1 Definitions

1.1 Misc

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

$MSR =$ most signifcant digit region

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

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

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

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

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

1.2 Determining the starting value $\mathcal{C}_0$

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

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

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

Proof.

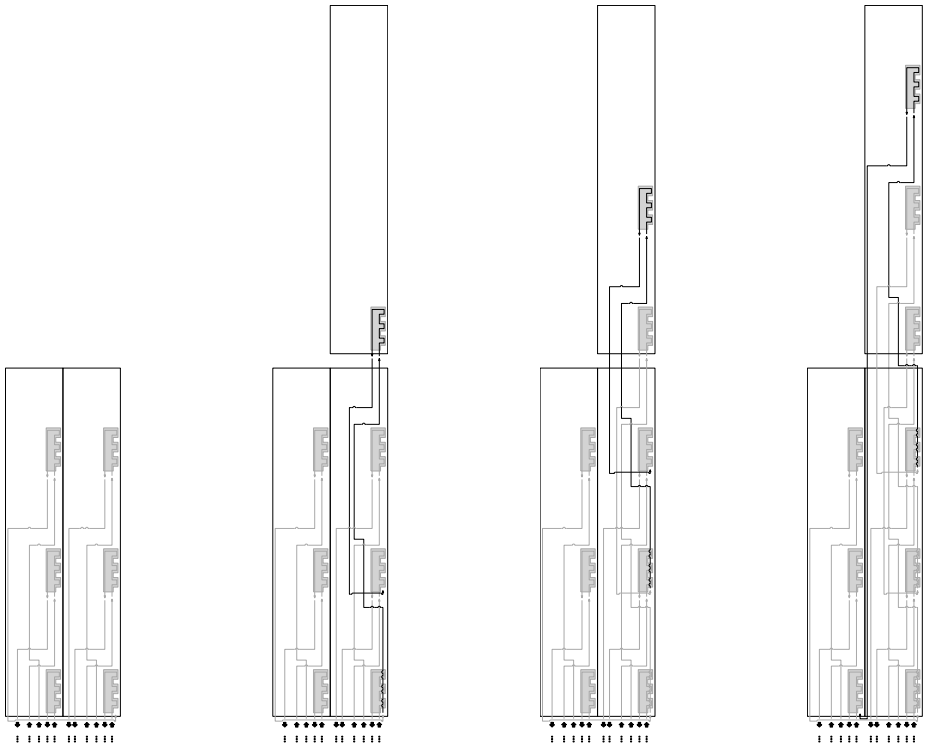
$$\begin{aligned} N = 102\left(\frac{N}{102}\right) &= 102\left(\left(\frac{N}{102}\right)^{\frac{1}{d}}\right)^d \leq 102\left\lceil\left(\frac{N}{102}\right)^{\frac{1}{d}}\right\rceil^d \\ &= 102m^d \leq 12lm^d + 90m^d \leq 12lm^d + 90m^d + 12l + 76 \\ &= m^d(12l + 90) + 12l + 76 \end{aligned}$$

□

1.3 Filling in the gaps

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

2 General counter



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

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

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

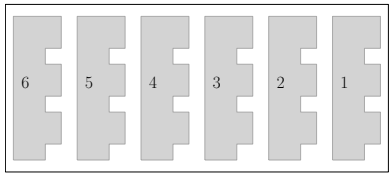
(d) Read digit 3 in the current row, write digit 3 in the next row.

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

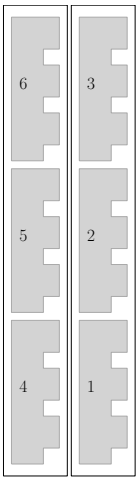
2.1 Digit region explanation (in progress)

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

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



(a) Digits in a typical counter



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

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

2.2 Detecting the edges

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

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

2.3 Tile set

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

2.3.1 Line Gadgets

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

Figure 3: `Line` gadgets

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

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

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

while $i < d$:

