(e) Digit 1 - case 1.



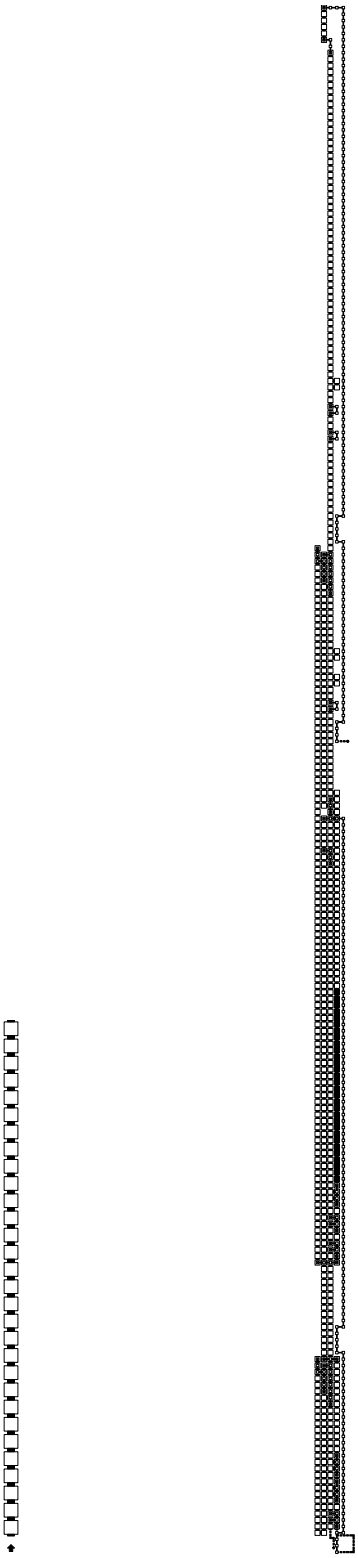
(f) Digit 1 - case 1 overview. The black tiles in this figure correspond to the gadget shown in subfigure e.



(g) Digit 1 - case 2.



(h) Digit 1 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure g.



(i) Digit 2 - case 2.

(j) Digit 2 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure i.

Figure 5: The `Pre_Warp` gadgets.

- `First_Warp`: The idea of the `First_Warp` gadget is to transport the information read by the `Counter_Read` gadgets, usually across a distance $O(\log m)$. We do this using a single tile that assembles an infinite line in the north direction, and has one unique glue either in the east direction or west direction. This unique glue will at some point later in the assembly, that is determined by earlier parts of the assembly, no longer be blocked. When this occurs, it can finally attach to the `Warp_Bridge` gadget (except in a few special cases). This process signifies the "waking up" of the `First_Warp` gadgets. When this gadget wakes up, it must also be blocked in the north direction, which prevents a truly infinite line from assembling. The geometry required for this process is guarenteed to be in place by earlier-assembled `Digit_Top` gadgets.

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

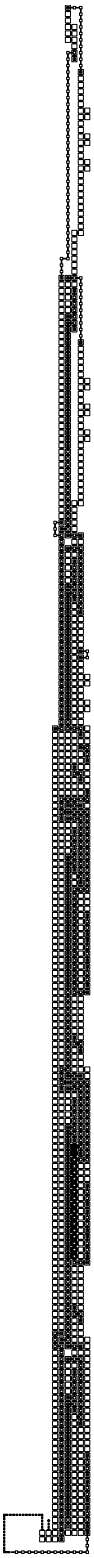
  - Create `First_Warp`($\langle \texttt{FirstWarp}, i, u, op \rangle$,
    $\langle \texttt{FirstWarp}, i, u, op \rangle$,
    $\langle \texttt{WarpBridge}, i, u, op \rangle$)
    from the single tile gadget, shown in Figure 6a if $i = 1$ or Figure 6b if $i = 2$, otherwise from Figure 6c if $i = 3$.

  - Create `First_Warp`($\langle \texttt{FirstWarp}, 1, u, op, \texttt{msr} \rangle$,
    $\langle \texttt{FirstWarp}, 1, u, op, \texttt{msr} \rangle$,
    $\langle \texttt{PostWarp}, 1, u, op, \texttt{msr} \rangle$)
    from the single tile gadget shown in Figure 6f.

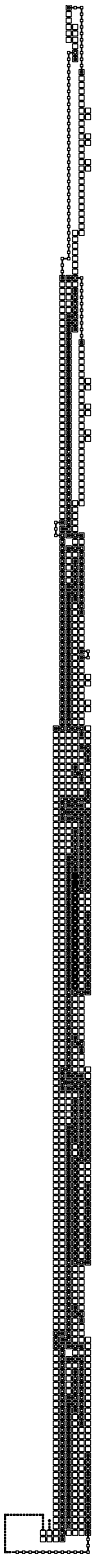
  - Create `First_Warp`($\langle \texttt{FirstWarp}, 1, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{FirstWarp}, 1, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{PostWarp}, 1, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the single tile gadget shown in Figure 6e.

  - Create `First_Warp`($\langle \texttt{FirstWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{FirstWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{WarpBridge}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the single tile gadget shown in Figure 6g.

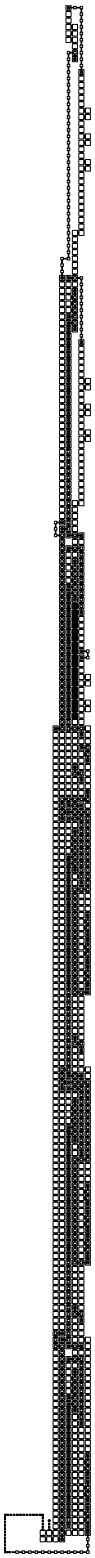
  - Create `First_Warp`($\langle \texttt{FirstWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{FirstWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{WarpBridge}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the single tile gadget shown in Figure 6h.



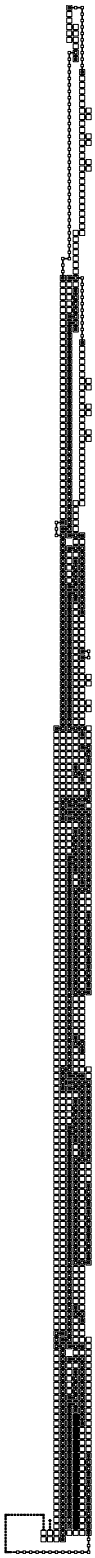
(a) Digit 1 - general overview.



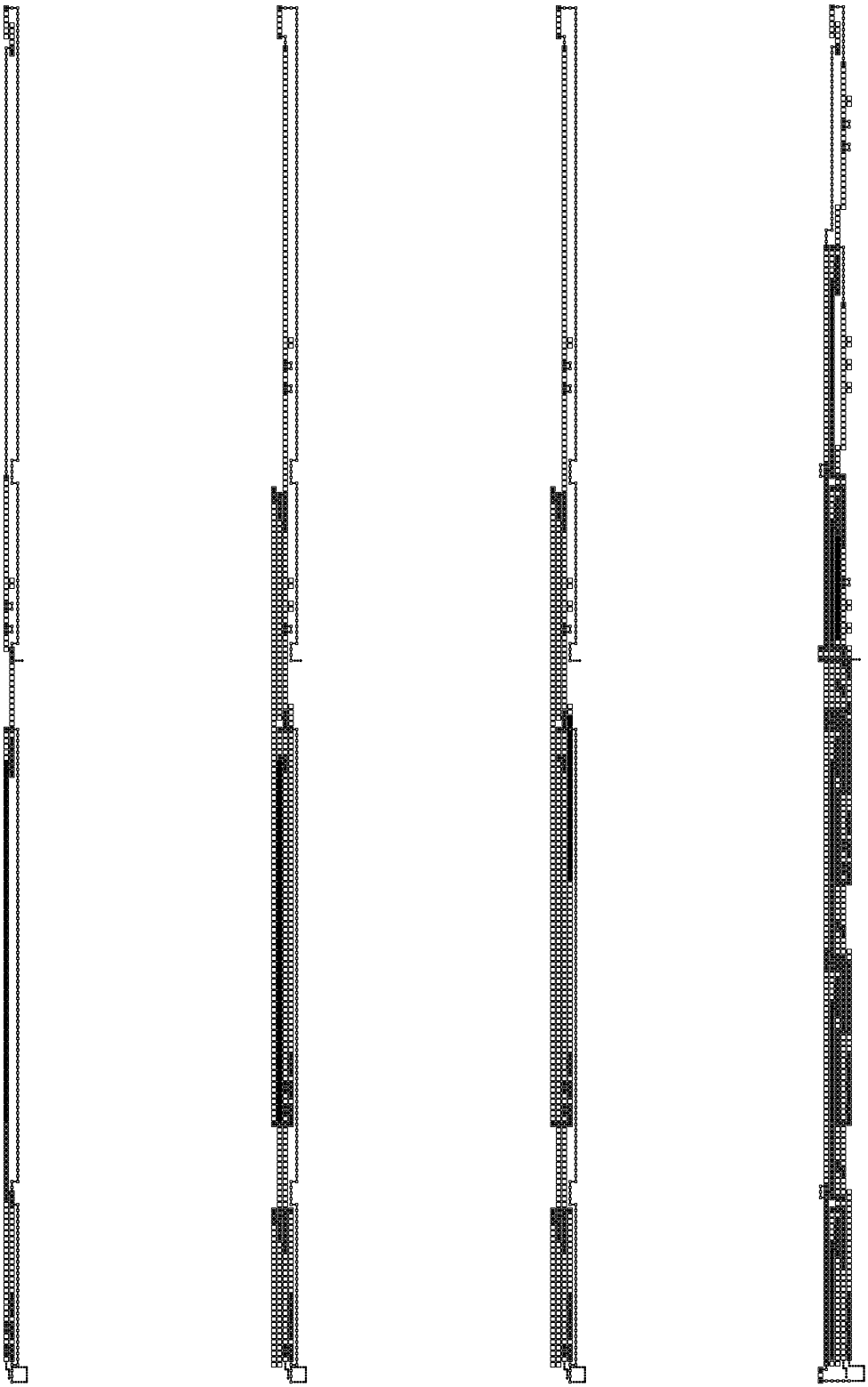
(b) Digit 2 - general overview.



(c) Digit 3 - general overview.



(d) Digit 3 - general (seed) overview.



(e) Digit 1 - case 1 overview. (f) Digit 1 - case 2 overview. (g) Digit 2 - case 2 overview. (h) Digit 3 - case 3 overview.

Figure 6: The `First_Warp` gadget overviews.

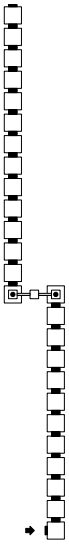
- `Warp_Bridge`: a `Warp_Bridge` gadget is a gadget which assembles when the `First_Warp` gadget makes it to its final destination. The goal of the `Warp_Bridge` is to assemble a path from the end of the `First_Warp` gadgets to the start of the `Second_Warp` gadgets. For digit 1 in cases 1 and 2, the `Warp_Bridge` is omitted from the `Warp_Unit`.

