Automatic Pipelining for CGRA Applications

Techniques, Analysis, and Future Directions

Jack Melchert

Motivation

Problem: The applications running on our CGRA run at very low frequencies

Gaussian Blur	103 MHz
Harris Corner Detection	26 MHz
Camera Pipeline	17 Mhz

The maximum frequency of the CGRA is 780 MHz at 1.1 V

Motivation

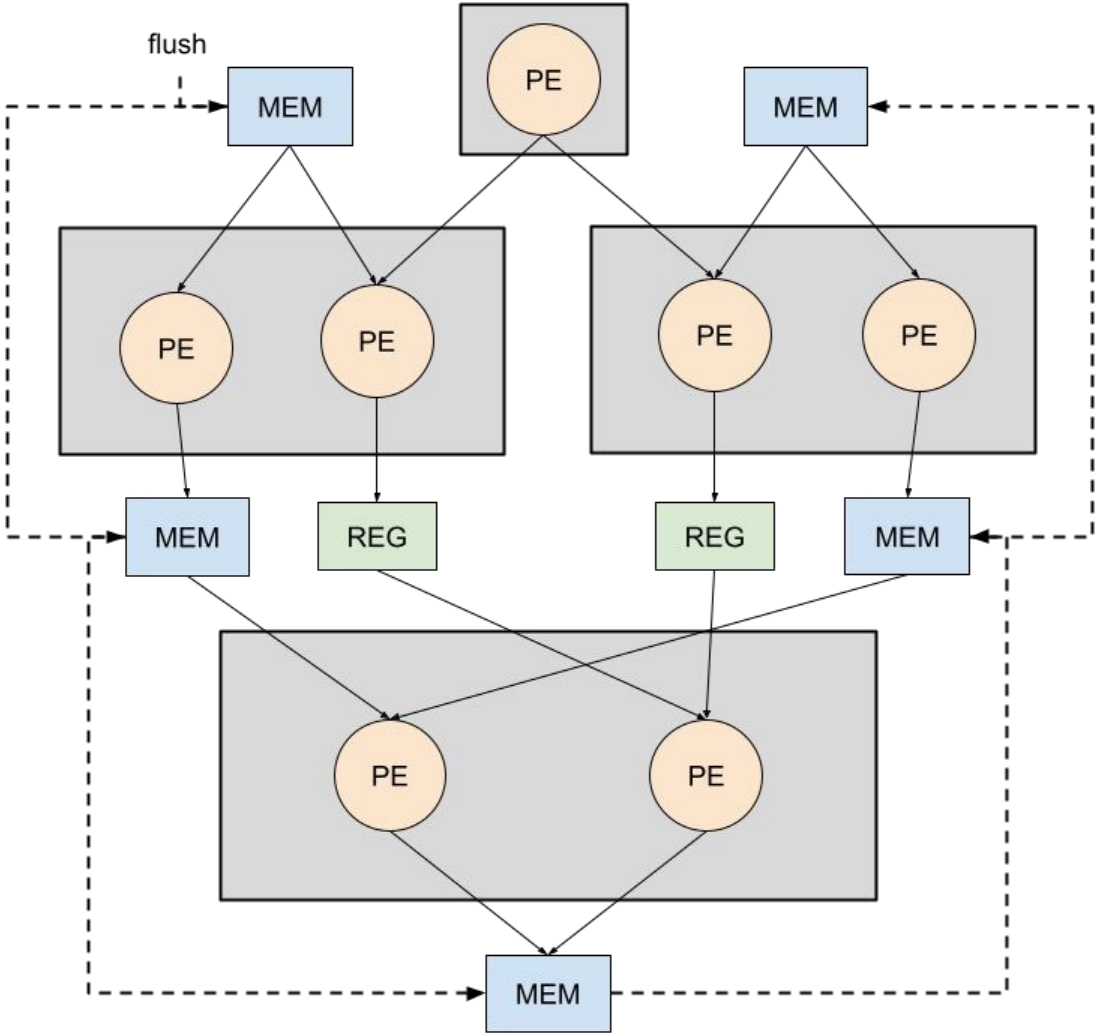
Solution: Pipeline applications

- Un-pipelined applications have critical paths that typically start or end at a memory tile or outside of the array
- Adding pipelining registers to these paths should dramatically increase the maximum frequency of the applications