- if $i = 0$: create `SeedStart`($\langle$`SeedDigit1`$, i\rangle$)
- `Digit1` - for each $j = 0, \ldots, l$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Bit_Writer`($\langle$`SeedDigit1`$, i\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
  - if $0 \leqslant j \leqslant l$: create `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
  - if $j = l$: `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedDigitTop`$, i\rangle$)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 11a.
  - Create `North_Line5`($\langle$`SeedDigitTop`$, i\rangle$, $\langle$`SeedDigitTopA`$, i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`SeedDigitTopA`$, i\rangle$, $\langle$`SeedDigitTopB`$, i\rangle$)
    from the micro-gadget shown in Figure 10a.
  - Create `South_Line4`l($\langle$`SeedDigitTopB`$, i\rangle$, $\langle$`SeedWarpInitializer`$, i\rangle$)
    from the micro-gadget shown in Figure 3b.
- $i \leftarrow i + 1$
- Create `Seed_Warp_Initializer`($\langle$`SeedWarpInitializer`$, i-1\rangle$, $\langle$`SeedSecondWarp`$, i\rangle$)
- Create `Second_Warp`($\langle$`SeedSecondWarp`$, i\rangle$, $\langle$`SeedPostWarp`$, i\rangle$)
- Create `Post_Warp`($\langle$`SeedPostWarp`$, i\rangle$, $\langle$`SeedDigit2`$, i\rangle$)
  from the general gadget show in Figure 8b.
- `Digit2`: for each $j = 0, \ldots, l$ and each b in $bin(C_0[i])[j]$:
  - if $j = 0$: create `Bit_Writer`($\langle$`SeedDigit2`$, i\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
  - if $0 \leqslant j \leqslant l$: create `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedBit`$, i, j+1\rangle$)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
  - if $j = l$: `Bit_Writer`($\langle$`SeedBit`$, i, j\rangle$, $\langle$`SeedDigitTop2`$, i\rangle$)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
- `Digit_Top`: the following statements create the gadget shown in Figure 11a.
  - Create `North_Line5`($\langle$`SeedDigitTop`$, i\rangle$, $\langle$`SeedDigitTopA`$, i\rangle$)
    from the micro-gadget shown in Figure 3a.
  - Create `Topper`($\langle$`SeedDigitTopA`$, i\rangle$, $\langle$`SeedDigitTopB`$, i\rangle$)
    from the micro-gadget shown in Figure 10a.

  - Create South_Line4l(⟨SeedDigitTopB, i⟩ , ⟨SeedReturnPath, i⟩)
    from the micro-gadget shown in Figure 3b.

- Return_Path: create Return_From_Digit(⟨SeedReturnPath, i⟩ , ⟨SeedInternalBridge, i⟩)
  from the gadget in Figure 12a.

- $i \leftarrow i + 1$

- Create Seed_Internal_Bridge(⟨SeedInternalBridge, $i - 1$⟩ , ⟨SeedFirstWarp, i⟩)
  from the general gadget shown in Figure 15c.

- Create First_Warp(⟨SeedFirstWarp, i⟩ , ⟨SeedWarpBridge, i⟩)

- Create Warp_Bridge(⟨SeedWarpBridge, i⟩ , ⟨SeedSecondWarp, i⟩)
  from the general gadget shown in Figure 7a.

- Create Second_Warp(⟨SeedSecondWarp, i⟩ , ⟨SeedPostWarp, i⟩)

- Create Post_Warp(⟨SeedPostWarp, i⟩ , ⟨SeedDigit3⟩)
  from the general gadget shown in Figure 8b.

- Digit3: for each $j = 0, \ldots, l$ and each b in $bin(C_0[i])[j]$:

  - if $j = 0$: create Bit_Writer(⟨SeedDigit3, i⟩ , ⟨SeedBit, $i, j + 1$⟩)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
  - if $0 \leqslant j \leqslant l$: create Bit_Writer(⟨SeedBit, i, j⟩ , ⟨SeedBit, $i, j + 1$⟩)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.
  - if $j = l$: create Bit_Writer(⟨SeedBit, i, j⟩ , ⟨SeedDigitTop2, i⟩)
    from the general gadget shown in Figure 9a if $b = 0$ or Figure 9b if $b = 1$.

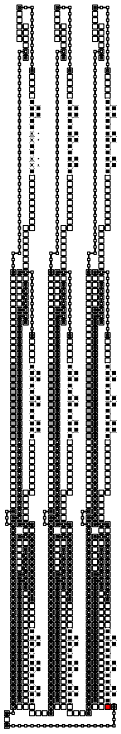
- Digit_Top: the following statements create the gadget shown in Figure 11a.

  - Create North_Line5(⟨SeedDigitTop, i⟩ , ⟨SeedDigitTopA, i⟩)
    from the micro-gadget shown in Figure 3a.
  - Create Topper(⟨SeedDigitTopA, i⟩ , ⟨SeedDigitTopB, i⟩)
    from the micro-gadget shown in Figure 10a.
  - Create South_Line4l(⟨SeedDigitTopB, i⟩ , ⟨SeedReturnPath, i⟩)
    from the micro-gadget shown in Figure 3b.

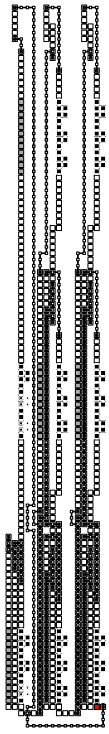
- Return_Path: Create Return_From_Digit(⟨SeedReturnPath, i⟩ , ⟨SeedRegionEnd, i⟩)
  from the gadget in Figure 13.

- if $i + 1 \neq d$: create Seed_External_Bridge(⟨SeedRegionEnd, i⟩ , ⟨SeedDigit, $i + 1$⟩)
  from the general gadget shown in Figure 15b.