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

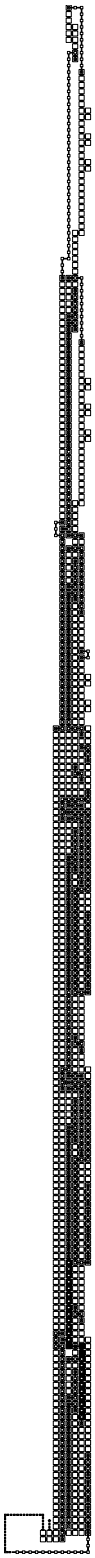
  - Create `Warp_Bridge`($\langle \texttt{WarpBridge}, i, u, op \rangle$, $\langle \texttt{SecondWarp}, i, u, op \rangle$)
    from the general gadget in Figure 7a.

  - Create `Warp_Bridge`($\langle \texttt{WarpBridge}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{SecondWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the general gadget in Figure 7g.

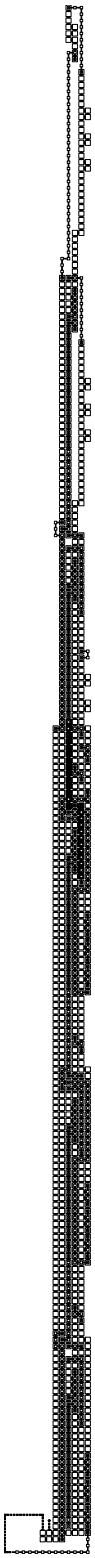
  - Create `Warp_Bridge`($\langle \texttt{WarpBridge}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{SecondWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the general gadget in Figure 7a.



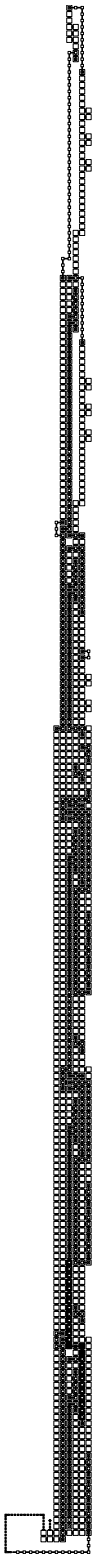
(a) Digits 1, 2, & 3 general.



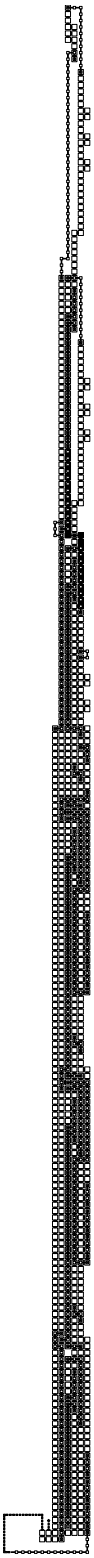
(b) Digit 1 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



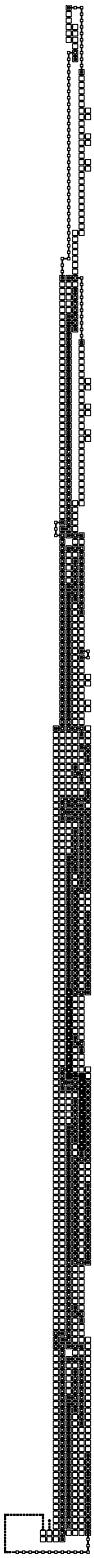
(c) Digit 2 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



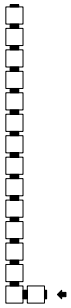
(d) Digit 2 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



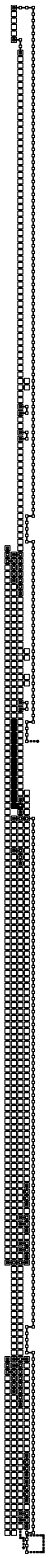
(e) Digit 3 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



(f) Digit 3 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



(g) Digit 2 - case 2.



(h) Digit 2 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure g.

Figure 7: The `Warp_Bridge` gadgets.

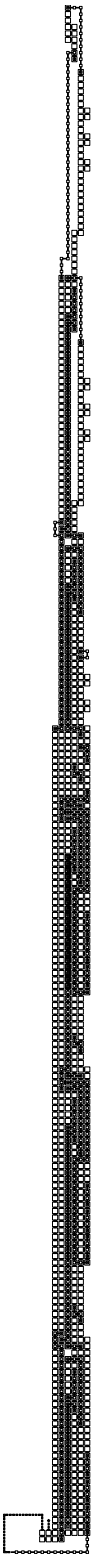
- `Second_Warp`: Similar to the `First_Warp` gadgets, the idea of the `Second_Warp` gadget is to also to transport the information read by the `Counter_Read` gadgets. We do this using a single tile that assembles an infinite line in the north direction, and has one unique glue either in the east direction or up direction. This unique glue will at some point later in the assembly, that is determined by earlier parts of the assembly, no longer be blocked. When this occurs, it can finally attach to the `Post_Warp` gadget. This process signifies the "waking up" of the `Second_Warp` gadgets. When this gadget wakes up, it must also be blocked in the north direction, which prevent a truly infinite line from assembling. The geometry required for this process is guarenteed to be in place by earlier-assembled `Digit_Top` gadgets.

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

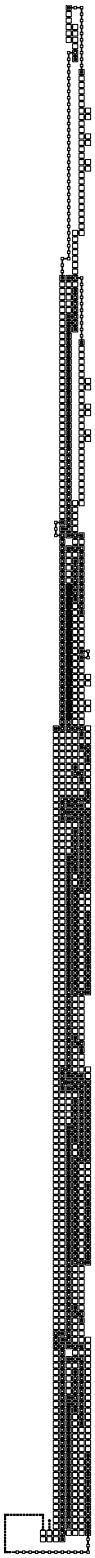
  - Create `Second_Warp`($\langle \texttt{SecondWarp}, i, u, op \rangle$, $\langle \texttt{SecondWarp}, i, u, op \rangle$, $\langle \texttt{PostWarp}, i, u, op \rangle$)
    from the single tile gadget, shown in Figure 8a if $i = 1$ or Figure 8b if $i = 2$, otherwise from Figure 8c if $i = 3$.

  - Create `Second_Warp`($\langle \texttt{SecondWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{SecondWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{PostWarp}, 2, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the single tile gadget shown in Figure 8f.

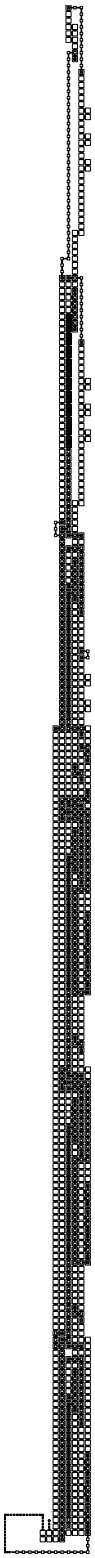
  - Create `Second_Warp`($\langle \texttt{SecondWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{SecondWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$,
    $\langle \texttt{PostWarp}, 3, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the single tile gadget shown in Figure 8h.



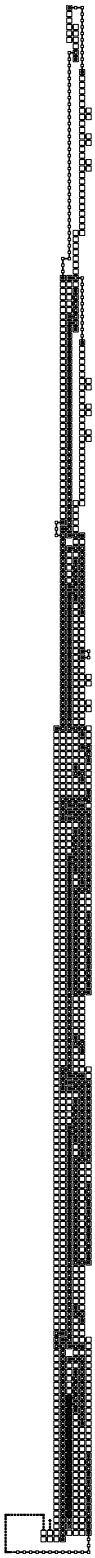
(a) Digit 1 - general overview.



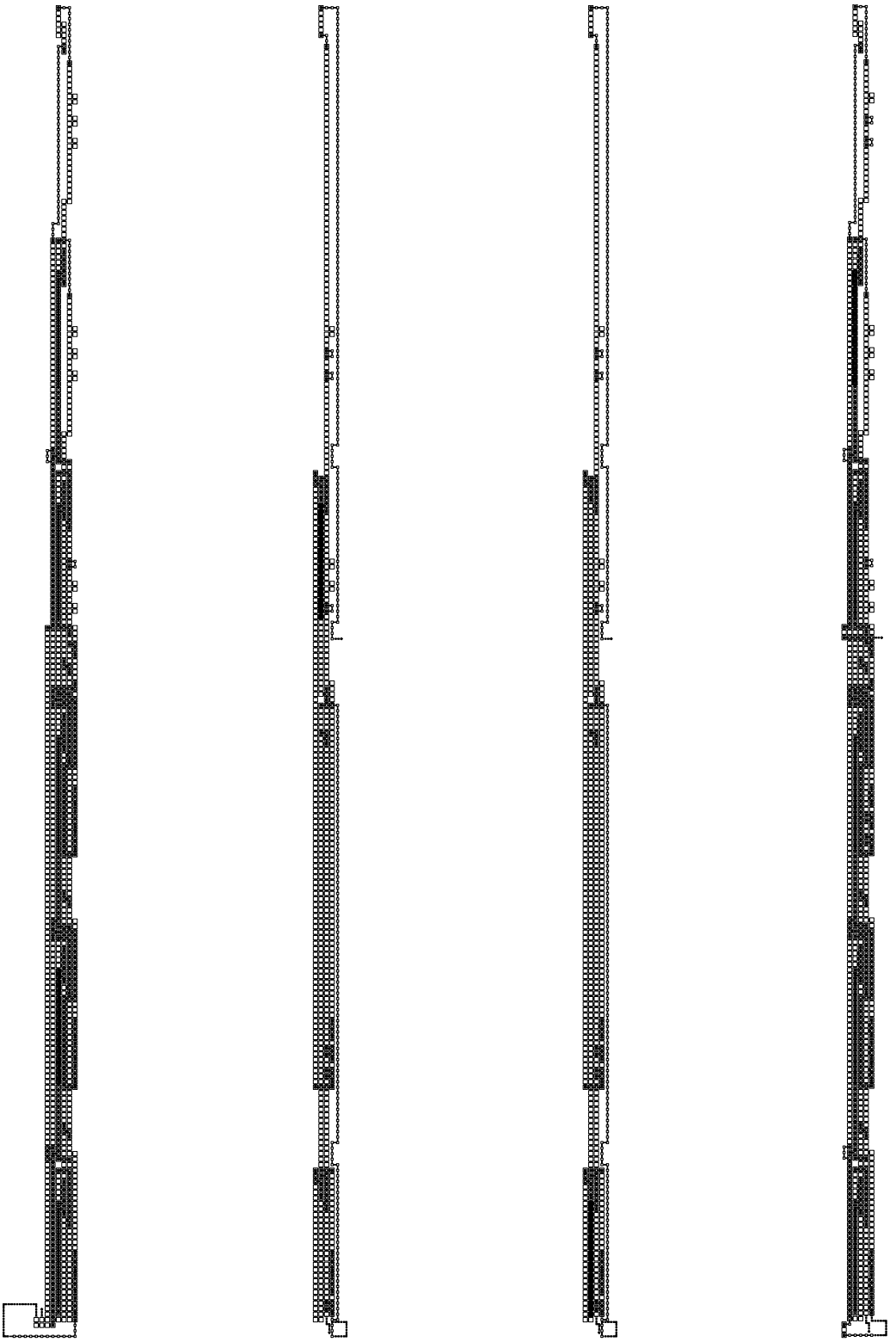
(b) Digit 2 - general overview.



