Arm® Ethos™-N NPU Static Performance Analyzer Tool

The **Arm® Ethos™-N NPU Static Performance Analyzer Tool** (SPA) provides a performance report using Arm NN and parts of the Ethos-N NPU driver stack, without needing to execute the inference. This can be done without accessing the hardware, Ethos-N Kernel Driver or Ethos-N Driver Library.

The Static Performance Analyzer consists of two parts:

- The `AnalyzeNetwork` tool, shipped as part of the SPA package. See the README.md for instructions on how to build it.
- **This** Jupyter notebook, showing a report based on data generated by running `AnalyzeNetwork` .

All values unless stated are **predicted future** values based on future software improvements scheduled. For current performance, please set the `CURRENT` flag to 1.

Please note that there are some limitations of the performance figures reported here. See the Limitations section in the README.md file included in the distribution package.

In [1]:
```
# For filtering operator names
import re
# For data manipulation within this notebook
import pandas
# Setup pandas global parameters to help visualisation
pandas.set_option('display.max_colwidth', 120)
pandas.set_option('display.max_rows', None)
pandas.set_option('display.max_columns', None)
# For presentation of tables within this notebooks
from IPython.display import display
# For presentation of graphs within this notebooks
%matplotlib inline
# For presentation of SVG graphs within this notebooks
from IPython.display import HTML
# For the presentation of the notebook itself, use the full screen width
display(HTML('''
<style>
    div#notebook-container    { width: 95%; }
    div#menubar-container     { width: 65%; }
    div#maintoolbar-container { width: 99%; }
</style>
'''))


# The spa module is required to extract the data from the raw json report
import spa
# Apply the global style for all charts
spa.apply_global_style()
```

Notebook Input

Input Required:
The following parameters need to be set by the user in order to run this notebook successfully

`OUTPUT_DIR` The PERFORMANCE_OUTPUT_DIR value specified in the config file
This is the path to the output from executing the driver in Performance Mode

`CLK` The Ethos-N NPU Clock Frequency to use, in Hertz

`BANDWIDTH` The DRAM Bandwidth of the system to use, in Bytes per second

In [2]:

```
# Define the output folder path generated from your run
OUTPUT_DIR = './perfreport_output/'

# Define the Clock Frequency of the Ethos-N in the target system - in Hertz
CLK = 1.25e9  # 1.25GHz

# Define the DRAM Bandwidth of the target system - in Bytes per second
BANDWIDTH = 16e9  # 16GBps
```

Summary

The below table shows a summary view of some key metrics.

In [3]:

```
# Parse the given json file
results = spa.parse_output_dir(OUTPUT_DIR)

# Get the parameters of Subgraph 0 (assuming they are the same for the rest o
params, _, _ = results[0]

summary_table = pandas.DataFrame()
for subgraph, (_, data, _) in enumerate(results):
    summary = spa.get_summary(data, BANDWIDTH, CLK)
    summary.name = 'Subgraph_{}'.format(subgraph)
    summary_table = pandas.concat([summary_table, summary], axis=1, sort=Fals

summary_table = summary_table.style.set_caption(params['Variant']).set_table_
    'selector': 'caption',
    'props': [
        ('color', 'black'),
        ('font-size', '20px'),
        ('text-align', 'center')
    ]
}])
display(summary_table)
```

```
--------------------------------------------------
Arm® Ethos™-N NPU Static Performance Analyzer Tool
--------------------------------------------------
Parsing file:
./perfreport_output/subgraph_0/report.json
Config:
Variant Ethos-N78_4TOPS_4PLE_RATIO
SramSizeBytesOverride 0
ActivationCompressionSavings 0
WeightCompressionSavings Not Specified
Current 0
--------------------------------------------------
```

Ethos-N78_4TOPS_4PLE_RATIO

	Subgraph_0
Total Cycle Count	8373024.000000
Time Taken (ms)	6.700000
Inferences per second	149.290000
Network Complexity (Millions of Operations)	54262.240000
Total DRAM Transactions (MB)	31.330000
Total SRAM Transactions (MB)	37.160000
Total Input Transactions (MB)	42.270000
Total Output Transactions (MB)	23.260000
Total Weights Transactions (MB)	2.960000
Average weight compression	0.620000

Operations

List of supported operations

This is the list of **unique** operators found in the report that are supported.

In [4]:

```
regex = re.compile(r'(:[0-9]+){2}')
ops = []
for (_, data, _) in results:
    ops.extend(regex.sub('', y) for x in data['Operation', 'Names'] for y in
uniqops = set(ops)
print(*uniqops, sep='\n')
```

```
MaxPool2D
Conv2D
Conv2D:RELU6
```