Goals

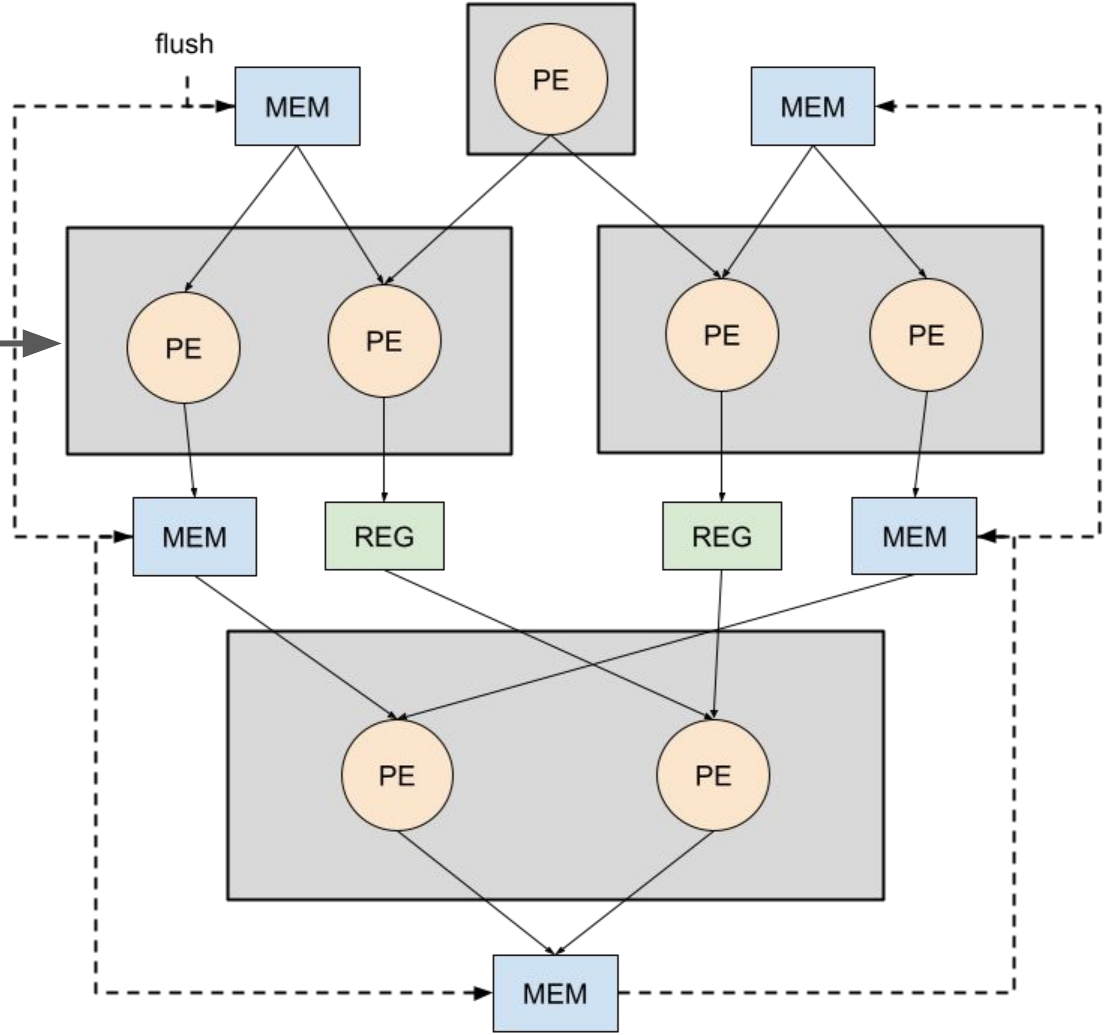
- Understand and model the delays in the array
  - Measure all contributions to the critical path
  - Integrate that data into a timing analysis tool that will accurately model the critical path in an application
- Create an automated pipelining approach
  - Our applications and hardware change frequently, so an automated approach is required