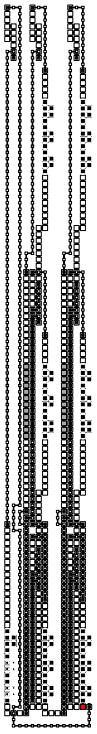
- $i \leftarrow i + 1$



(a) Initial value case 3



(b) Initial value case 2

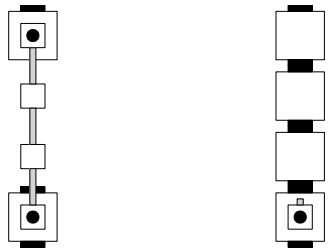


(c) Initial value case 1

2.4 Counter Unit

2.4.1 Digit readers

- For each $i = 1, 2, 3$, $j = l-1, \ldots, 1$, $u \in \{0,1\}^j$, and $op \in \{\texttt{increment}, \texttt{copy}\}$:
  - if $j = 0$: create $\texttt{Bit\_Reader}(\ \langle \texttt{DigitReader}, i, \lambda, op\rangle\ ,\ \langle \texttt{DigitReader}, i, 0, op\rangle\ ,\ \langle \texttt{DigitReader}, i, 1, op\rangle\ )$ from the general gadget in Figure 5
  - if $1 \leqslant j \leqslant l-2$: create $\texttt{Bit\_Reader}(\ \langle \texttt{DigitReader}, i, u, op\rangle\ ,\ \langle \texttt{DigitReader}, i, 0u, op\rangle\ ,\ \langle \texttt{DigitReader}, i, 1u, op\rangle\ )$ from the general gadget in Figure 5
  - if $j = l-1$: create $\texttt{Bit\_Reader}(\ \langle \texttt{DigitReader}, i, u, op\rangle\ ,\ \langle \texttt{PreWarp}, i, 0u, op\rangle\ ,\ \langle \texttt{PreWarp}, i, 1u, op\rangle\ )$ from the general gadget in Figure 5



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

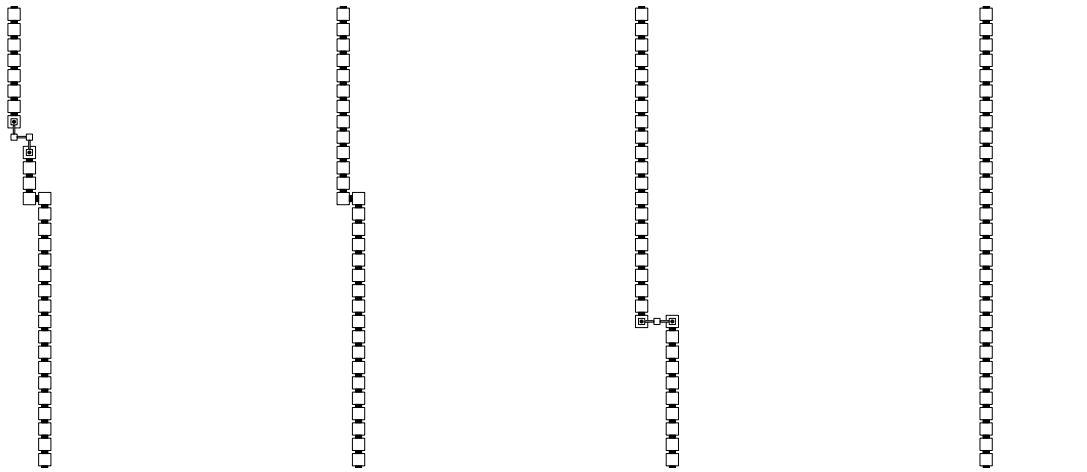
Figure 5: `Read` gadgets

2.4.2 Warping

We now explain the Warp units. 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 next digit region, so that the `Counter_Write` gadgets can write the next value in the correct location.

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

- `Pre_Warp`: These gadgets take the bits read from the `Bit_Reader` gadgets and convert it into a signal used until the `Digit_Top` gadgets are attached after writing the current digit. The signal started by this gadget is used to tell the counter whether to begin reading another digit in the current row, or cut across the rectangle and begin reading the first digit in the next row.
  - if u ends with 00: create `Pre_Warp`($\langle \texttt{PreWarp}, i, u, op \rangle$, $\langle \texttt{FirstWarp}, i, u, op \rangle$) from the general gadget in Figure 6a
  - if u ends with 01: create `Pre_Warp`($\langle \texttt{PreWarp}, i, u, op \rangle$, $\langle \texttt{FirstWarp}, i, u, op, \texttt{msr} \rangle$) from the general gadget in Figure 6c
  - if u ends with 11: create `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 6b if $i = 1$ (case 1), or Figure 6d if $i = 2$ (case 2), or Figure 6a if $i = 3$ (case 3).