List of unsupported operations

This is the list of operators that are **not** supported!

```
In [5]: df = pandas.Series(dtype=str)
        for subgraph, (a, b, issues) in enumerate(results):
            print("a: {}".format(a))
            print("b: {}".format(b))
            s = pandas.Series(issues)
            s.index = [('Subgraph {}'.format(subgraph), idx, name) for idx, name in s
            df = df.append(s)
        if not df.empty:
            df.index = pandas.MultiIndex.from_tuples(df.index)
        df.name = 'Issues'
        df.to_frame()
```

```
a: {'Variant': 'Ethos-N78_4TOPS_4PLE_RATIO', 'SramSizeBytesOverride': 0, 'Act
ivationCompressionSavings': 0, 'WeightCompressionSavings': 'Not Specified',
'Current': 0}
b:                 Input                                                      \
   DramParallelBytes DramNonParallelBytes   SramBytes NumCentralStripes
0                0.0                  0.0   2359296.0              12.0
1                0.0                  0.0  18874368.0               0.0
2                0.0                  0.0   4915200.0              32.0
3          2162688.0             196608.0         0.0              24.0
4          2516582.0             629145.0         0.0              12.0
5                0.0                  0.0   1572864.0              12.0
6           655360.0             131072.0         0.0              12.0
7          1258291.0             314572.0         0.0               6.0
8                0.0                  0.0    786432.0               6.0
9                0.0                  0.0    393216.0               1.0
10               0.0                  0.0   1572864.0               6.0
11               0.0             786432.0         0.0               1.0
12          314572.0              78643.0         0.0               1.0
13               0.0                  0.0    786432.0               1.0
14               0.0             393216.0         0.0               1.0
15               0.0                  0.0    786432.0               1.0
16               0.0             786432.0         0.0               1.0

                                             Output                         \
   NumBoundaryStripes NumReloads DramParallelBytes DramNonParallelBytes
0                 0.0        0.0         1887436.0             471859.0
1                 0.0        0.0         4669440.0              49152.0
2                16.0        0.0         1887436.0             471859.0
3                 0.0        0.0         3014656.0             131072.0
4                 0.0        0.0               0.0                  0.0
5                 0.0        0.0          629145.0             157286.0
6                 0.0        0.0         1441792.0             131072.0
7                 0.0        0.0               0.0                  0.0
8                 0.0        0.0               0.0                  0.0
9                 0.0        0.0               0.0                  0.0
10                0.0        0.0          629145.0             157286.0
11                0.0        0.0          380928.0              12288.0
12                0.0        0.0               0.0                  0.0
13                0.0        0.0          314572.0              78643.0
14                0.0        0.0          749568.0              36864.0
15                0.0        0.0          629145.0             157286.0
16                0.0        0.0               0.0              57600.0

                                                                   \
    SramBytes NumCentralStripes NumBoundaryStripes NumReloads
0         0.0              12.0                0.0        0.0
1         0.0              96.0                0.0        0.0
2         0.0              32.0                0.0        0.0
3         0.0              24.0                0.0        0.0
4   1572864.0              12.0                0.0        0.0
5         0.0              12.0                0.0        0.0
6         0.0              12.0                0.0        0.0
7    786432.0               6.0                0.0        0.0
8    393216.0               6.0                0.0        0.0
9   1572864.0               8.0                0.0        0.0
10        0.0               6.0                0.0        0.0
```

```
11        0.0              32.0                0.0        0.0
12   786432.0              22.0                0.0        0.0
13        0.0              11.0                0.0        0.0
14        0.0              22.0                0.0        0.0
15        0.0              64.0                0.0        0.0
16        0.0               1.0                0.0        0.0

             Weights                                                   \
   DramParallelBytes DramNonParallelBytes SramBytes NumCentralStripes
0                0.0                  0.0       0.0               0.0
1                0.0                512.0       0.0               1.0
2             4096.0                  0.0       0.0               1.0
3                0.0               8704.0       0.0               1.0
4                0.0               1792.0       0.0               1.0
5             6656.0                  0.0       0.0               1.0
6                0.0              16384.0       0.0               1.0
7                0.0               4608.0       0.0               1.0
8            16384.0                  0.0       0.0               1.0
9           100352.0                  0.0       0.0               8.0
10           43520.0                  0.0       0.0               1.0
11          174592.0               5632.0       0.0              32.0
12          591360.0              28160.0       0.0              22.0
13          118272.0                  0.0       0.0              11.0
14          591360.0              28160.0       0.0              22.0
15         1196032.0                  0.0       0.0              64.0
16               0.0              22528.0       0.0               1.0

                                                              Mce
\
   NumBoundaryStripes NumReloads CompressionSavings    Operations CycleCount
0                 0.0        0.0           0.000000  0.000000e+00        0.0
1                 0.0        0.0           0.000000  3.774874e+07   196608.0
2                 0.0        0.0           0.604938  4.076863e+09   589824.0
3                 0.0        0.0           0.685185  2.717909e+09   294912.0
4                 0.0        0.0           0.125000  2.013266e+08    49152.0
5                 0.0        0.0           0.638889  1.811939e+09   196608.0
6                 0.0        0.0           0.777778  1.811939e+09   196608.0
7                 0.0        0.0           0.437500  2.013266e+08    49152.0
8                 0.0        0.0           0.777778  1.811939e+09   196608.0
9                 0.0        0.0           0.829861  3.623879e+09   393216.0
10                0.0        0.0           0.667969  8.053064e+08   196608.0
11                0.0        0.0           0.847222  7.247757e+09   786432.0
12                0.0        0.0           0.868707  7.247757e+09   786432.0
13                0.0        0.0           0.774414  8.053064e+08   196608.0
14                0.0        0.0           0.868707  7.247757e+09   786432.0
15                0.0        0.0           0.873264  1.449551e+10  1572864.0
16                0.0        0.0           0.706667  1.179648e+08    30720.0

          Ple                                     Overhead
\
   CycleCount NumOfPatches Operation StartAndEndCycleCount StripesCycleCount
0         0.0          0.0       0.0                   0.0            1200.0
1   1179648.0      98304.0       5.0                   0.0            5030.0
2    442368.0      36864.0       5.0                   0.0            9810.0
3     73728.0      12288.0      10.0                   0.0            6770.0
4     36864.0       6144.0      10.0                   0.0            3410.0
5    147456.0      12288.0       5.0                   0.0            3410.0
6     36864.0       6144.0      10.0                   0.0            3410.0
7     18432.0       3072.0      10.0                   0.0            1730.0
8     73728.0       6144.0       5.0                   0.0            1730.0
9     36864.0       6144.0      10.0                   0.0            2290.0
10    18432.0       3072.0      10.0                   0.0            1730.0
11    73728.0       6144.0       5.0                   0.0            9010.0
12    18432.0       3072.0      10.0                   0.0            6210.0
13     9216.0       1536.0      10.0                   0.0            3130.0
14    18432.0       3072.0      10.0                   0.0            6210.0
15    18432.0       3072.0      10.0                   0.0           17970.0
16     1440.0        240.0      10.0                   0.0             330.0
```

```
                                      Parent         Operation  \
   StripesNonParallelCycleCount    Ids                  Ids
0                           0.0   [[]]                  [0]
1                           0.0  [[0]]         [1, 2, 3, 4]
2                           0.0    [1]         [5, 6, 7, 8]
3                           0.0    [2]          [9, 10, 11]
4                           0.0    [3]         [12, 13, 14]
5                           0.0    [4]     [15, 16, 17, 18]
6                           0.0    [5]         [19, 20, 21]
7                           0.0    [6]         [22, 23, 24]
8                           0.0    [7]     [25, 26, 27, 28]
9                           0.0    [8]         [29, 30, 31]
10                          0.0    [9]         [32, 33, 34]
11                          0.0   [10]     [35, 36, 37, 38]
12                          0.0   [11]         [39, 40, 41]
13                          0.0   [12]         [42, 43, 44]
14                          0.0   [13]         [45, 46, 47]
15                          0.0   [14]         [48, 49, 50]
16                          0.0   [15]     [51, 52, 53, 54]


                                               Names
0                                [Input from input]
1          [None, None, Conv2D:0:0, MaxPool2D:0:1]
2          [None, None, Conv2D:0:2, MaxPool2D:0:3]
3                        [None, None, Conv2D:0:4]
4                        [None, None, Conv2D:0:5]
5          [None, None, Conv2D:0:6, MaxPool2D:0:7]
6                        [None, None, Conv2D:0:8]
7                        [None, None, Conv2D:0:9]
8        [None, None, Conv2D:0:10, MaxPool2D:0:11]
9                       [None, None, Conv2D:0:12]
10                      [None, None, Conv2D:0:13]
11       [None, None, Conv2D:0:14, MaxPool2D:0:15]
12                      [None, None, Conv2D:0:16]
13                      [None, None, Conv2D:0:17]
14                      [None, None, Conv2D:0:18]
15                      [None, None, Conv2D:0:19]
16   [None, None, Conv2D:0:20, Conv2D:0:20:RELU6]
```

Out[5]:

			Issues
Subgraph 0	**1**	**NaN**	Could not be estimated: Please provide a mapping file entry for this operation
	2	**NaN**	Could not be estimated: Please provide a mapping file entry for this operation
	3	**Conv2D:0:0**	Could not be estimated: Please provide a mapping file entry for this operation