(c) Digit 3 - general overview.



(d) Digit 2 - general (seed) overview.



(e) Digit 3 - general (seed) overview.

(f) Digit 2 - case 2 overview.

(g) Digit 2 - case 2 (seed) overview.

(h) Digit 3 - case 3 overview.

Figure 8: The `Second_Warp` gadget overviews.

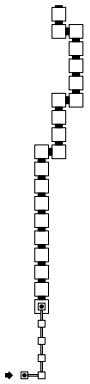
- `Post_Warp`: The `Post_Warp` gadget is final gadget to assemble in the `Warp_Unit`. The idea of this gadget is to assemble from wherever the last warping tiles "wake up", forming a path that ends where the next digit needs to start being written.

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

  - Create
    `Post_Warp`($\langle \texttt{PostWarp}, i, u, op \rangle$, $\langle \texttt{CounterWrite}, i, u, op \rangle$)
    from the general gadget shown in Figure 9a if $i = 1$, or Figure 9c if $i = 2$ or $i = 3$.

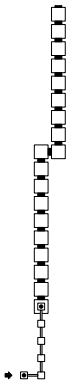
  - Create `Post_Warp`($\langle \texttt{PostWarp}, 1, u, op, \texttt{msr} \rangle$, $\langle \texttt{CounterWrite}, 1, u, op, \texttt{msr} \rangle$)
    from the general gadget in Figure 9j.

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

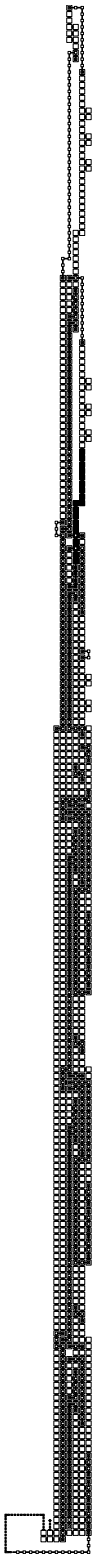


(a) Digit 1 - general.

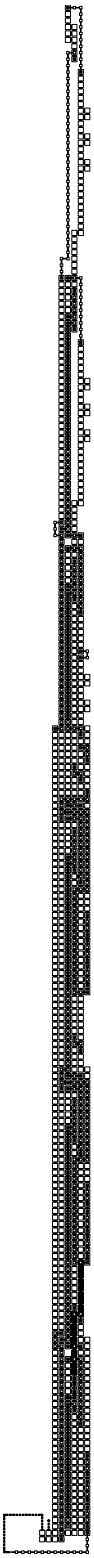
(b) Digit 1 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



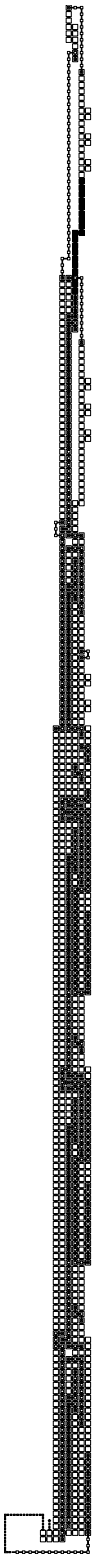
(c) Digit 2 & 3 - general.



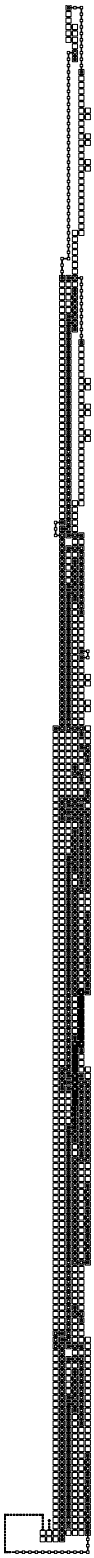
(d) Digit 2 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure c.



(e) Digit 2 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure c.



(f) Digit 3 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure c.

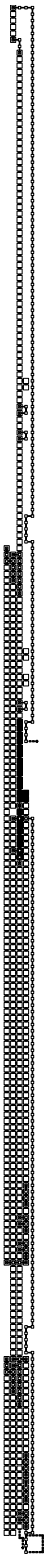


(g) Digit 3 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure c.

(h) Digit 1 - case 1.

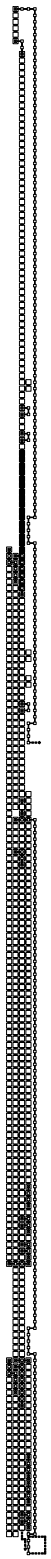
(i) Digit 1 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure h.

(j) Digit 1 - case 2.

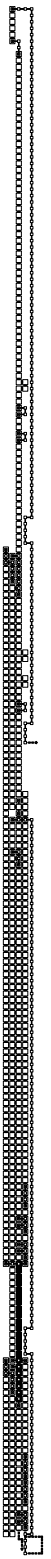


(k) Digit 1 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure j.

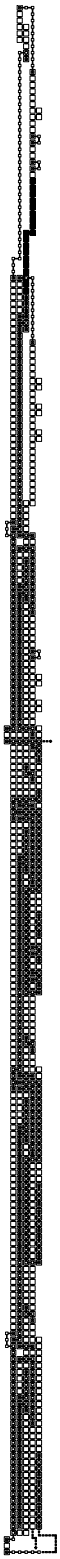
(l) Digit 2 - case 2.



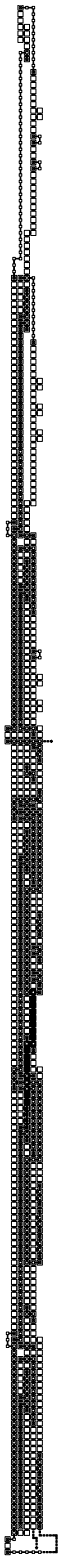
(m) Digit 2 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure l.



(n) Digit 2 - case 2 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure l.



(o) Digit 3 - case 3 overview. The black tiles in this figure correspond to the gadget shown in subfigure c.

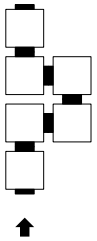


(p) Digit 3 - case 3 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure c.

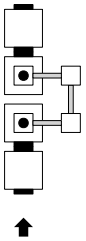
Figure 9: The `Post_Warp` gadgets.

4.1.4 Counter_Write

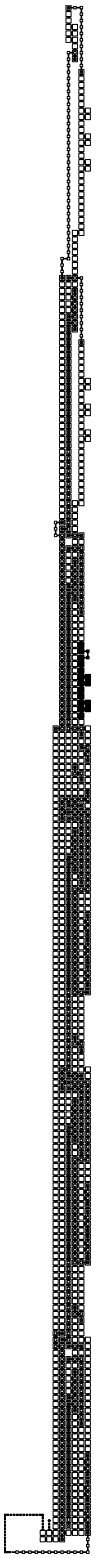
- For each $i = 1, 2, 3$, $j = l - 1, \ldots, 1$, $u \in \{0,1\}^j$, and each $op \in \{\texttt{increment}, \texttt{copy}, \texttt{halt}\}$:
  - Create Counter_Write($\langle \texttt{CounterWrite}, i, u0, op \rangle$, $\langle \texttt{CounterWrite}, i, u, op \rangle$) from the general gadget in Figure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



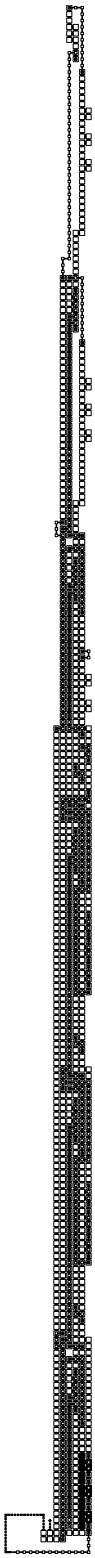
(a) `Counter_Write_0`



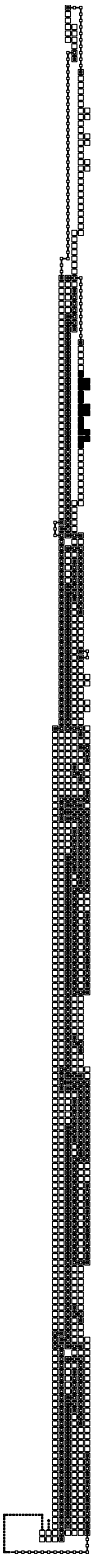
(b) `Counter_Write_1`



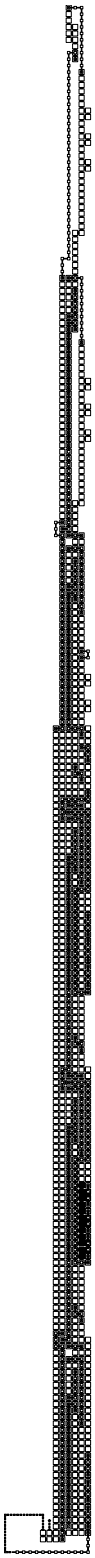
(c) Digit 1 - general overview.



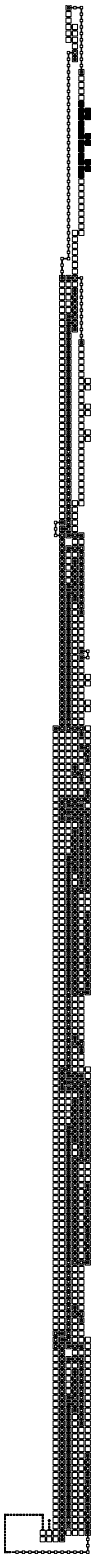
(d) Digit 1 - general (seed) overview.



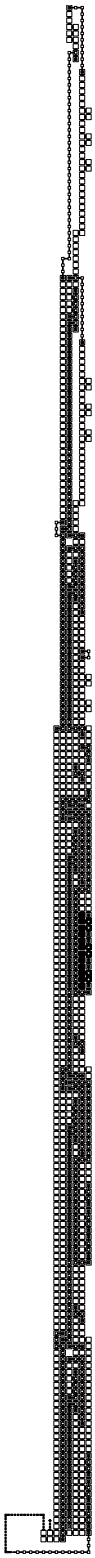
(e) Digit 2 - general overview.



(f) Digit 2 - general (seed) overview.

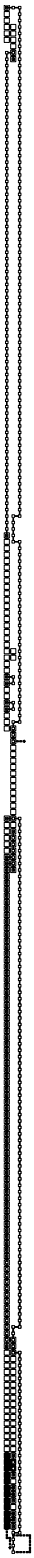


(g) Digit 3 - general overview.

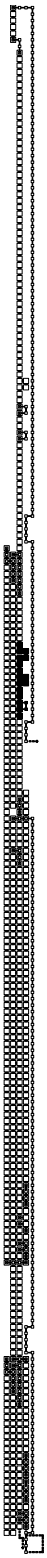


(h) Digit 3 - general (seed) overview.