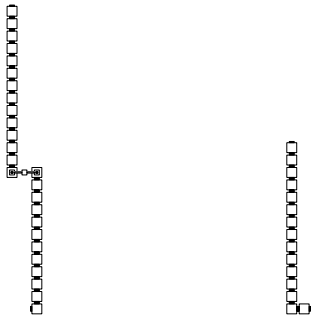


(a) General (b) Digit 1 – Case 1 (c) Digit 1 – Case 2 (d) Digit 2 – Case 2

Figure 6: `Pre_Warp` gadgets

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

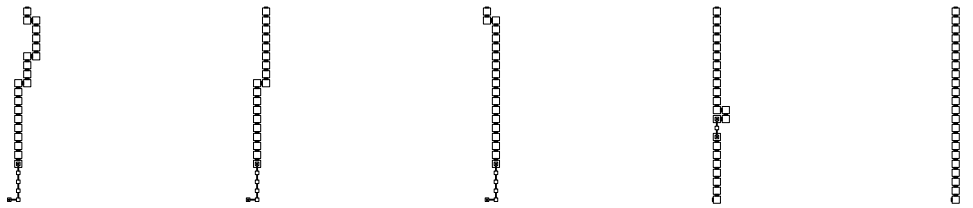
- `Warp_Bridge`: a `Warp_Bridge` gadget binds the last tile of the `First_Warp` gadgets to the first tile of the `Second_Warp` gadgets. For digit 1 in cases 1 and 2, the `Warp_Bridge` is omitted from the `Warp_Unit`.
  - if u ends with 00:
    create `Warp_Bridge`($\langle$`WarpBridge`$, i, u, op\rangle$, $\langle$`SecondWarp`$, i, u, op\rangle$)
    from the general gadget in Figure 7a
  - if u ends with 11 and i is 2:
    create `Warp_Bridge`($\langle$`WarpBridge`$, i, u, op,$ `msr`$,$ `msd`$\rangle$, $\langle$`SecondWarp`$, i, u, op,$ `msr`$,$ `msd`$\rangle$)
    from the general gadget in Figure 7b
  - if u ends with 11 and i is 3:
    create `Warp_Bridge`($\langle$`WarpBridge`$, i, u, op,$ `msr`$,$ `msd`$\rangle$, $\langle$`SecondWarp`$, i, u, op,$ `msr`$,$ `msd`$\rangle$)
    from the general gadget in Figure 7a



(a) General (b) Digit 2 – Case 2

Figure 7: `Warp_Bridge` gadgets

- `Second_Warp`:
  - Create `Second_Warp`($\langle$`SecondWarp`$, i, u, op\rangle$,
    $\langle$`SecondWarp`$, i, u, op\rangle$,
    $\langle$`PostWarp`$, i, u, op\rangle$)
  - Create `Second_Warp`($\langle$`SecondWarp`$, i, u, op,$`msr`$\rangle$,
    $\langle$`SecondWarp`$, i, u, op,$`msr`$\rangle$,
    $\langle$`PostWarp`$, i, u, op,$`msr`$\rangle$)
  - Create `Second_Warp`($\langle$`SecondWarp`$, i, u, op,$`msr`$,$`msd`$\rangle$,
    $\langle$`SecondWarp`$, i, u, op,$`msr`$,$`msd`$\rangle$,
    $\langle$`PostWarp`$, i, u, op,$`msr`$,$`msd`$\rangle$)
- `Post_Warp`:
  - if u ends with 00:
    create `Post_Warp`($\langle$`PostWarp`$, i, u, op\rangle$, $\langle$`DigitWriter`$, i, u, op\rangle$)
    from the general gadget shown in Figure 8a if $i = 1$, or Figure 8b if $i = 2$ or $i = 3$.
  - if u ends with 01:
    create `Post_Warp`($\langle$`PostWarp`$, i, u, op,$`msr`$\rangle$, $\langle$`DigitWriter`$, i, u, op,$`msr`$\rangle$)
    from the general gadget in Figure 8d.
  - if u ends with 11:
    create `Post_Warp`($\langle$`PostWarp`$, i, u, op,$`msr`$,$`msd`$\rangle$, $\langle$`DigitWriter`$, i, u, op,$`msr`$,$`msd`$\rangle$)
    from the general gadget shown in Figure 8c if $i = 1$, or Figure 8e if $i = 2$, or Figure 8b if $i = 3$.



(a) General Digit 1 (b) General Digits 2 and 3 (c) Digit 1 – Case 1 (d) Digit 1 – Case 2 (e) Digit 2 – Case 2