Breakdown

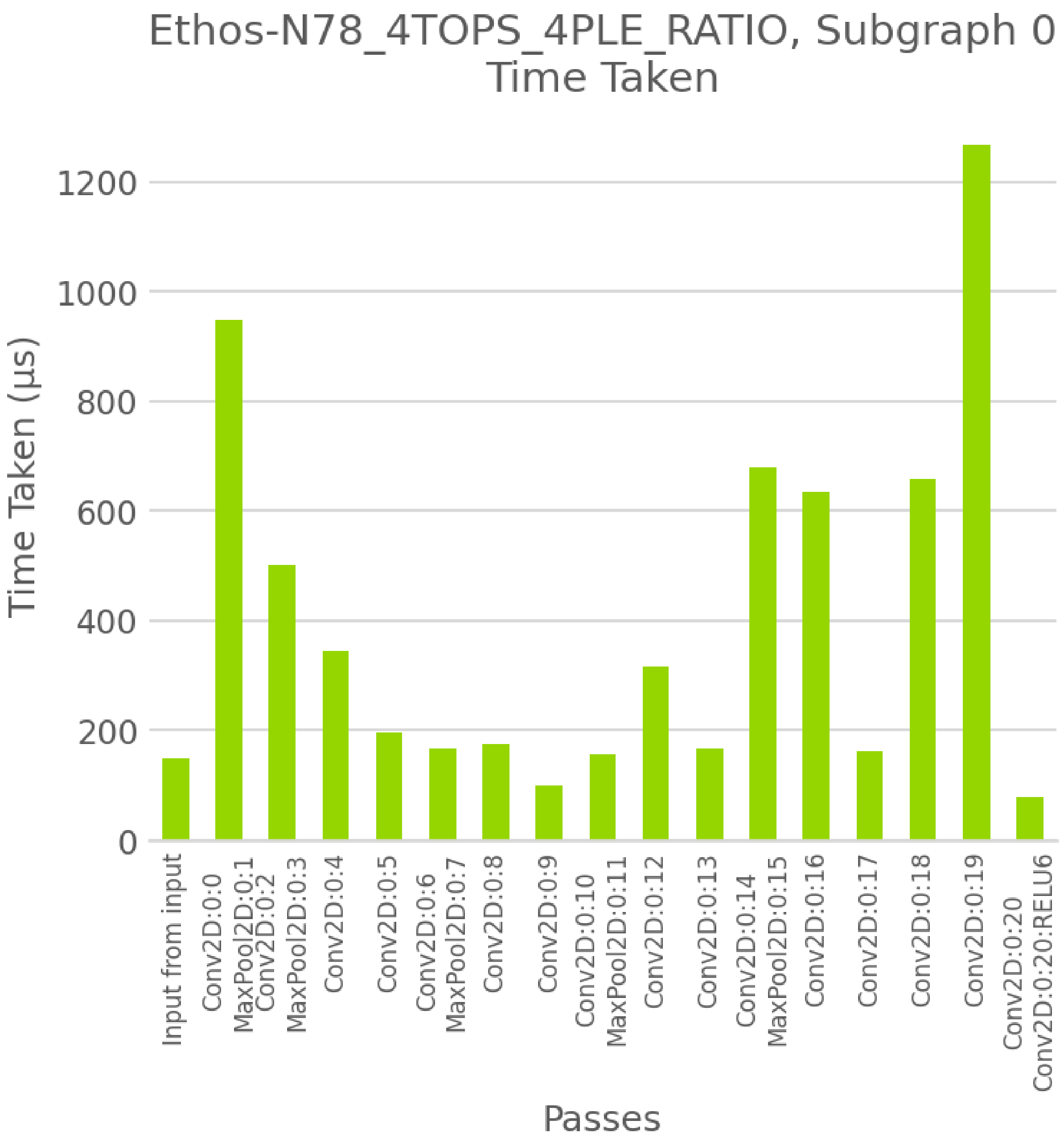
Time Taken

This is a breakdown of the time taken for each pass.

It requires a global `BANDWIDTH` value to be set to specify what the system DRAM bandwidth limit is, and it also requires a global `CLK` value to be set to specify what the clock frequency of the NPU is.

In [6]:

```
for subgraph, (_, data, _) in enumerate(results):
    cycles = spa.calc_total_cycles_range(data, BANDWIDTH / CLK)['Predicted']
    # Convert Cycles to Time, and scale to microseconds
    df = cycles.div(CLK).mul(1e6)
    ax = df.plot.bar(color=spa.ARM_COLORS['green'])
    spa.apply_formatting(ax, '{}, Subgraph {}\nTime Taken'.format(params['Var
```



DRAM Bandwidth

This is a breakdown of DRAM Bandwidth per pass, showing both Reads and Writes (stacked).

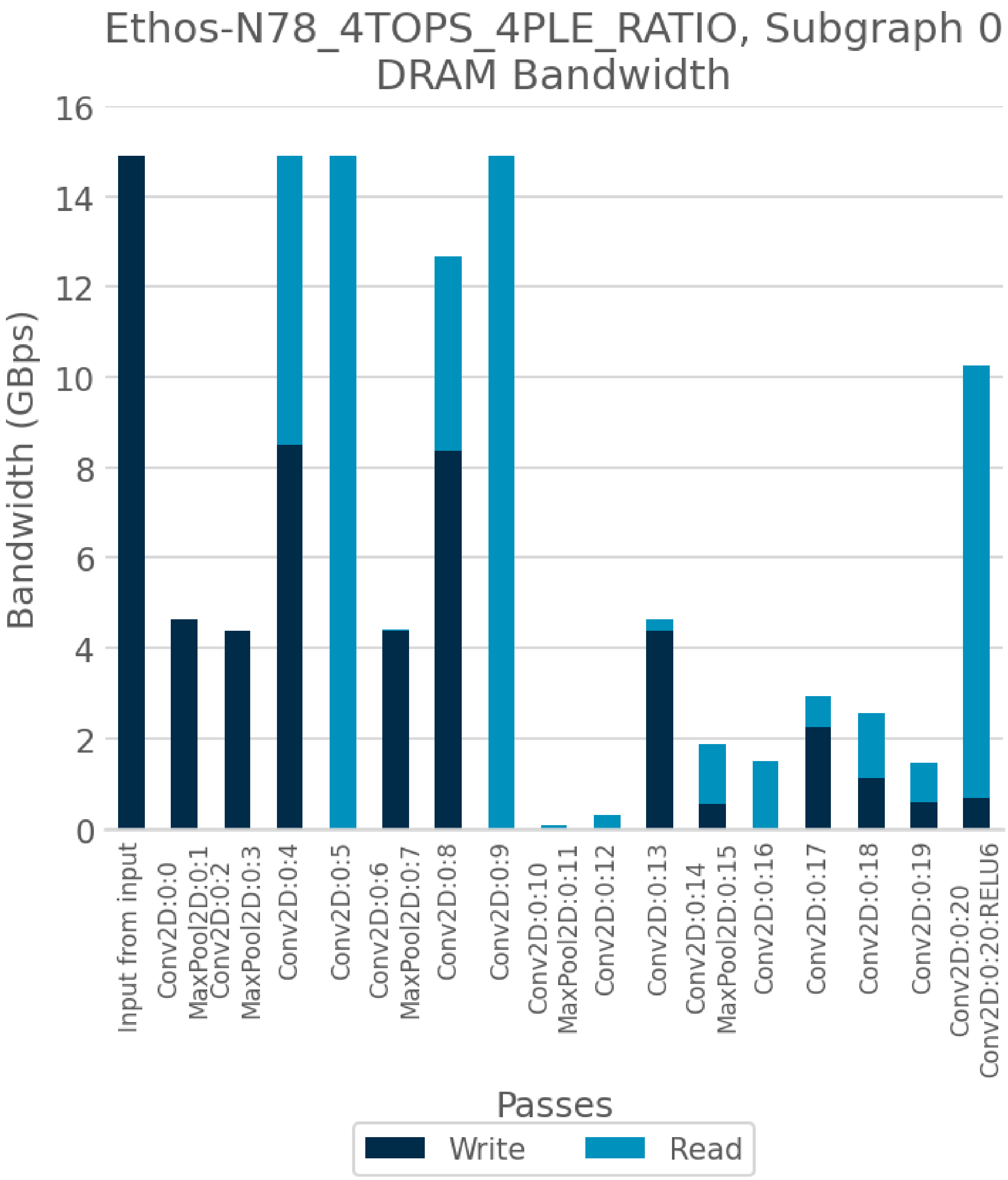
It requires a global `BANDWIDTH` value to be set to specify what the system DRAM bandwidth limit is, and it also requires a global `CLK` value to be set to specify what the clock frequency of the NPU is.

