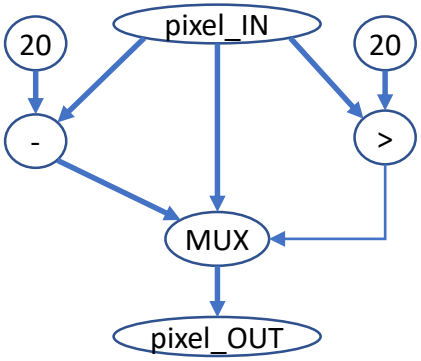
Compile and Map

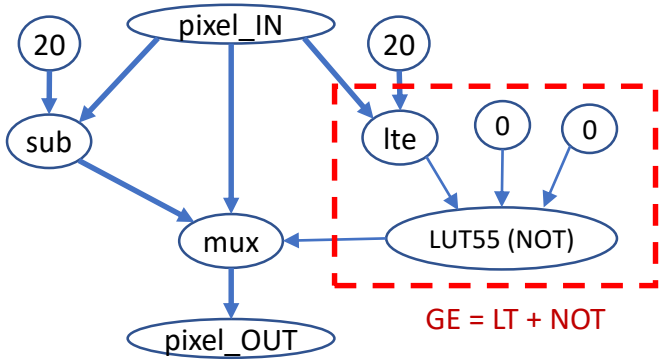
1. Start w/simple Halide program e.g. “darken” (someone else will have to help me code this as Halide)

```
foreach pixel in IMAGE:
 if pixel > 20: pixel = (pixel - 20)
 else:          pixel = 0
```

2. **Halide compiler** turns the kernel (loop body) into a directed acyclic graph (DAG):

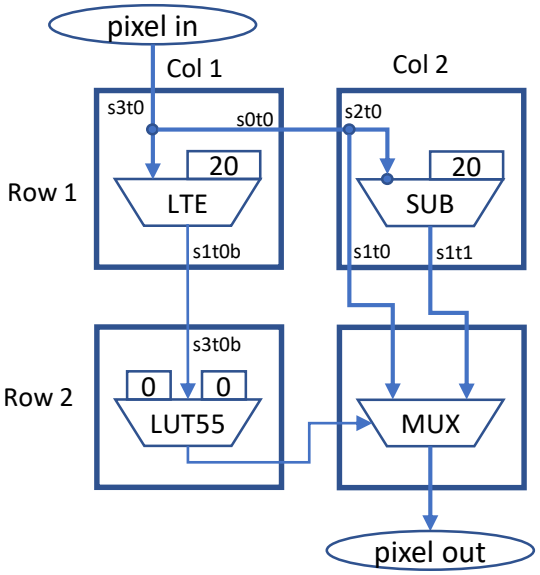


3. **Mapper** transforms DAG to use only CGRA instructions. In this example, CGRA has no greater-than so it converts that to unsigned less-than-or-equal plus NOT, where the NOT is implemented as a LUT



Place-and-Route

4. **PNR (place-and-route)** puts each operation into a separate CGRA tile and routes a path from input to output



(4x4 grid with two rows and two columns)

5. Textually PNR output looks like this (CGRA "bsb" assembly language):

```
#Program the tiles
Tx0101_lte(wire, const20)
Tx0102_sub(wire, const20)
Tx0201_lut55(const0, wire, const0)
Tx0202_mux(wire, wire, wire)

#Route the tiles
Tx0101_in_s3t0 -> Tx0101_out_s0t0
Tx0101_in_s3t0 -> Tx0101_data0
Tx0101_pe_outb -> Tx0101_s1t0b

Tx0102_in_s2t0 -> Tx0102_out_s1t0
Tx0102_in_s2t0 -> Tx0102_data0
Tx0102_pe_out -> Tx0102_out_s1t1

Tx0201_in_s3t0b -> Tx0201_bit1
Tx0201_pe_outb -> Tx0201_out_s0t0b

Tx0202_in_s3t0 -> Tx0202_data0
Tx0202_in_s3t1 -> Tx0202_data1
Tx0202_pe_out -> Tx0202_out_s1t0
```

Notes

- "Tx0101" indicates the tile at row 1, column 1, etc.
- "s3t0" means "side 3, track 0" where sides (0,1,2,3) map to (E,S,W,N) i.e. (right,bottom,left,top)

bsbuilder

6. **bsbuilder** (assembler) turns each line of assembly code turns into one address-data pair in the bitstream (), e.g.

```
# Tx0101_lte(wire, const20)
FF000101 0008D004
# data[ 5,  0] =  4 :: alu_op 'lte'
# data[ 6,  6] =  0 :: sign   'u'
# data[15, 12] = 13 :: flag   'pe'
# data[17, 16] =  2 :: data0 'wire_b' (REG_BYPASS)
# data[19, 18] =  0 :: data1  'const_a' (REG_CONST)
```

The address “FF00101” is decoded as follows:

- Bits [15,0] = tile number, eight bits each for row and column (0x0101 = r1c1)
- Bits[13,16] = element number; 0x00 indicates “ALU”
- Bits[31,24] = register number; 0xFF means “program ALU according to data word”

Here’s a sample bitstream command to route one-bit wire tracks in the tile at row 3, column 1. In this example, 0x09 is the element number for the tile’s switchbox, and the tile id is 0x0301, and the register number is 0 (i.e. address 00.09.0301), so this command sets values for wires connected to register 0 of the switchbox.

```
# Tx0301_pe_outb -> Tx0301_out_s0t2b
# Tx0301_in_s0t3b -> Tx0301_out_s1t3b
# Tx0301_in_s1t1b -> Tx0301_out_s3t1b

00090301 00800030
# data[( 5,  4)] : @ tile (3, 1) connect wire 3 (pe_out_res_p)  to out_BUS1_S0_T2
# data[(17, 16)] : @ tile (3, 1) connect wire 0 (in_BUS1_S0_T3) to out_BUS1_S1_T3
# data[(23, 22)] : @ tile (3, 1) connect wire 2 (in_BUS1_S3_T1) to out_BUS1_S2_T1
```