Figure 8: `Post_Warp` gadgets

2.4.3 Digit writers

- For each $i = 1, 2, 3$, $j = l - 1, \ldots, 1$, $u \in \{0,1\}^j$, and $op \in \{\texttt{increment}, \texttt{copy}\}$:
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, u0, op \rangle$, $\langle \texttt{DigitWriter}, i, u, op \rangle$)
    from the general gadget in Figure 9a
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, u1, op \rangle$, $\langle \texttt{DigitWriter}, i, u, op \rangle$)
    from the general gadget in Figure 9b
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, u0, op, \texttt{msr} \rangle$, $\langle \texttt{DigitWriter}, i, u, op, \texttt{msr} \rangle$)
    from the general gadget in Figure 9a
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, u1, op, \texttt{msr} \rangle$, $\langle \texttt{DigitWriter}, i, u, op, \texttt{msr} \rangle$)
    from the general gadget in Figure 9b
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, u0, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{DigitWriter}, i, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the general gadget in Figure 9a
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, u1, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{DigitWriter}, i, u, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the general gadget in Figure 9b
- For each $i = 1, 2, 3$ and each $op \in \{\texttt{increment}, \texttt{copy}\}$:
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, 0, op \rangle$, $\langle \texttt{DigitTop}, i, op \rangle$)
    from the general gadget in Figure 9a
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, 1, op \rangle$, $\langle \texttt{DigitTop}, i, op \rangle$)
    from the general gadget in Figure 9b
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, 0, op, \texttt{msr} \rangle$, $\langle \texttt{DigitTop}, i, op, \texttt{msr} \rangle$)
    from the general gadget in Figure 9a
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, 1, op, \texttt{msr} \rangle$, $\langle \texttt{DigitTop}, i, op, \texttt{msr} \rangle$)
    from the general gadget in Figure 9b
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, 0, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{DigitTop}, i, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the general gadget in Figure 9a
  - Create `Bit_Writer`($\langle \texttt{DigitWriter}, i, 1, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{DigitTop}, i, op, \texttt{msr}, \texttt{msd} \rangle$)
    from the general gadget in Figure 9b

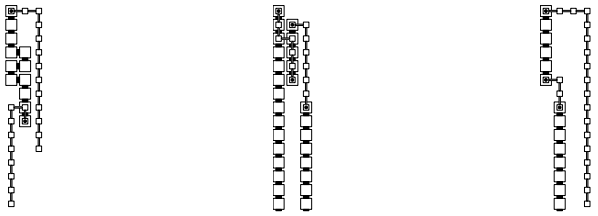
(a) Digit_Write_0 (b) Digit_Write_1

Figure 9: `Digit_Write` gadgets

2.4.4 Digit tops

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

If we examine the topper shown in Figure 10a,



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

Figure 10: Topper micro-gadgets

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

- Digit 1 (general): the following statements create the gadget shown in Figure 11a
  - Create `North_Line5`($\langle \texttt{DigitTop}, 1, op \rangle$, $\langle \texttt{DigitTop\_1\_A}, op \rangle$) from the micro-gadget shown in Figure 3a
  - Create `Topper`($\langle \texttt{DigitTop\_1\_A}, op \rangle$, $\langle \texttt{DigitTop\_1\_B}, op \rangle$) from the micro-gadget shown in Figure 10a
  - Create `South_Line4`l($\langle \texttt{DigitTop\_1\_B}, op \rangle$, $\langle \texttt{ReturnPath}, 1, op \rangle$) from the micro-gadget shown in Figure 3b

- Digit 1 (MSR): the following statements create the gadget shown in Figure 11d
  - Create `Topper`($\langle \texttt{DigitTop}, 1, op, \texttt{msr} \rangle$, $\langle \texttt{DigitTop\_1\_MSR\_A}, op \rangle$) from the micro-gadget shown in Figure 10b
  - Create `South_Line4`l($\langle \texttt{DigitTop\_1\_MSR\_A}, op, \rangle$, $\langle \texttt{ReturnPath}, op, \texttt{msr} \rangle$) from the micro-gadget shown in Figure 3b

- Digit 1 (MSD): the following statements create the gadget shown in Figure 11c

- Create North_Line4l(⟨DigitTop, 1, op, msr, msd⟩ , ⟨DigitTop_1_MSD_A, op⟩) from the micro-gadget shown in Figure 3a
- Create North_Line4(⟨DigitTop_1_MSD_A, op⟩ , ⟨DigitTop_1_MSD_B, op⟩) from the micro-gadget shown in Figure 3a
- Create Topper(⟨DigitTop_1_MSD_B, op⟩ , ⟨DigitTop_1_MSD_C, op⟩) from the micro-gadget shown in Figure 10a
- Create South_Line4l(⟨DigitTop_1_MSD_C, op⟩ , ⟨DigitTop_1_MSD_D, op⟩) from the micro-gadget shown in Figure 3b
- Create South_Line30(⟨DigitTop_1_MSD_D, op⟩ , ⟨DigitTop_1_MSD_E, op⟩) from the micro-gadget shown in Figure 3b
- Create South_Line4l(⟨DigitTop_1_MSD_E, op⟩ , ⟨DigitTop_1_MSD_F, op⟩) from the micro-gadget shown in Figure 3b
- Create South_Line14(⟨DigitTop_1_MSD_F, op⟩ , ⟨DigitTop_1_MSD_G, op⟩) from the micro-gadget shown in Figure 3b
- Create South_Line17(⟨DigitTop_1_MSD_G, op⟩ , ⟨ReturnPath, 1, op, msr, msd⟩) from the micro-gadget shown in Figure 3b