The first chart illustrates the bandwidth on a per-pass basis.

The second chart illustrates the bandwidth over time.

In [7]:

```
for subgraph, (_, data, _) in enumerate(results):
    df = spa.extract_dram_bandwidth(data, BANDWIDTH, CLK)
    df = df[df.columns[::-1]]  # Plot Writes first then Reads
    ax = df.plot.bar(stacked=True, color=spa.COOL_COLOR_PALETTE[::-1])
    ymax = BANDWIDTH / 1e9  # Scale to GBps
    spa.apply_formatting(ax, '{}, Subgraph {}\nDRAM Bandwidth'.format(params[
```

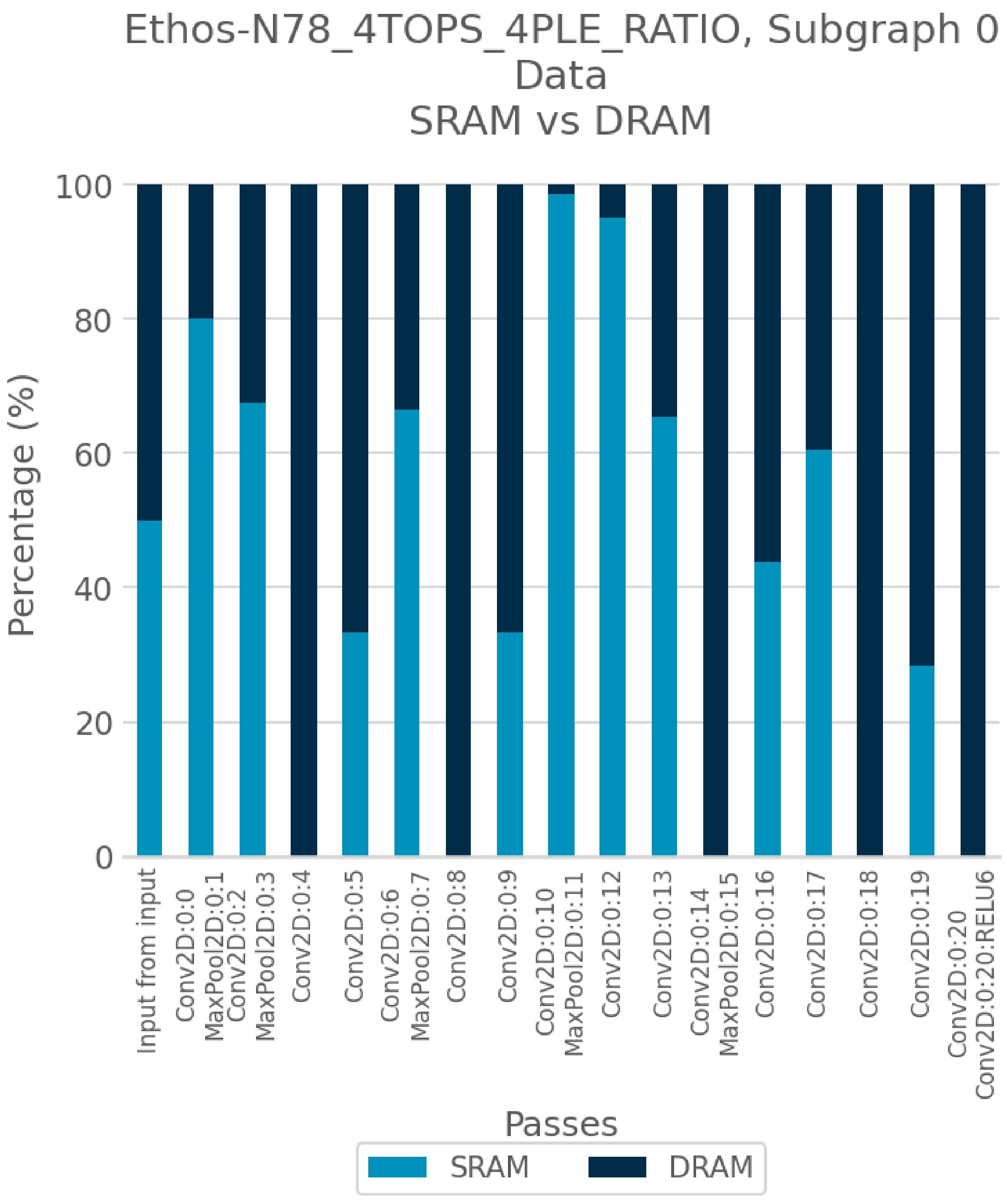


for subgraph, (_, data, _) in enumerate(results): df = spa.extract_dram_bandwidth(data, BANDWIDTH, CLK) timetaken = spa.extract_time_taken(data, BANDWIDTH, CLK) # Scale the timeline to ms timetaken *= 1e3 # Plot a custom stacked bar chart where the width of the bar is the time taken spa.plt.bar(x=timetaken.cumsum(), height=df['Write'], width=-timetaken, align='edge', label='Write', color=spa.ARM_COLORS['dark blue']) spa.plt.bar(x=timetaken.cumsum(), height=df['Read'], bottom=df['Write'], width=-timetaken, align='edge', label='Read', color=spa.ARM_COLORS['blue']) # Get the current axes for manipulating ax = spa.plt.gca() # Set the x-axis limits ax.set_xlim(0, timetaken.sum()) ymax = BANDWIDTH / 1e9 # Scale to GBps spa.apply_formatting(ax, '{}, Subgraph {}\nDRAM Bandwidth'.format(params['Variant'], subgraph), 'Time (ms)', 'Bandwidth (GBps)', ymax=ymax, figwidth=12)