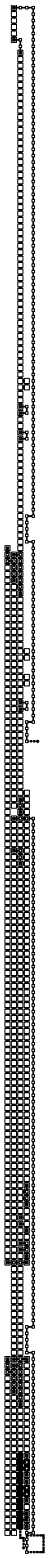
(i) Digit 1 - case 1.



(j) Digit 1 - case 1 (seed).



(k) Digit 1 - case 2.



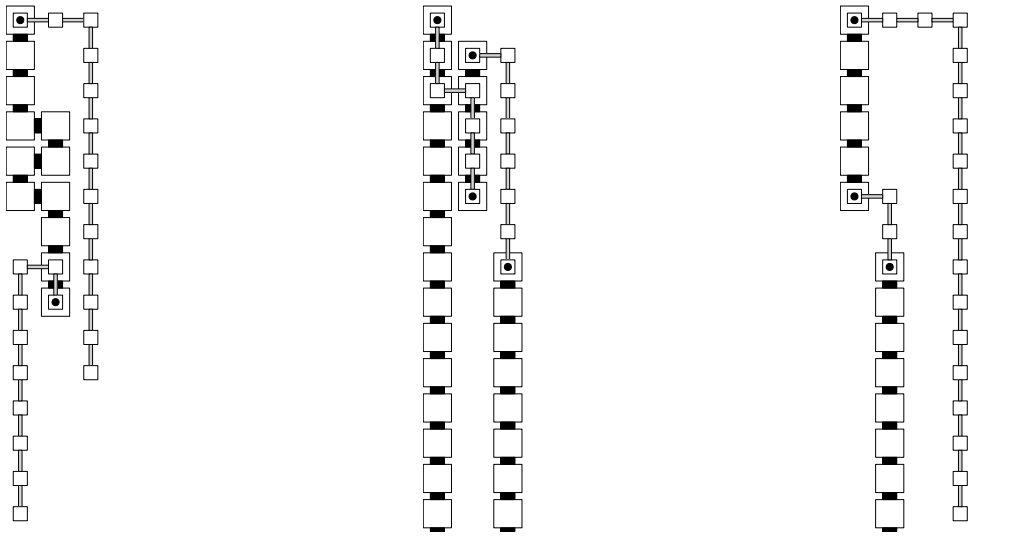
(l) Digit 1 - case 2 (seed).

(m) Digit 2 - case 2. (n) Digit 2 - case 2 (seed). (o) Digit 3 - case 3. (p) Digit 3 - case 3 (seed).

Figure 10: The `Counter_Write` gadgets.

4.1.5 Digit_Top

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($\langle \texttt{DigitTop}, 1, op \rangle$, $\langle \texttt{DigitTopA}, 1, op \rangle$) from the micro-gadget shown in Figure 3a.
  - Create Topper($\langle \texttt{DigitTopA}, 1, op \rangle$, $\langle \texttt{DigitTopB}, 1, op \rangle$) from the micro-gadget shown in Figure 11a.
  - Create South_Line4l($\langle \texttt{DigitTopB}, 1, op \rangle$, $\langle \texttt{Return\_Path}, 1, op \rangle$) from the micro-gadget shown in Figure 3b.

- Digit 1 (MSR): the following statements create the gadget shown in Figure 12k.
  - Create Topper($\langle \texttt{DigitTop}, 1, op, \texttt{msr} \rangle$, $\langle \texttt{DigitTopA}, 1, op, \texttt{msr} \rangle$) from the micro-gadget shown in Figure 11b.
  - Create South_Line4l($\langle \texttt{DigitTopA}, 1, op, \texttt{msr} \rangle$, $\langle \texttt{Return\_Path}, 1, op, \texttt{msr} \rangle$) from the micro-gadget shown in Figure 3b.

- Digit 1 (MSD): the following statements create the gadget shown in Figure 12h.

- Create North_Line4l(⟨DigitTop, 1, op, msr, msd⟩ , ⟨DigitTopA, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3a.
- Create North_Line4(⟨DigitTopA, 1, op, msr, msd⟩ , ⟨DigitTopB, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3a.
- Create Topper(⟨DigitTopB, 1, op, msr, msd⟩ , ⟨DigitTopC, 1, op, msr, msd⟩)
  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⟩ , ⟨Return_Path, 1, op, msr, msd⟩)
  from the micro-gadget shown in Figure 3b.

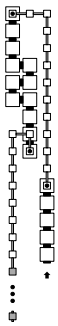
- Digit 2 (general): the following statements create the gadget shown in Figure 12a.
  - Create North_Line5(⟨DigitTop, 2, op⟩ , ⟨DigitTopA2, op⟩)
    from the micro-gadget shown in Figure 3a.
  - Create Topper(⟨DigitTopA2, op⟩ , ⟨DigitTopB2, op⟩)
    from the micro-gadget shown in Figure 11a.
  - Create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

- Digit 2 (MSD): the following statements create the gadget shown in Figure 12n.
  - 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⟩ , ⟨Return_Path, 2, op, msr, msd⟩)
    from the micro-gadget shown in Figure 3b.

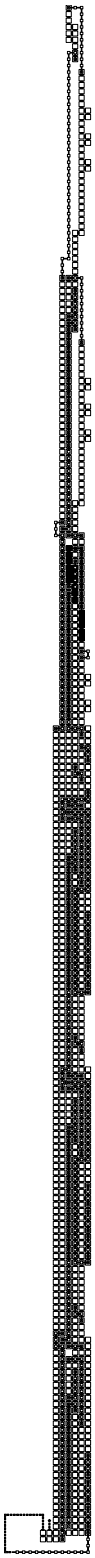
- Digit 3 (general): the following statements create the gadget from Figure 12a.
  - Create North_Line5(⟨DigitTop, 3, op⟩ , ⟨DigitTopA, 3, op⟩)
    from the micro-gadget shown in Figure 3a.

- Create `Topper`(⟨`DigitTopA`, 3, *op*⟩ , ⟨`DigitTopB`, 3, *op*⟩)
  from the micro-gadget shown in Figure 11a.
- Create `South_Line4`*l*(⟨`DigitTopB`, 3, *op*⟩ , ⟨`Return_Path`, 3, *op*⟩)
  from the micro-gadget shown in Figure 3b.

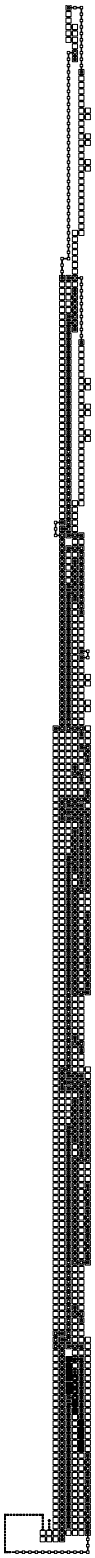
- Digit 3 (MSD): the following statements create the gadget from Figure 12a.
  - Create `North_Line5`(⟨`DigitTop`, 3, *op*, `msr`, `msd`⟩ , ⟨`DigitTopA`, 3, *op*, `msr`, `msd`⟩)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`(⟨`DigitTopA`, 3, *op*, `msr`, `msd`⟩ , ⟨`DigitTopB`, 3, *op*,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



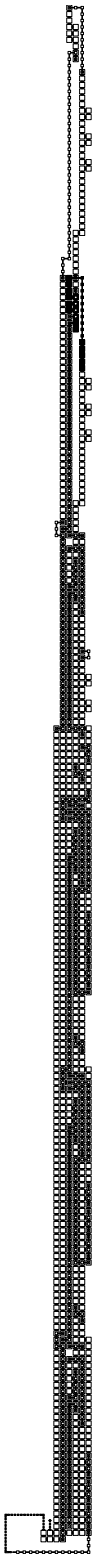
(a) Digits 1, 2, & 3 - general.



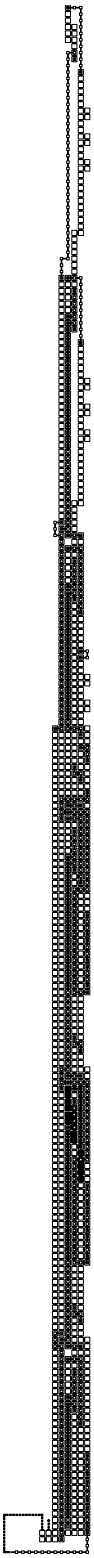
(b) Digit 1 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



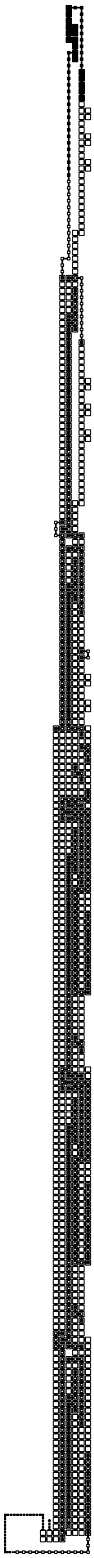
(c) Digit 1 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



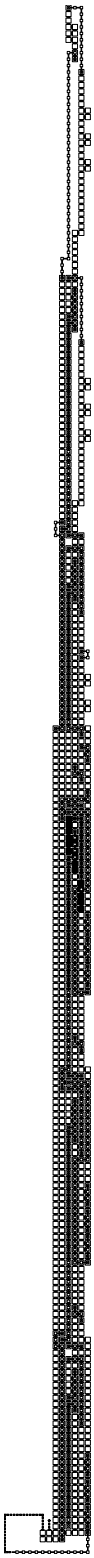
(d) Digit 2 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



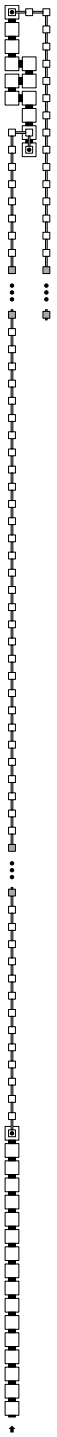
(e) Digit 2 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure a.



(f) Digit 3 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

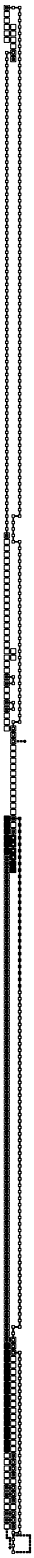


(g) Digit 3 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

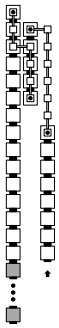


(h) Digit 1 - case 1.

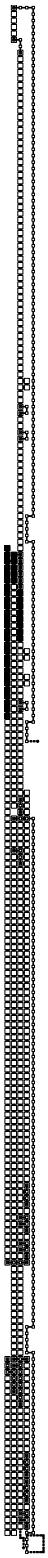
(i) Digit 1 - case 1 overview. The black tiles in this figure correspond to the gadget shown in subfigure h.



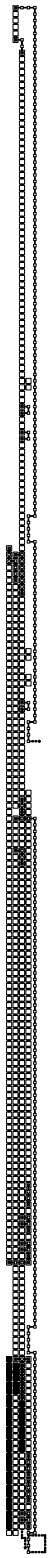
(j) Digit 1 - case 1 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure h.



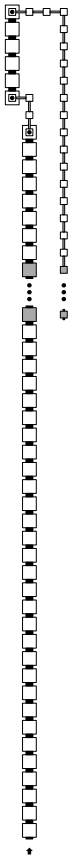
(k) Digit 1 - case 2.