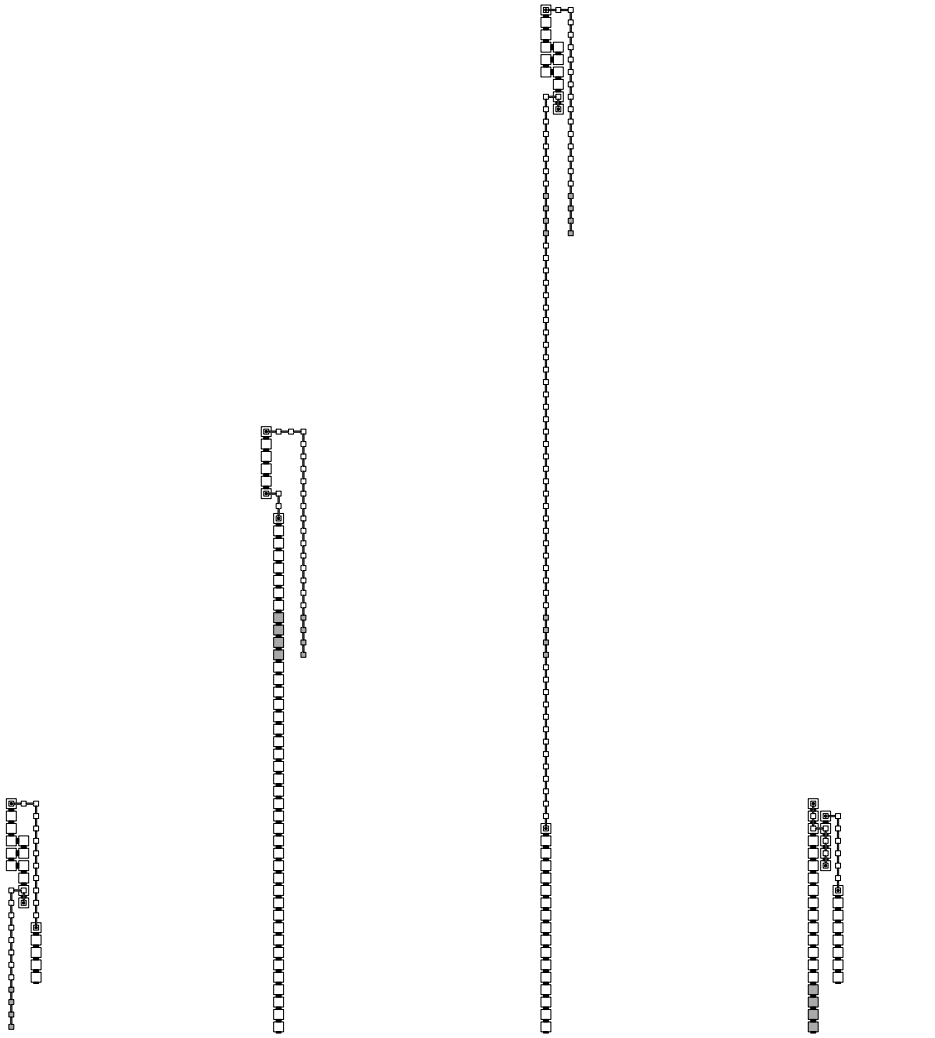
- Digit 2 (general): the following statements create the gadget shown in Figure 11a
  - Create North_Line5(⟨DigitTop, 2, op⟩ , ⟨DigitTop_2_A, op⟩) from the micro-gadget shown in Figure 3a
  - Create Topper(⟨DigitTop_2_A, op⟩ , ⟨DigitTop_2_B, op⟩) from the micro-gadget shown in Figure 10a
  - Create South_Line4l(⟨DigitTop_2_B, op⟩ , ⟨ReturnPath, 2, op⟩) from the micro-gadget shown in Figure 3b

- Digit 2 (MSD): the following statements create the gadget shown in Figure 11b
  - Create North_Line4l(⟨DigitTop, 2, op, msr, msd⟩ , ⟨DigitTop_2_MSD_A, op⟩) from the micro-gadget shown in Figure 3a
  - Create Topper(⟨DigitTop_2_MSD_A, op⟩ , ⟨DigitTop_2_MSD_B, op⟩) from the micro-gadget shown in Figure 10c
  - Create South_Line4l(⟨DigitTop_2_MSD_B, op⟩ , ⟨DigitTop_2_MSD_C, op⟩) from the micro-gadget shown in Figure 3b
  - Create South_Line30(⟨DigitTop_2_MSD_C, op⟩ , ⟨ReturnPath, 2, op, msr, msd⟩) from the micro-gadget shown in Figure 3b

- Digit 3 (general): the following statements create the gadget from Figure 11a
  - Create North_Line5(⟨DigitTop, 3, op⟩ , ⟨DigitTop_3_A, op⟩) from the micro-gadget shown in Figure 3a

  - Create `Topper`(⟨`DigitTop_3_A`, *op*⟩ , ⟨`DigitTop_3_B`, *op*⟩) from the micro-gadget shown in Figure 10a
  - Create `South_Line4`*l*(⟨`DigitTop_3_B`, *op*⟩ , ⟨`ReturnPath`, 3, *op*⟩) from the micro-gadget shown in Figure 3b