Data SRAM vs DRAM

This is a breakdown of where the data is for each pass (both Read and Write).

```
In [8]: for subgraph, (_, data, _) in enumerate(results):
    df = spa.extract_sram_v_dram(data)
    ax = df.plot.bar(stacked=True)
    spa.apply_formatting(ax, '{}, Subgraph {}\nData\nSRAM vs DRAM'.format(par
```



DRAM Memory Transactions

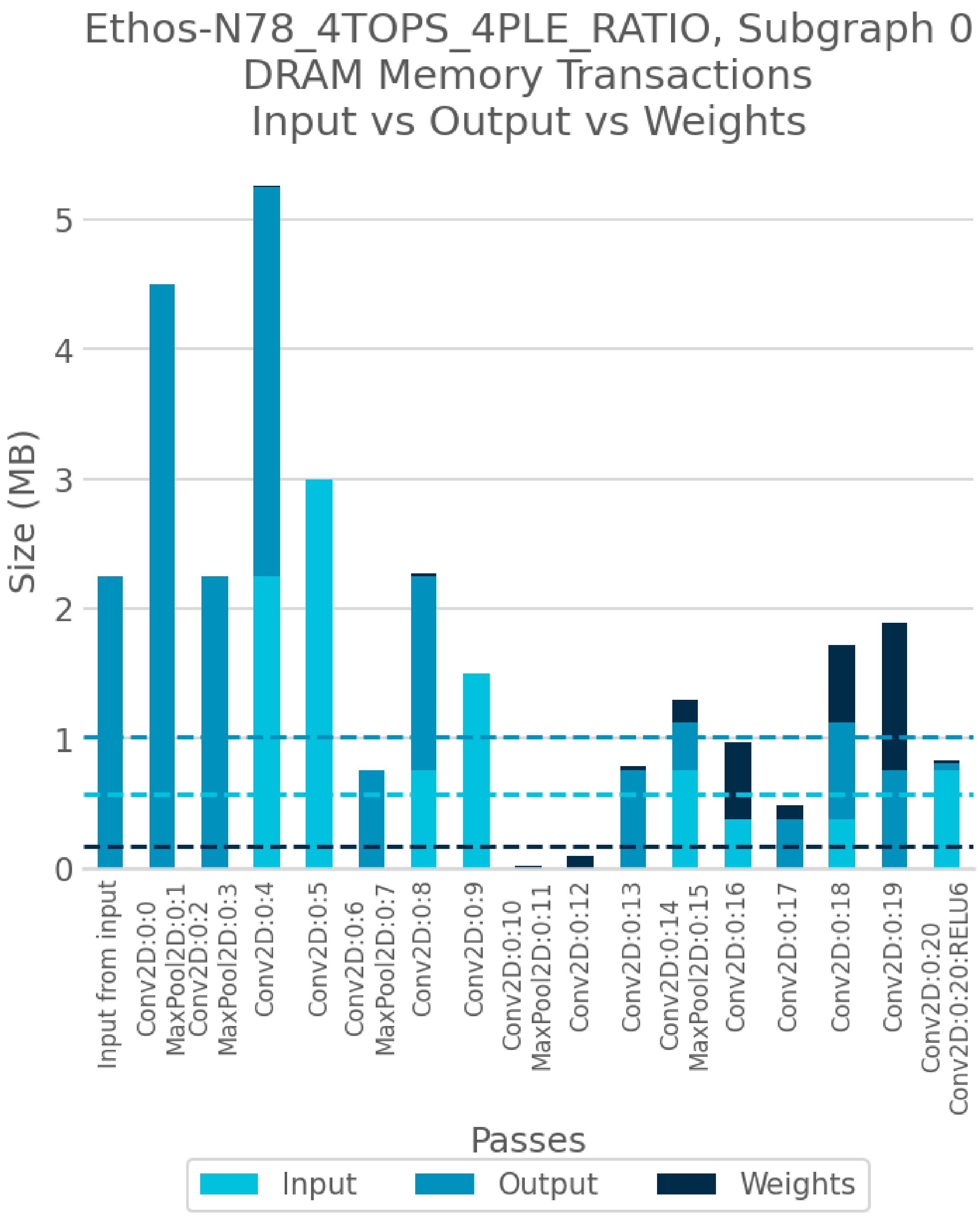
This is a breakdown of DRAM memory transactions per pass.

The first chart illustrates the split between Inputs, Outputs and Weights for the DRAM transactions that occur during that pass.

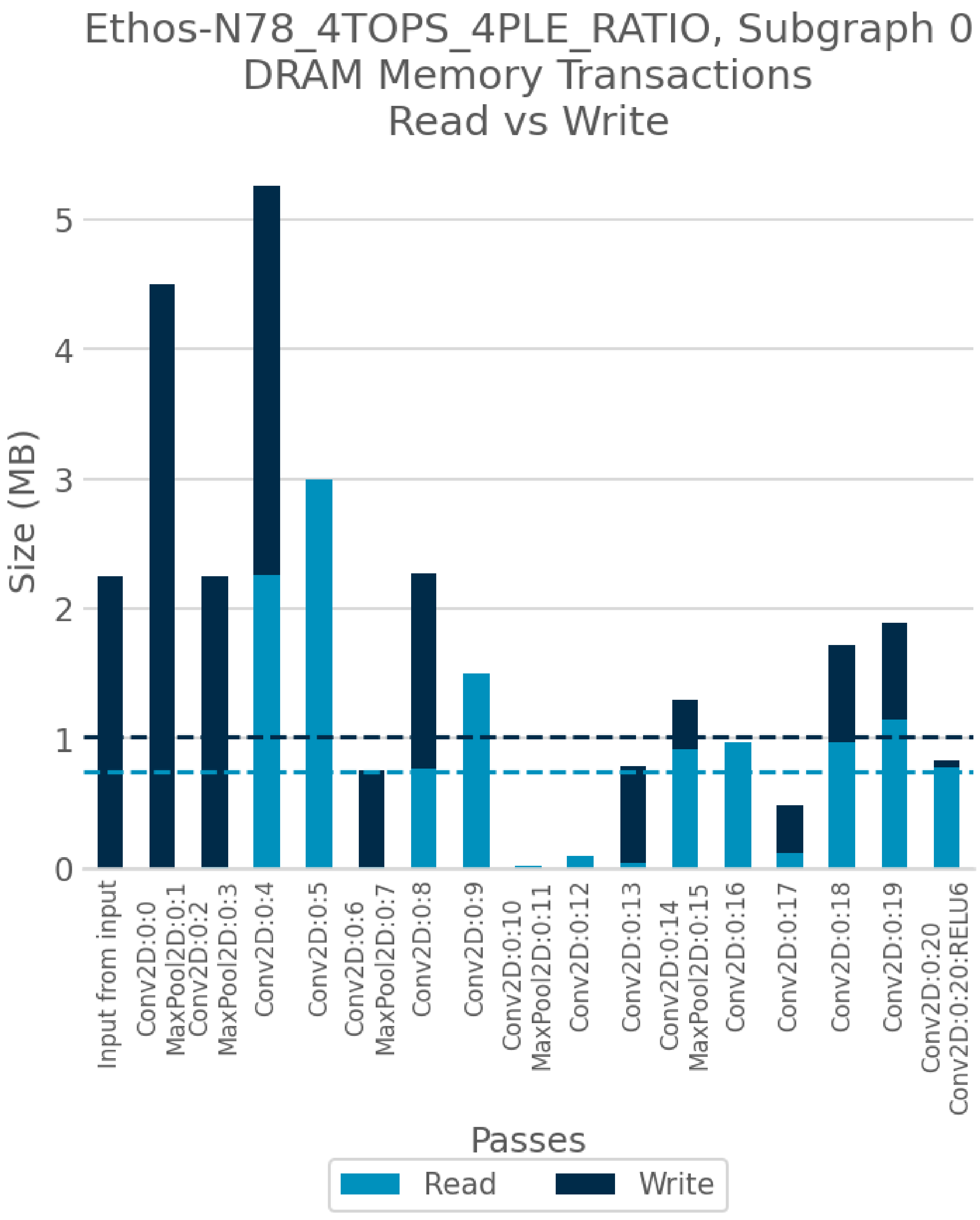
The second chart illustrates the split between Reads and Writes for the same data.