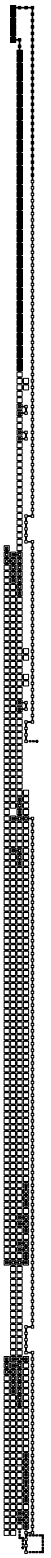
(l) Digit 1 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure k.



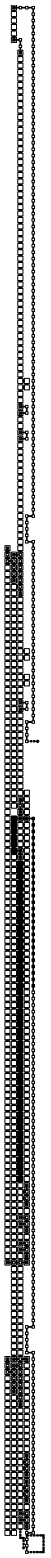
(m) Digit 1 - case 2 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure k.



(n) Digit 2 - case 2.



(o) Digit 2 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure n.



(p) Digit 2 - case 2 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure n.

(q) Digit 3 - case 3.

(r) Digit 3 - case 3 overview. The black tiles in this figure correspond to the gadget shown in subfigure q.

(s) Digit 3 - case 3 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure q.

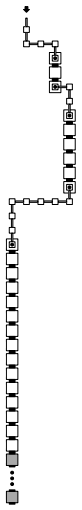
Figure 12: The `Digit_Top` gadgets.

4.1.6 `Return_Path`

After a `Digit_Top` has assembled, we know that the geometry the allows for the `Warp_Unit` to work has been placed, therfore the counter is free to return back near where it started reading the previous digit. The next gadget that assembles is the `Return_Path` gadget. The basic idea of this gadget is simply to provide a path from the end of `Digit_Top` to some general area closer to where the most recently read digit is located. Once the `Return_Path` has completely assembled, it output's a glue for the `Next_Read` gadgets to determine where the counter needs to assemble next.

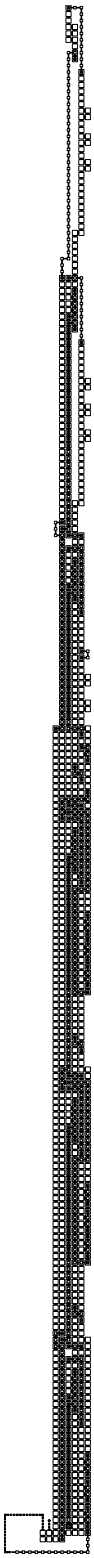
For each $op \in \{\texttt{increment}, \texttt{copy}\}$:

- Create `Return_Path`($\langle \texttt{ReturnPath}, 1, op \rangle$, $\langle \texttt{NextRead}, 1, op \rangle$)
  from the general gadget shown in Figure 13a.
- Create `Return_Path`($\langle \texttt{ReturnPath}, 1, op, \texttt{msr} \rangle$, $\langle \texttt{NextRead}, 1, op, \texttt{msr} \rangle$)
  from the general gadget shown in Figure 13j
- Create `Return_Path`($\langle \texttt{ReturnPath}, 1, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{NextRead}, 1, op, \texttt{msr}, \texttt{msd} \rangle$)
  from the general gadget shown in Figure 13m.
- Create `Return_Path`($\langle \texttt{ReturnPath}, 2, op \rangle$, $\langle \texttt{NextRead}, 2, op \rangle$)
  from the general gadget shown in Figure 13d.
- Create `Return_Path`($\langle \texttt{ReturnPath}, 2, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{NextRead}, 2, op, \texttt{msr}, \texttt{msd} \rangle$)
  from the general gadget shown in Figure 13m.
- Create `Return_Path`($\langle \texttt{ReturnPath}, 3, op \rangle$, $\langle \texttt{NextRead}, 3, op \rangle$)
  from the general gadget shown in Figure 13g.
- Create `Return_Path`($\langle \texttt{ReturnPath}, 3, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{NextRead}, 3, op, \texttt{msr}, \texttt{msd} \rangle$)
  from the general gadget shown in Figure 13g.



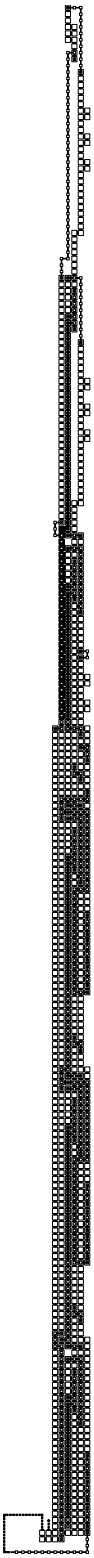
(a) Digit 1 - general.

(b) Digit 1 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

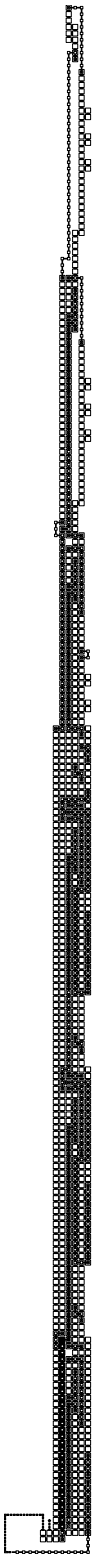


(c) Digit 1 - general (seed) overview. The black tile in this figure is a single tile gadget used only in the initial value.

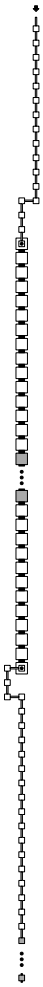
(d) Digit 2 - general.



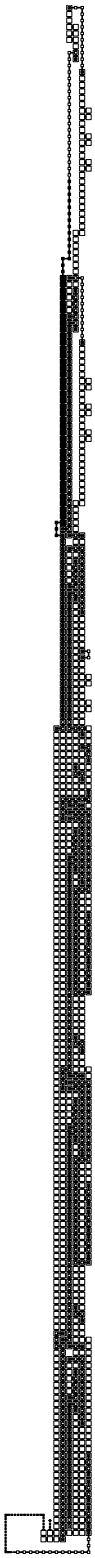
(e) Digit 2 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure d.



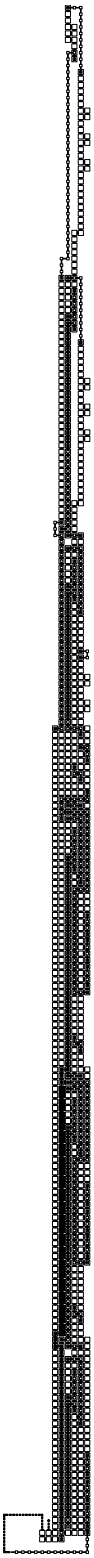
(f) Digit 2 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure d.



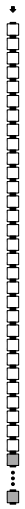
(g) Digit 3 - general.



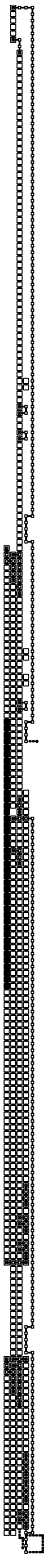
(h) Digit 3 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure g.



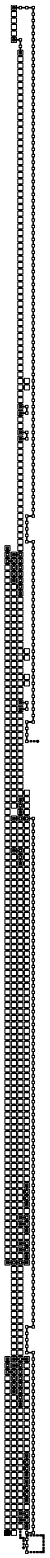
(i) Digit 3 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure g.



(j) Digit 1 - case 2.

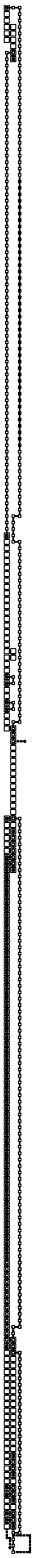


(k) Digit 1 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure j.

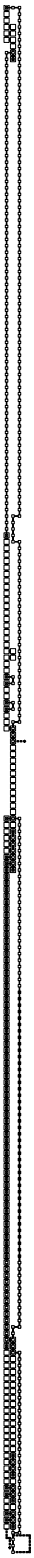


(l) Digit 1 - case 2 (seed) overview. The black tile in this figure is a single tile gadget used only in the initial value.

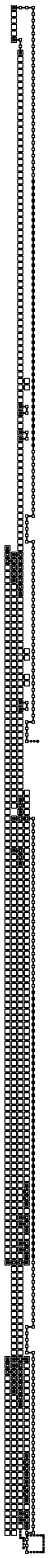
(m) Digit 1 - case 1, Digit 2 case 2.



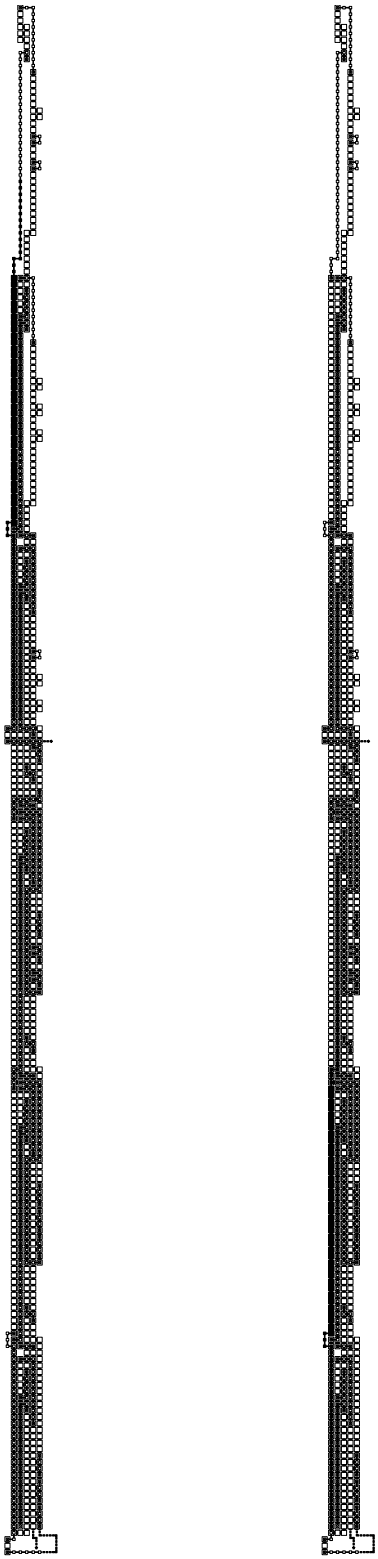
(n) Digit 1 - case 1 overview. The black tiles in this figure correspond to the gadget shown in subfigure m.



(o) Digit 1 - case 1 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure m.



(p) Digit 2 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure m.



(q) Digit 3 - case 3 overview. The black tiles in this figure correspond to the gadget shown in subfigure g.

(r) Digit 3 - case 3 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure g.

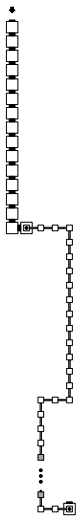
Figure 13: The `Return_Path` gadgets.