- Digit 3 (MSD): the following statements create the gadget from Figure 11a
  - Create `North_Line5`(⟨`DigitTop`, 3, *op*, `msr`, `msd`⟩ , ⟨`DigitTop_3_MSD_A`, *op*⟩) from the micro-gadget shown in Figure 3a
  - Create `Topper`(⟨`DigitTop_3_MSD_A`, *op*⟩ , ⟨`DigitTop_3_MSD_B`, *op*⟩) from the micro-gadget shown in Figure 10a
  - Create `South_Line4`*l*(⟨`DigitTop_3_MSD_B`, *op*⟩ , ⟨`ReturnPath`, 3, *op*, `msr`, `msd`⟩) from the micro-gadget shown in Figure 3b



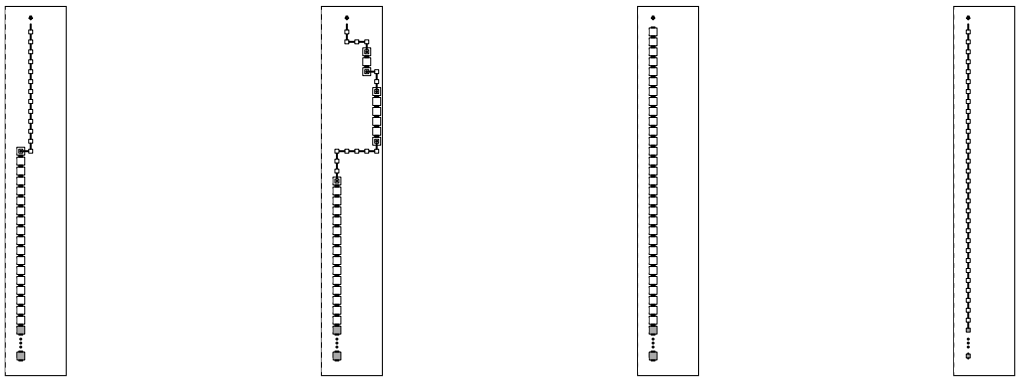
(a) General (b) Digit 2 – Case 2 (c) Digit 1 – Case 1 (d) Digit 1 – Case 2

Figure 11: `Digit_Top` gadgets

2.4.5 Return paths

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

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



(a) (b) (c) (d)

Figure 12: The return paths for digits 1 and 2

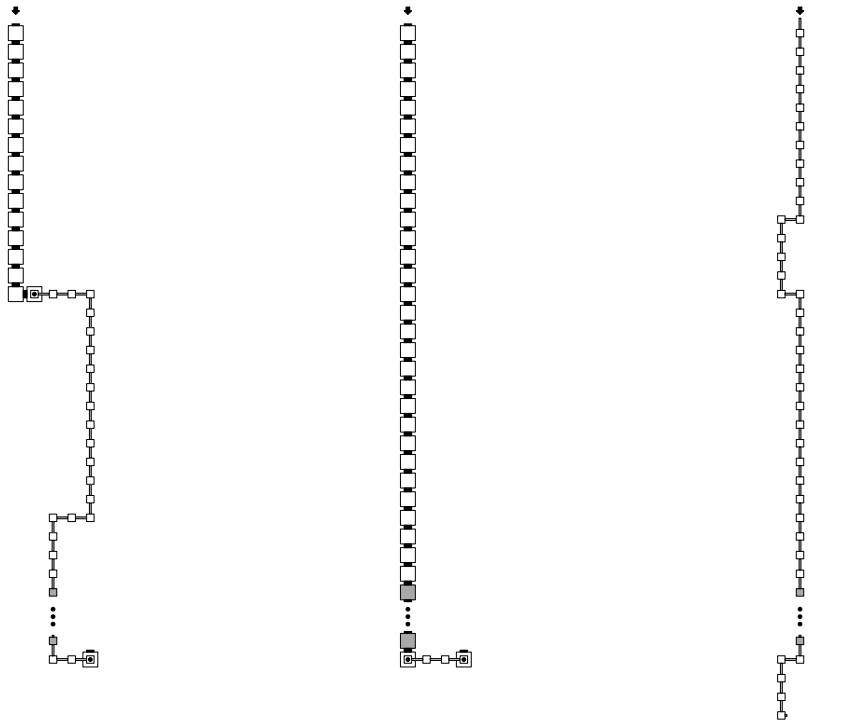
Figure 13: The `Return_From_Digit` gadget for digit 3.

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

- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 1, op \rangle$, $\langle \texttt{NextRead}, 1, op \rangle$)
  from the micro-gadget shown in Figure 12b.
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 1, op, \texttt{msr} \rangle$, $\langle \texttt{NextRead}, 1, op, \texttt{msr} \rangle$)
  from the micro-gadget shown in Figure 12c
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 1, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{NextRead}, 1, op, \texttt{msr}, \texttt{msd} \rangle$)
  from the micro-gadget shown in Figure 12d.
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 1, \texttt{seed} \rangle$, $\langle \texttt{TODO}, 1 \rangle$)
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 2, op \rangle$, $\langle \texttt{NextRead}, 2, op \rangle$)
  from the micro-gadget shown in Figure 12a.
- Create `Return_From_Digit`($\langle \texttt{ReturnPath}, 2, op, \texttt{msr}, \texttt{msd} \rangle$, $\langle \texttt{NextRead}, 2, op, \texttt{msr}, \texttt{msd} \rangle$)
  from the micro-gadget shown in Figure 12d.