The dotted horizontal line is the AVERAGE for each.

In [9]:
```
for subgraph, (_, data, _) in enumerate(results):
    df = spa.extract_dram_transaction_type(data)
    ax = df.plot.bar(stacked=True, color=spa.COOL_COLOR_PALETTE)
    spa.apply_mean(ax, df, color=spa.COOL_COLOR_PALETTE)
    spa.apply_formatting(ax, '{}, Subgraph {}\nDRAM Memory Transactions\nInpu
```



In [10]:
```
for subgraph, (_, data, _) in enumerate(results):
    df = spa.extract_dram_transaction_rw(data)
    ax = df.plot.bar(stacked=True)
    spa.apply_mean(ax, df)
    spa.apply_formatting(ax, '{}, Subgraph {}\nDRAM Memory Transactions\nRead
```



Weight Compression Savings

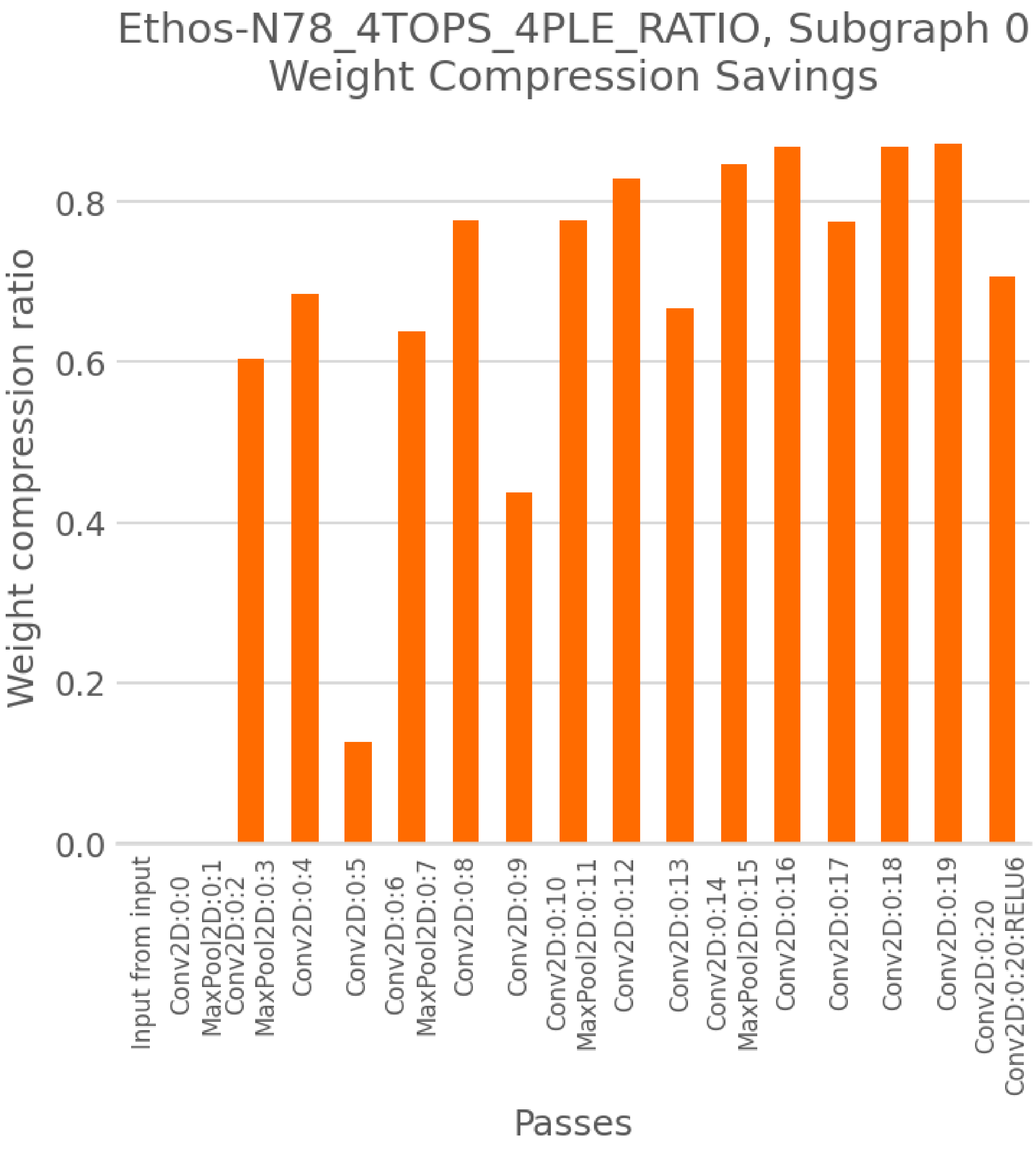
This is a breakdown of the weight compression of each pass

The weight compression savings for a pass is the percentage reduction of the space taken up by the weights, that has been saved by using compression. Higher values therefore indicate more compression has taken place.