4.1.7 Next_Read

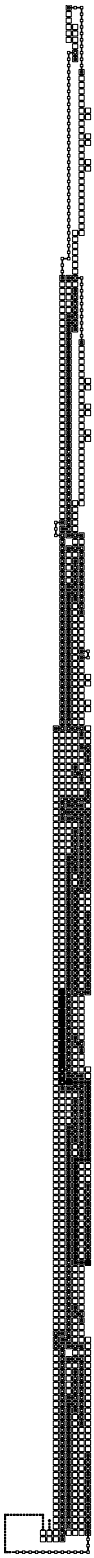
Once the Return_Path gadget has assembled, the counter has a few options to choose from. The gadget that controls what happens after a digit is written is the Next_Read gadget. If there is a msr and msd signal in the input, this gadget knows that the most significant digit was just read and will output a glue for the Cross_Next_Row gadget to assemble so that the counter crosses back to the right and begins reading the first digit in the next row. Otherwise (for regular digits not in the MSR), this gadget will assemble the second half of the return path, terminating at the next digit in the current row. When this happens, the gadget increments the digit index (unless it is already digit 3, in which case it resets to 1) and outputs an empty Counter_Read signal to force the counter to begin reading the next digit.

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

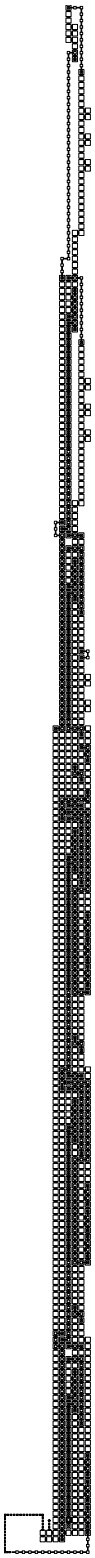
- Create Next_Read($\langle \texttt{NextRead}, 1, op \rangle$, $\langle \texttt{CounterRead}, 2, \lambda, op \rangle$)
  from the gadget shown in Figure 14a.
- Create Next_Read($\langle \texttt{NextRead}, 1, op, \texttt{msr} \rangle$, $\langle \texttt{CounterRead}, 2, \lambda, op \rangle$)
  from the gadget shown in Figure 14o.
- Create Next_Read($\langle \texttt{NextRead}, 1, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{CrossNextRow}, op \rangle$)
  from the gadget shown in Figure 14k.
- Create Next_Read($\langle \texttt{NextRead}, 2, op \rangle$, $\langle \texttt{CounterRead}, 3, \lambda, op \rangle$)
  from the gadget shown in Figure 14a.
- Create Next_Read($\langle \texttt{NextRead}, 2, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{CrossNextRow}, op \rangle$)
  from the gadget shown in Figure 14k.
- Create Next_Read($\langle \texttt{NextRead}, 3, op \rangle$, $\langle \texttt{CounterRead}, 1, \lambda, op \rangle$)
  from the gadget shown in Figure 14e.
- Create Next_Read($\langle \texttt{NextRead}, 3, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{CrossNextRow}, op \rangle$)
  from the gadget shown in Figure 14q.



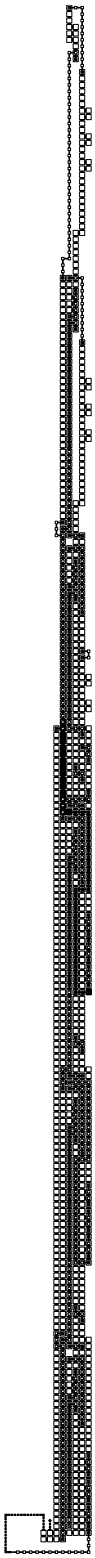
(a) Digit 1 & 2 - general.



(b) Digit 1 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

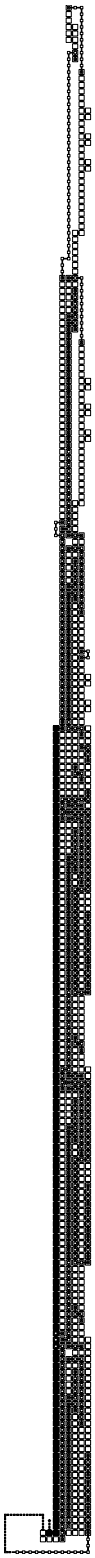


(c) Digit 1 - general (seed) overview. The black tile in this figure is a single tile gadget used only in the initial value.



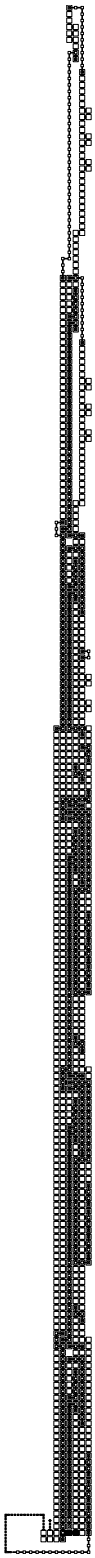
(d) Digit 2 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure a.

(e) Digit 3 - general.



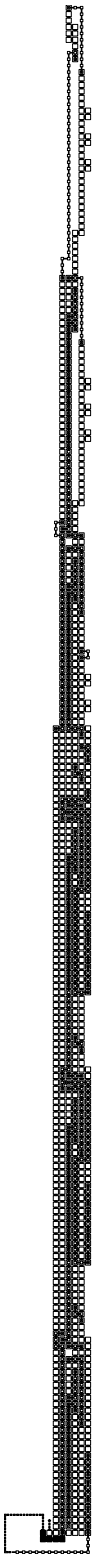
(f) Digit 3 - general overview. The black tiles in this figure correspond to the gadget shown in subfigure e.

(g) Digit 2 - seed.



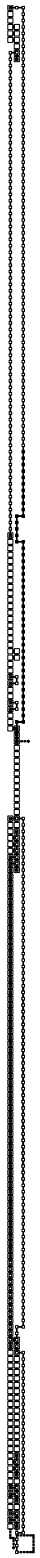
(h) Digit 2 - seed overview. The black tiles in this figure correspond to the gadget shown in subfigure g.

(i) Digit 3 - seed.

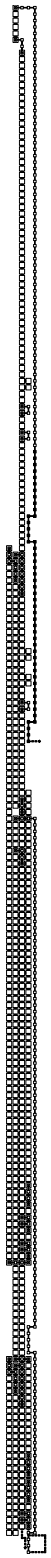


(j) Digit 3 - general (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure i.

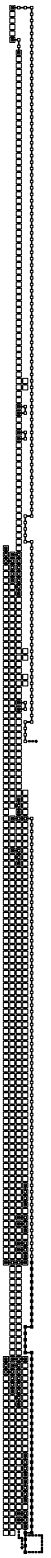
(k) Digit 1 - case 1, Digit 2 - case 2.



(l) Digit 1 - case 1 overview. The black tiles in this figure correspond to the gadget shown in subfigure k.

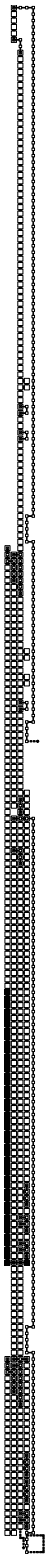


(m) Digit 2 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure k.



(n) Digit 2 - case 2 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure k.

(o) Digit 1 - case 2.



(p) Digit 1 - case 2 overview. The black tiles in this figure correspond to the gadget shown in subfigure o.

(q) Digit 3 - case 3.

(r) Digit 3 - case 3 overview. The black tiles in this figure correspond to the gadget shown in subfigure q.

(s) Digit 3 - case 3 (seed) overview. The black tiles in this figure correspond to the gadget shown in subfigure q.

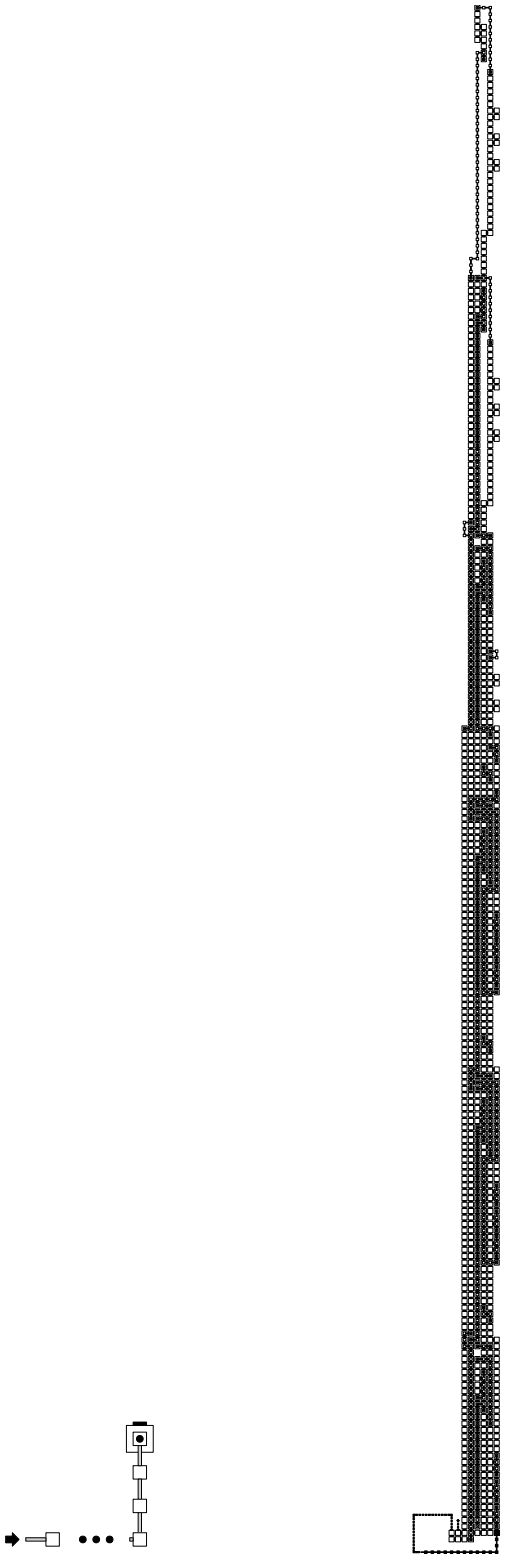
Figure 14: The `Next_Read` gadgets.

4.1.8 `Cross_Next_Row`

The idea this gadget is to assemble after reading the MSD, routing the counter 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 `Cross_Next_Row`($\langle \texttt{CrossNextRow}, op \rangle$, $\langle \texttt{CounterRead}, 1, \lambda, op \rangle$)
  from the gadget shown in Figure 15a.



(a) General.