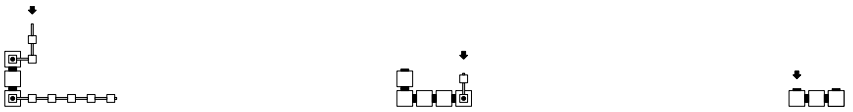
- Create `Return_From_Digit`($\langle$`ReturnPath`, 2, `seed`$\rangle$, $\langle$`TODO`, 2, $op\rangle$)
  from the micro-gadget shown in Figure 12a.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, $op\rangle$, $\langle$`NextRead`, 3, $op\rangle$)
  from the micro-gadget shown in Figure 13.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, op, `msr`, `msd`$\rangle$, $\langle$`NextRead`, 3, op, `msr`, `msd`$\rangle$)
  from the micro-gadget shown in Figure 13.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, `seed`$\rangle$, $\langle$`TODO`, 3, `seed`$\rangle$)
  from the micro-gadget shown in Figure 13.
- Create `Return_From_Digit`($\langle$`ReturnPath`, 3, `seed`, `msr`, `msd`$\rangle$, $\langle$`TODO`, 3, `seed`, `msr`, `msd`$\rangle$)
  from the micro-gadget shown in Figure 13.

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



(a) (b) (c)

Figure 14: The `Next_Read` gadgets



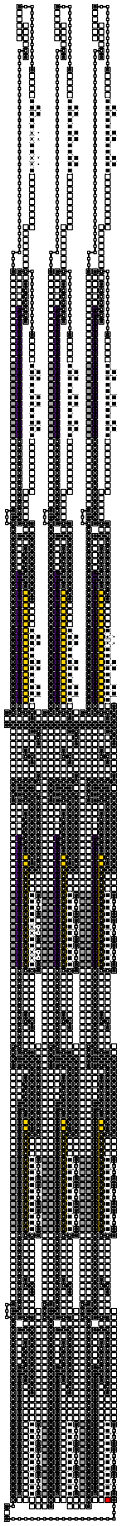
(a) (b) (c)

Figure 15: The `Next_Read` gadgets

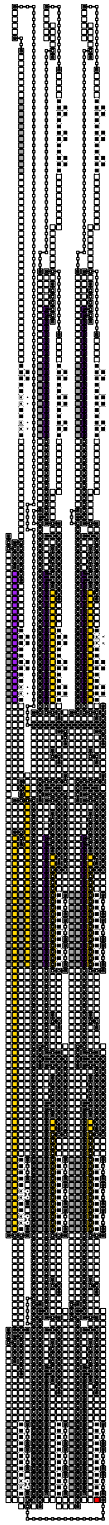
For each $op \in \{\texttt{increment}, \texttt{copy}\}$:

- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 1, op\rangle, \langle \texttt{DigitReader}, 2, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 14a.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 1, op, \texttt{msr}\rangle, \langle \texttt{DigitReader}, 2, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 14b.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 1, op, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{Cross\_Next\_Row}, op\rangle)$
  from the micro-gadget shown in Figure 14c.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 2, op\rangle, \langle \texttt{DigitReader}, 3, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 14a.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 2, op, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{Cross\_Next\_Row}, op\rangle)$
  from the micro-gadget shown in Figure 14c.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 3, op\rangle, \langle \texttt{DigitReader}, 1, \lambda, op\rangle)$
  from the micro-gadget shown in Figure 13.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 3, op, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{Cross\_Next\_Row}, op\rangle)$
  from the micro-gadget shown in Figure 15a.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 3, \texttt{seed}\rangle, \langle \texttt{TODO}, 3, \texttt{seed}\rangle)$
  from the micro-gadget shown in Figure 15b.
- Create $\texttt{Next\_Read}(\langle \texttt{NextRead}, 3, \texttt{seed}, \texttt{msr}, \texttt{msd}\rangle, \langle \texttt{Cross\_Next\_Row}, \texttt{increment}\rangle)$
  from the micro-gadget shown in Figure 15a.

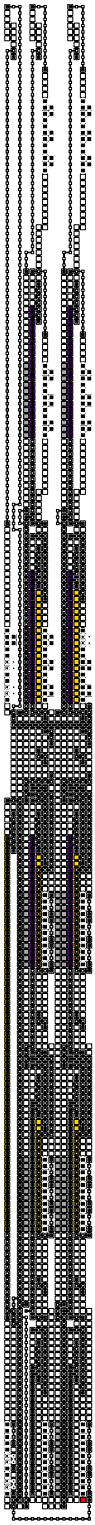
2.5 Overviews



(a) Full overview case 3



(b) Full overview case 2



(c) Full overview case 1