The value is a "Space Saving" ratio, following the formula: `1 - (compressed / uncompressed)`

In [11]:

```
for subgraph, (_, data, _) in enumerate(results):
    df = spa.extract_weight_compression(data)
    ax = df.plot.bar(color=spa.ARM_COLORS['orange'])
    '{}, Subgraph {}\nWeight Compression Savings'.format(params['Variant'], s
    spa.apply_formatting(ax, '{}, Subgraph {}\nWeight Compression Savings'.fo
```

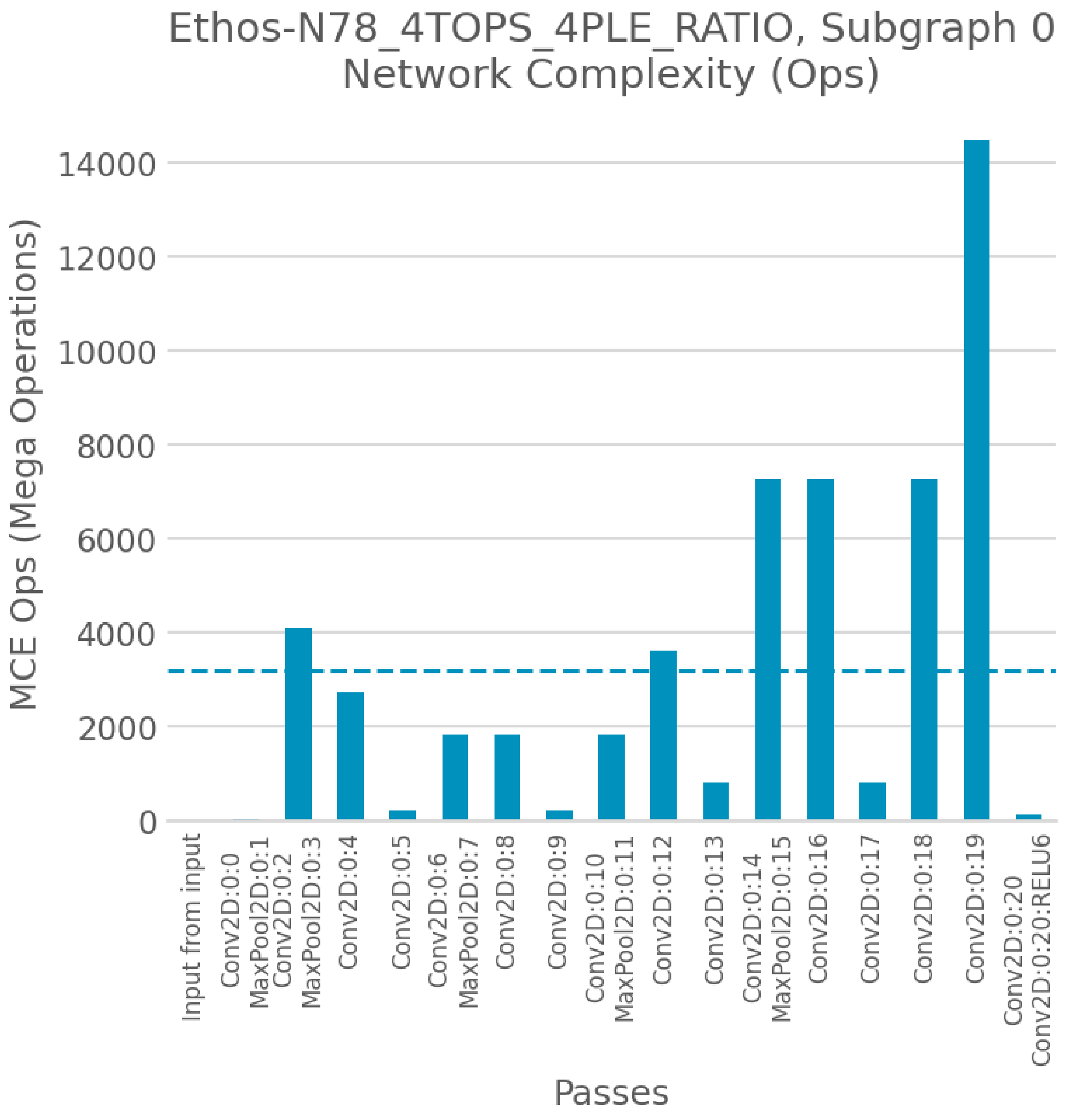


Network Complexity

This is a breakdown of complexity of each pass.

Complexity is the number of operations required for the MCE to perform that pass.

In [12]:
```
for subgraph, (_, data, _) in enumerate(results):
    df = spa.extract_mce_complexity(data, scale=1e6)
    ax = df.plot.bar()
    spa.apply_mean(ax, df)
    spa.apply_formatting(ax, '{}, Subgraph {}\nNetwork Complexity (Ops)'.form
```



Cycle Count

This is a breakdown of cycles per pass.

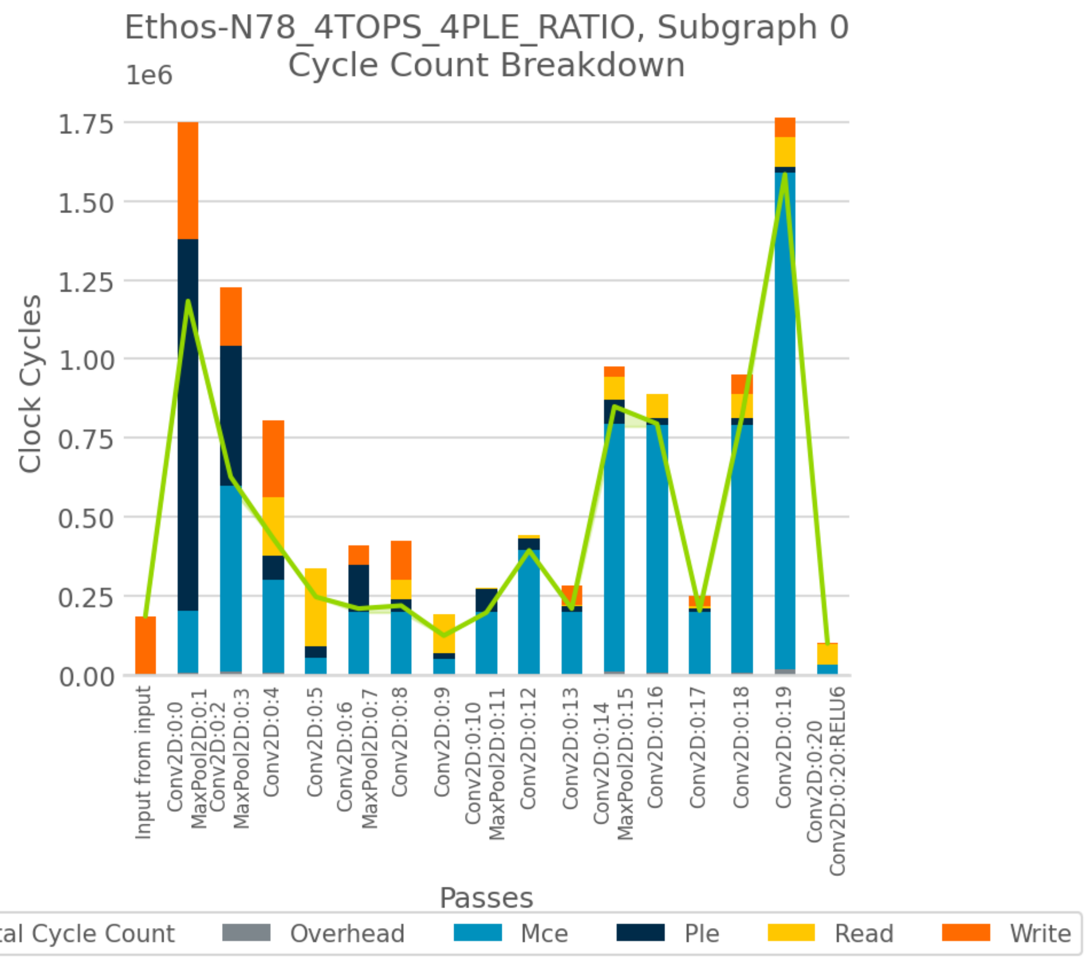
The stacked bar chart displays a cycle count for each major unit: MCE, PLE, DMA Read, DMA Write; as well as the current firmware overhead.