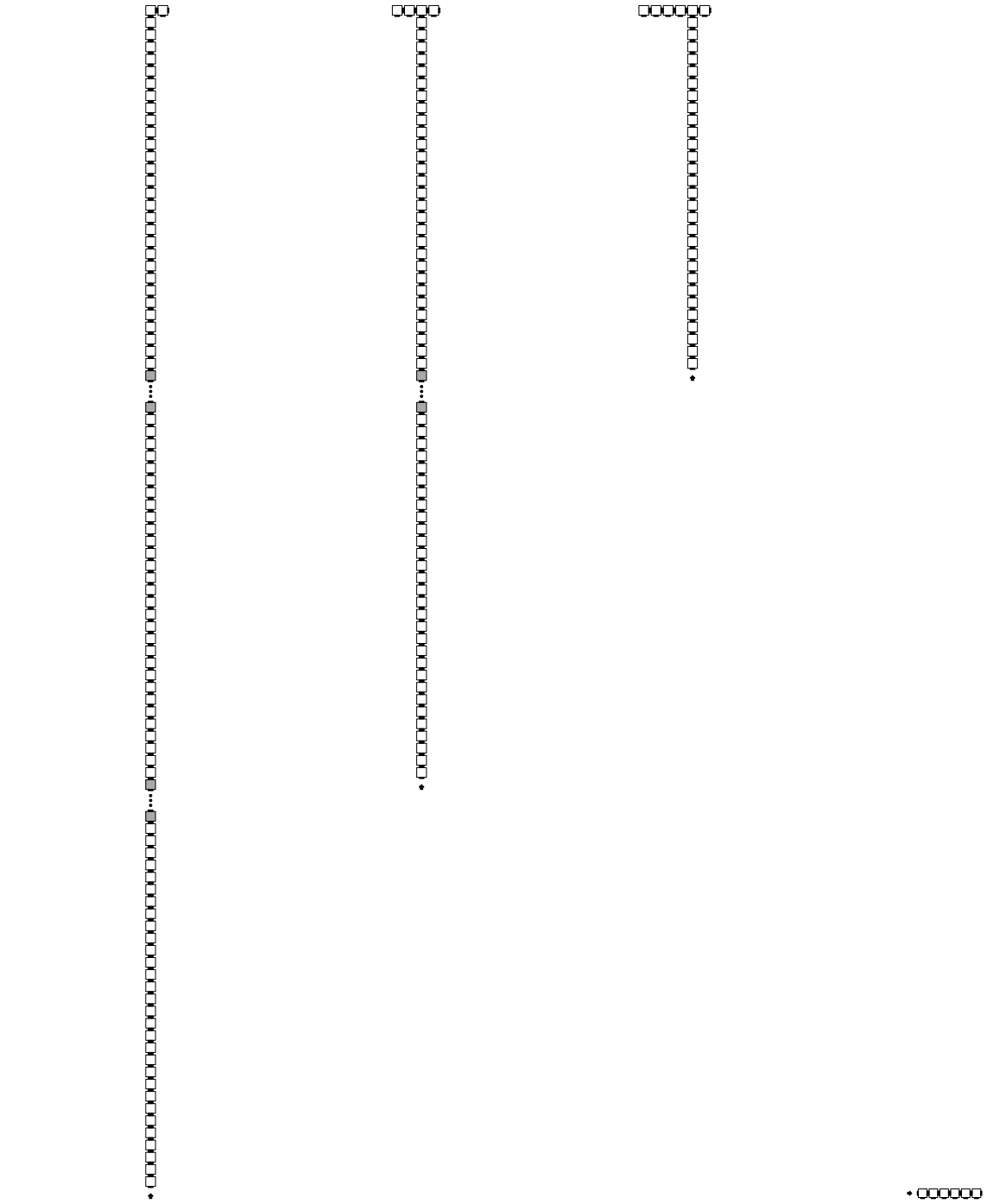
(b) General overview. The black tiles in this figure is the gadget shown in subfigure a.

Figure 15: The `Cross_Next_Row` gadget.

4.1.9 `Roof_Unit`

The `Roof_Unit` gadgets consist of two parts, the scaffolding and the shingles. The scaffolding is responsible for assembling a path from the top of the final last digit and adding a height of 15 tiles to the current (final) counter row. Since the most significant digit is positioned at a different height within a digit region depending on how many digits are in the MSR, the `Roof_Scaffolding` gadget's height must account for this. If the MSR has 3 digits, the height is only 15 tiles. If the MSR has 2 digits, the height is $30 + 12l + 15$ tiles. Otherwise, if the MSR has 1 digit the height is $2(30 + 12l) + 15$ tiles.

- If case 1: create `Roof_Scaffolding`($\langle$`RoofScaffolding`, 1, `halt`, `msr`, `msd`$\rangle$, $\langle$`RoofShingle`, 0$\rangle$) from the gadget shown in Figure 16a.
- If case 2: create `Roof_Scaffolding`($\langle$`RoofScaffolding`, 2, `halt`, `msr`, `msd`$\rangle$, $\langle$`RoofShingle`, 0$\rangle$) from the gadget shown in Figure 16b.
- If case 3: create `Roof_Scaffolding`($\langle$`RoofScaffolding`, 3, `halt`, `msr`, `msd`$\rangle$, $\langle$`RoofShingle`, 0$\rangle$) from the gadget shown in Figure 16c.
- For each $i = 0, \ldots, \lceil \frac{d}{3} \rceil - 1$, create `Roof_Shingle`($\langle$`RoofShingle`, $i\rangle$, $\langle$`RoofShingle`, $i + 1\rangle$) from the gadget shown in Figure 16d.
- If k is odd, we add one additional tile to the rightmost shingle, with a west glue labeled $\left\langle \texttt{RoofShingle}, \lceil \frac{d}{3} \rceil \right\rangle$.



(a) Case 1 - roof unit scaffolding.

(b) Case 2 - roof unit scaffolding.

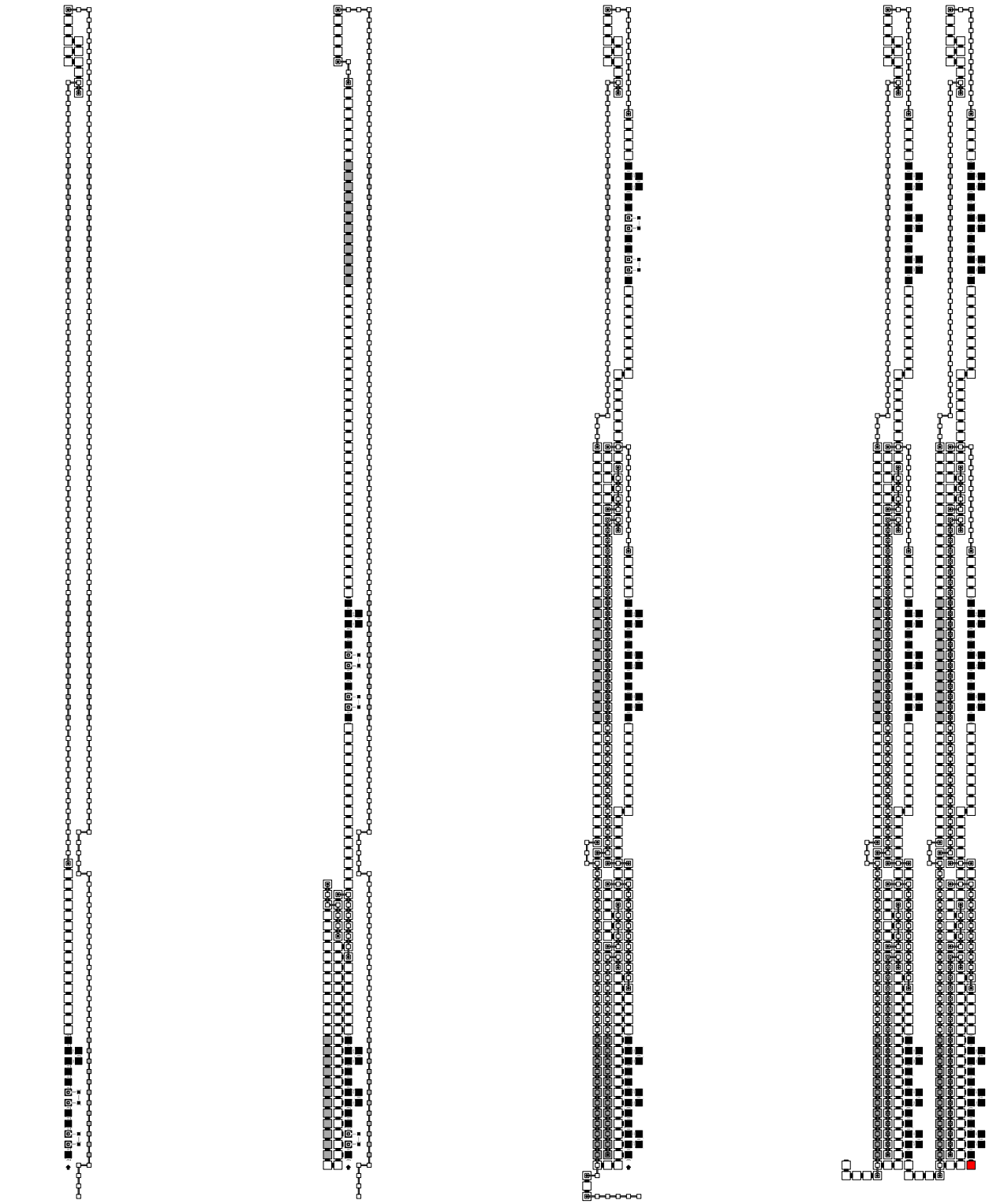
(c) Case 3 - roof unit scaffolding.

(d) Right shingles.

Figure 16: The `Roof_Unit` gadgets.

4.2 Initial value

We begin by encoding s 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 17: 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 assembly.

Note that we use i as the index of a digit in s and j as the index of a bit in a encoded digit.

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

The idea here is to repeat these steps starting from $i = 0$ and repeating until i is the index of the first digit in the MSR. These steps build general non-MSR digit regions shown in Figure 17d.

- `Start`.
- `Digit`, for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create `Counter_Write`($\langle$`CounterWrite`, 1, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 1, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 1, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 1, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$:
    create `Counter_Write`($\langle$`CounterWrite`, 1, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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_Path`($\langle$`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`, $i\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, 0\rangle$)
  from the general gadget show in Figure 9c.
- `Digit`: for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.

- if $1 < j < l - 1$:
  create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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_Path`($\langle$`ReturnPath`, 2, `seed`, $i\rangle$, $\langle$`NextRead`, 2, `seed`, $i\rangle$)
  from the gadget in Figure 13d.
- $i \leftarrow i + 1$.
- Create `Next_Read`($\langle$`NextRead`, 2, `seed`, $i - 1\rangle$, $\langle$`FirstWarp`, 3, `seed`, $i\rangle$)
  from the general gadget shown in Figure 14g.
- 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 7a.
- Create `Second_Warp`($\langle$`SecondWarp`, 3, `seed`, $i\rangle$, $\langle$`PostWarp`, 3, `seed`, $i\rangle$)
- Create `Post_Warp`($\langle$`PostWarp`, 3, `seed`, $i\rangle$, $\langle$`CounterWrite`, 3, `seed`, $i, 0\rangle$)
  from the general gadget shown in Figure 9c.
- `Digit`: for each $j = 0, \ldots, l - 1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create `Counter_Write`($\langle$`CounterWrite`, 3, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 3, `seed`, $i, j + 1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 3, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 3, `seed`, $i, j + 1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l - 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 3, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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_Line4l(⟨DigitTopB, 3, seed, i⟩ , ⟨ReturnPath, 3, seed, i⟩)
    from the micro-gadget shown in Figure 3b.

- Create Return_Path(⟨ReturnPath, 3, seed, i⟩ , ⟨NextRead, 3, seed, i, ⟩)
  from the gadget in Figure 13g.
- $i \leftarrow i + 1$.
- Create Next_Read(⟨NextRead, 3, seed, $i - 1$⟩ , ⟨CounterWrite, 1, seed, i⟩)
  from the general gadget shown in Figure 14i.
- 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 17a.