The line chart displays the total predicted number of cycles it took to perform that pass.

This line can be lower than the stacked chart, and when this happens, it denotes that parallelism is occurring.

The line also has a shaded area around it, which denotes theoretical upper and lower bounds for the predicted cycle count.

In [13]:
```
for subgraph, (_, data, _) in enumerate(results):
    totals, df = spa.extract_cycle_count(data, BANDWIDTH, CLK)
    lower = totals.xs('Lower', axis=1, level=1).squeeze()
    upper = totals.xs('Upper', axis=1, level=1).squeeze()
    predicted = totals.xs('Predicted', axis=1, level=1).squeeze()
    spa.plt.fill_between(totals.index, lower, upper, color=spa.ARM_COLORS['gr
    ax = predicted.plot.line(color=spa.ARM_COLORS['green'], lw=2)
    palette = [spa.ARM_COLORS['dark gray']] + spa.COOL_COLOR_PALETTE[1:] + sp
    df.plot.bar(ax=ax, stacked=True, color=palette)
    spa.apply_formatting(ax, '{}, Subgraph {}\nCycle Count Breakdown'.format(
```



Graph of operations

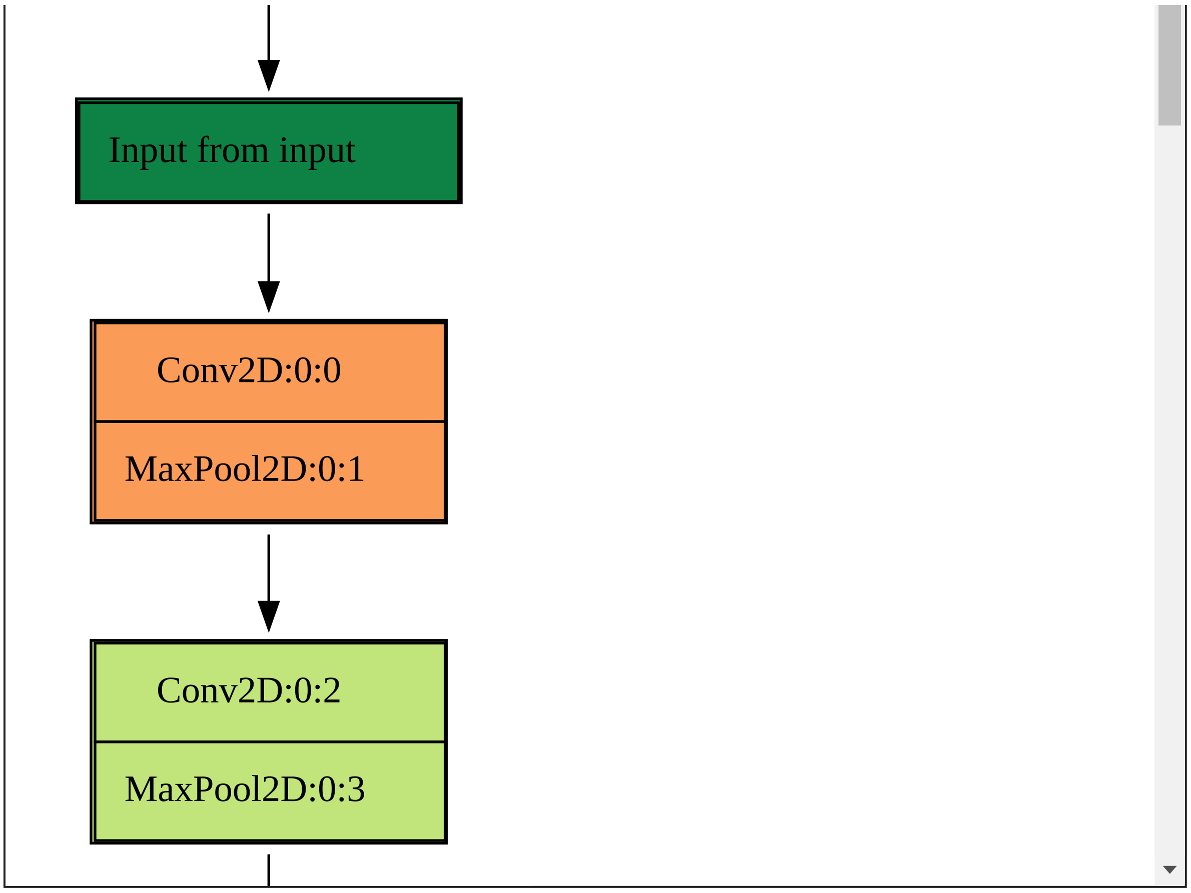
A graph representation of the operations that would be executed on the NPU.

Each node in this graph represents a command that would be passed to the hardware. Each command may be doing computations that correspond to more than one layer in the original network. Similarly a layer might be split across multiple commands.

Nodes are color coded with a RAG rated color map based on the estimated number of cycles that they would take to run. Red for nodes that take longer, green for nodes that take less time.

In [14]:
```
for subgraph, (_, data, _) in enumerate(results):
    # Heatmap the graph using the cycle count
    cycles = spa.calc_total_cycles_range(data, BANDWIDTH / CLK)['Predicted']
    svg = spa.get_graph_svg(data, cycles)
    print('Subgraph', subgraph)
    html = '<div style="max-height:500px;border-style:solid;border-width:1px;
    display(HTML(html))
```

```
Subgraph 0
```



Detailed Report

In [15]:
```
for subgraph, (_, data, _) in enumerate(results):
    # Display the parsed data in raw form
    print('Subgraph', subgraph)
    display(data)
```

```
Subgraph 0
```

	DramParallelBytes	DramNonParallelBytes	SramBytes	NumCentralStripes	NumBoundaryStripes
0	0.0	0.0	2359296.0	12.0	0.
1	0.0	0.0	18874368.0	0.0	0.
2	0.0	0.0	4915200.0	32.0	16.
3	2162688.0	196608.0	0.0	24.0	0.
4	2516582.0	629145.0	0.0	12.0	0.
5	0.0	0.0	1572864.0	12.0	0.

	DramParallelBytes	DramNonParallelBytes	SramBytes	NumCentralStripes	NumBoundaryStripe
6	655360.0	131072.0	0.0	12.0	0.
7	1258291.0	314572.0	0.0	6.0	0.
8	0.0	0.0	786432.0	6.0	0.
9	0.0	0.0	393216.0	1.0	0.
10	0.0	0.0	1572864.0	6.0	0.
11	0.0	786432.0	0.0	1.0	0.
12	314572.0	78643.0	0.0	1.0	0.
13	0.0	0.0	786432.0	1.0	0.
14	0.0	393216.0	0.0	1.0	0.
15	0.0	0.0	786432.0	1.0	0.
16	0.0	786432.0	0.0	1.0	0.

```
In [ ]:
```

```
In [ ]:
```