- Digit, for each $j = 0, \ldots, l - 1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create Counter_Write(⟨CounterWrite, 1, seed, i, j⟩ , ⟨CounterWrite, 1, seed, $i, j + 1$⟩)
    from the general gadget shown in Figure 10b.
  - if $j = 1$:
    create Counter_Write(⟨CounterWrite, 1, seed, i, j⟩ , ⟨CounterWrite, 1, seed, $i, j + 1$⟩)
    from the general gadget shown in Figure 10b.
  - if $1 < j < l - 1$:
    create Counter_Write(⟨CounterWrite, 1, seed, i, j⟩ , ⟨CounterWrite, 1, seed, $i, j + 1$⟩)
    from the general gadget shown in Figure 10a if $b = 0$ or Figure 10b if $b = 1$.
  - if $j = l - 1$: create Counter_Write(⟨CounterWrite, 1, seed, i, j⟩ , ⟨DigitTop, 1, seed, 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 12h
  - Create North_Line4l(⟨DigitTop, 1, seed, i⟩ , ⟨DigitTopA, 1, seed, i⟩)
    from the micro-gadget shown in Figure 3a.
  - Create North_Line4(⟨DigitTopA, 1, seed, i⟩ , ⟨DigitTopB, 1, seed, i⟩)
    from the micro-gadget shown in Figure 3a.
  - Create Topper(⟨DigitTopB, 1, seed, i⟩ , ⟨DigitTopC, 1, seed, i⟩)
    from the micro-gadget shown in Figure 11a.
  - Create South_Line4l(⟨DigitTopC, 1, seed, i⟩ , ⟨DigitTopD, 1, seed, i⟩)
    from the micro-gadget shown in Figure 3b.
  - Create South_Line30(⟨DigitTopD, 1, seed, i⟩ , ⟨DigitTopE, 1, seed, i⟩)
    from the micro-gadget shown in Figure 3b.
  - Create South_Line4l(⟨DigitTopE, 1, seed, i⟩ , ⟨DigitTopF, 1, seed, i⟩)
    from the micro-gadget shown in Figure 3b.
  - Create South_Line14(⟨DigitTopF, 1, seed, i⟩ , ⟨DigitTopG, 1, seed, i⟩)
    from the micro-gadget shown in Figure 3b.
  - Create South_Line17(⟨DigitTopG, 1, seed, i⟩ , ⟨ReturnPath, 1, seed, i⟩)
    from the micro-gadget shown in Figure 3b.

- Create `Return_Path`($\langle$`ReturnPath`, 1, `seed`, $i\rangle$, $\langle$`NextRead`, 1, `seed`, $i\rangle$)
  from the general gadget shown in Figure 13m
- Create `Next_Read`($\langle$`NextRead`, 1, `seed`, $i\rangle$, $\langle$`Cross_Next_Row`, `increment`$\rangle$)
  from the micro-gadget shown in Figure 14k.

Case 2 – if $d - i = 2$ to create the assembly shown in 17b.

- `Digit`, for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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 12k.
  - 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_Path`($\langle$`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`, $i\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, 0\rangle$)
  from the general gadget show in Figure 9l.
- `Digit`, for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $1 < j < l-1$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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 $b = 1$.

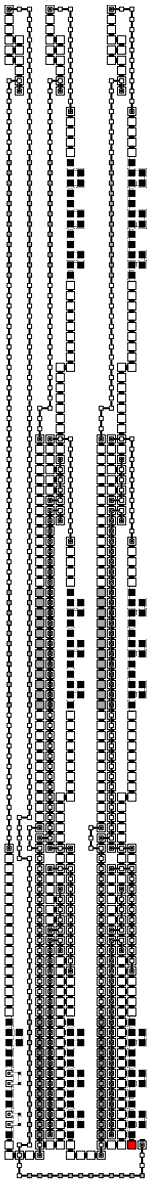
- `Digit_Top`, the following statements create the gadget shown in Figure 12n.
  - Create `North_Line4`l($\langle$`DigitTop`, 2, `seed`, $i\rangle$, $\langle$`DigitTopA`, 2, `seed`, $i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`DigitTopA`, 2, `seed`, $i\rangle$, $\langle$`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_Path`($\langle$`ReturnPath`, 2, `seed`, $i\rangle$, $\langle$`NextRead`, 2, `seed`, $i\rangle$)
  from the micro-gadget shown in Figure 13m.
- Create `Next_Read`($\langle$`NextRead`, 2, `seed`$\rangle$, $\langle$`Cross_Next_Row`, `increment`$\rangle$)
  from the micro-gadget shown in Figure 14k.

Case 3 – if $d - i = 3$ to create the assembly shown in 17c.

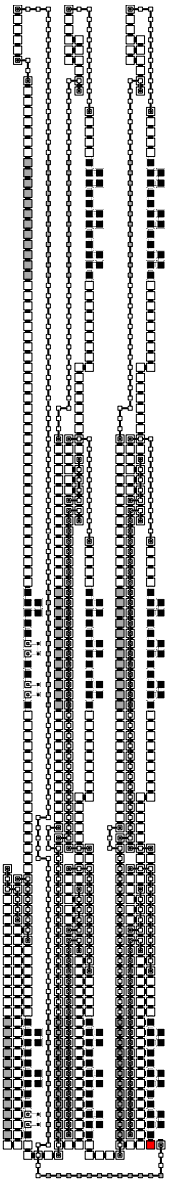
- `Digit`, for each $j = 0, \ldots, l - 1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create `Counter_Write`($\langle$`CounterWrite`, 1, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 1, `seed`, $i, j + 1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 1, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 1, `seed`, $i, j + 1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l - 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 1, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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_Path`($\langle$`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`, $i\rangle$) (single tile).
- Create `Post_Warp`($\langle$`PostWarp`, 2, `seed`, $i\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, 0\rangle$)
  from the general gadget show in Figure 9c.

- `Digit`, for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 2, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10a.
  - if $1 < j < l-1$:
    create `Counter_Write`($\langle$`CounterWrite`, 2, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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_Path`($\langle$`ReturnPath`, 2, `seed`, $i\rangle$, $\langle$`NextRead`, 2, `seed`, $i\rangle$)
  from the gadget in Figure 13d.
- $i \leftarrow i + 1$
- Create `Next_Read`($\langle$`NextRead`, 2, `seed`, $i-1\rangle$, $\langle$`FirstWarp`, 3, `seed`, $i\rangle$)
  from the general gadget shown in Figure 14g.
- 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 7a.
- Create `Second_Warp`($\langle$`SecondWarp`, 3, `seed`, $i\rangle$, $\langle$`PostWarp`, 3, `seed`, $i\rangle$)
- Create `Post_Warp`($\langle$`PostWarp`, 3, `seed`, $i\rangle$, $\langle$`CounterWrite`, 3, `seed`, $i, 0\rangle$)
  from the general gadget shown in Figure 9c.
- `Digit`, for each $j = 0, \ldots, l-1$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$:
    create `Counter_Write`($\langle$`CounterWrite`, 3, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $j = 1$:
    create `Counter_Write`($\langle$`CounterWrite`, 3, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 3, `seed`, $i, j+1\rangle$)
    from the general gadget shown in Figure 10b.
  - if $1 < j < l-1$:
    create `Counter_Write`($\langle$`CounterWrite`, 3, `seed`, $i, j\rangle$, $\langle$`CounterWrite`, 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`($\langle$`CounterWrite`, 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 $b = 1$.

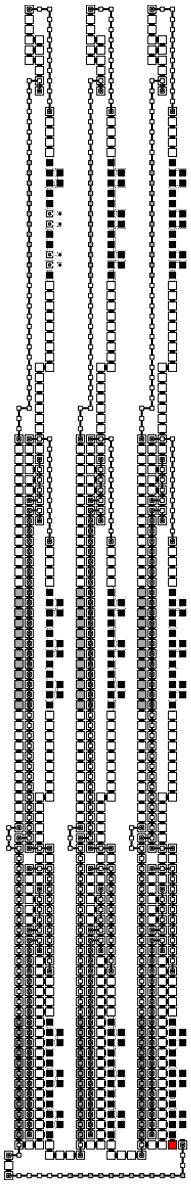
- `Digit_Top`, the following statements create the gadget shown in Figure 12a.
  - Create `North_Line5`(⟨`DigitTop`, 3, `seed`, i⟩ , ⟨`DigitTopA`, 3, `seed`, i⟩) from the micro-gadget shown in Figure 3a.
  - Create `Topper`(⟨`DigitTopA`, 3, `seed`, i⟩ , ⟨`DigitTopB`, 3, `seed`, i⟩) from the micro-gadget shown in Figure 11a.
  - Create `South_Line4`l(⟨`DigitTopB`, 3, `seed`, i⟩ , ⟨`ReturnPath`, 3, `seed`, i⟩) from the micro-gadget shown in Figure 3b.
- Create `Return_Path`(⟨`ReturnPath`, 3, `seed`, i⟩ , ⟨`NextRead`, 3, `increment`, `msr`, `msd`⟩) from the gadget in Figure 13g.



(a) Initial value case 1



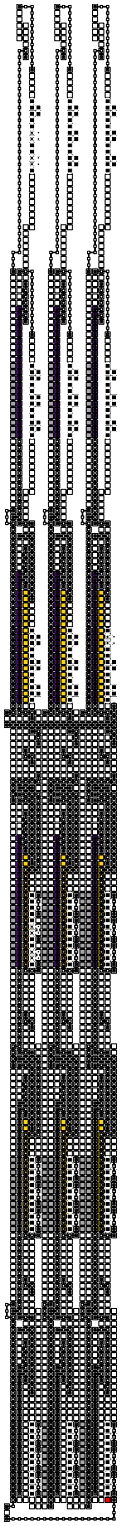
(b) Initial value case 2



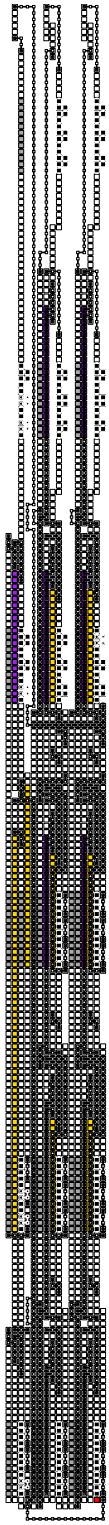
(c) Initial value case 3

Figure 18: These figures show a full example of an initial value, with both general regions and the MSRs together, instead of separated as shown above.

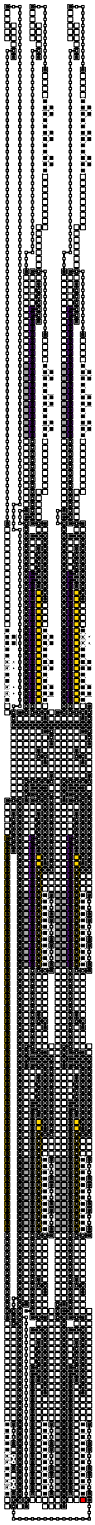
4.3 Overviews



(a) Full overview case 3



(b) Full overview case 2



(c) Full overview case 1