Terminology

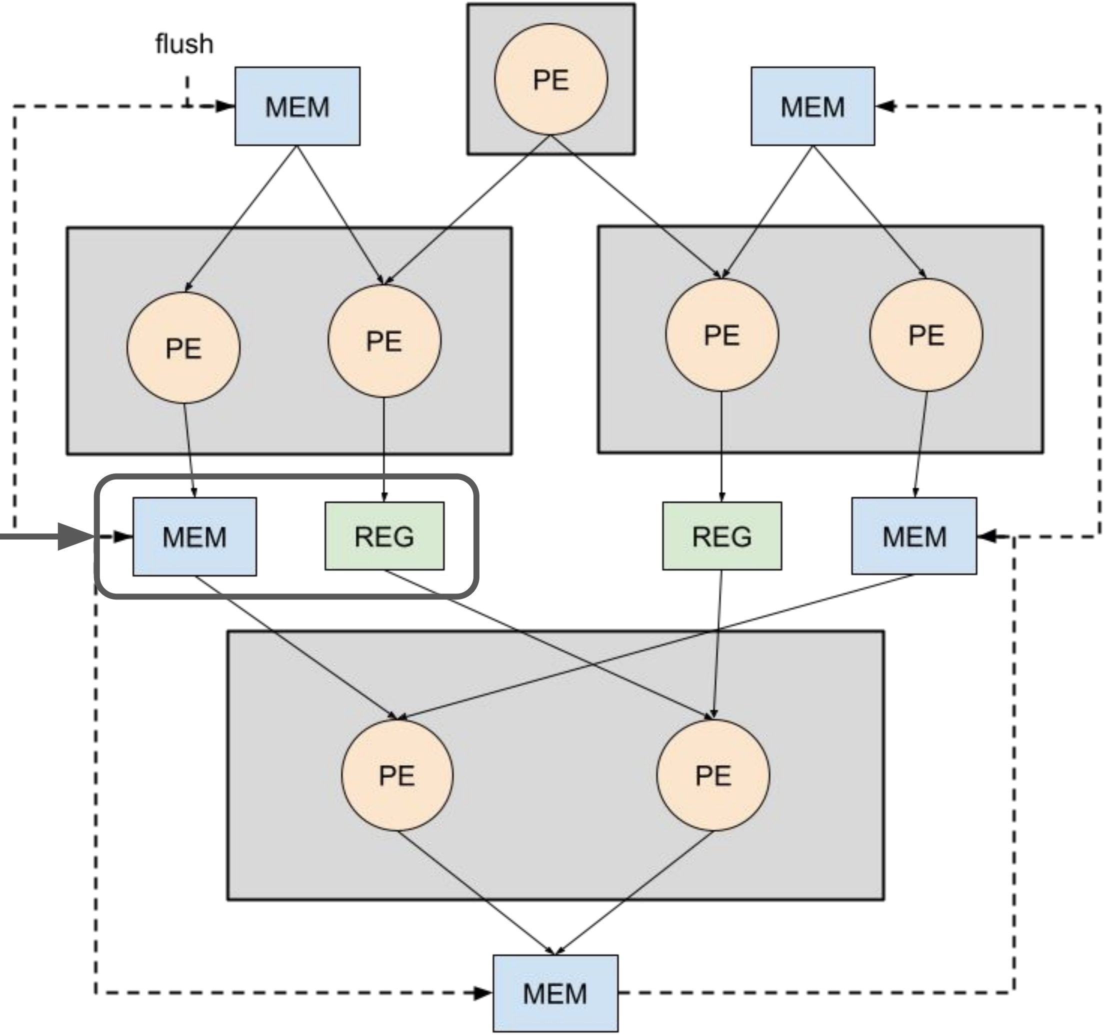


Terminology

Compute Kernel

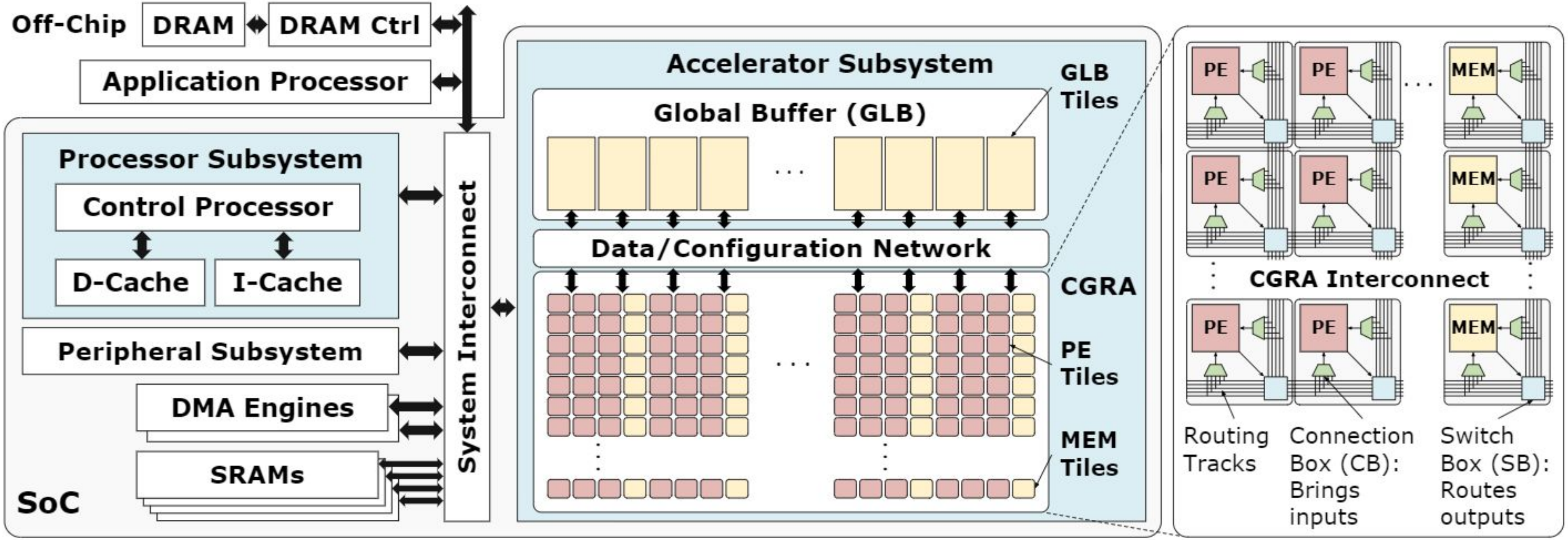


Terminology

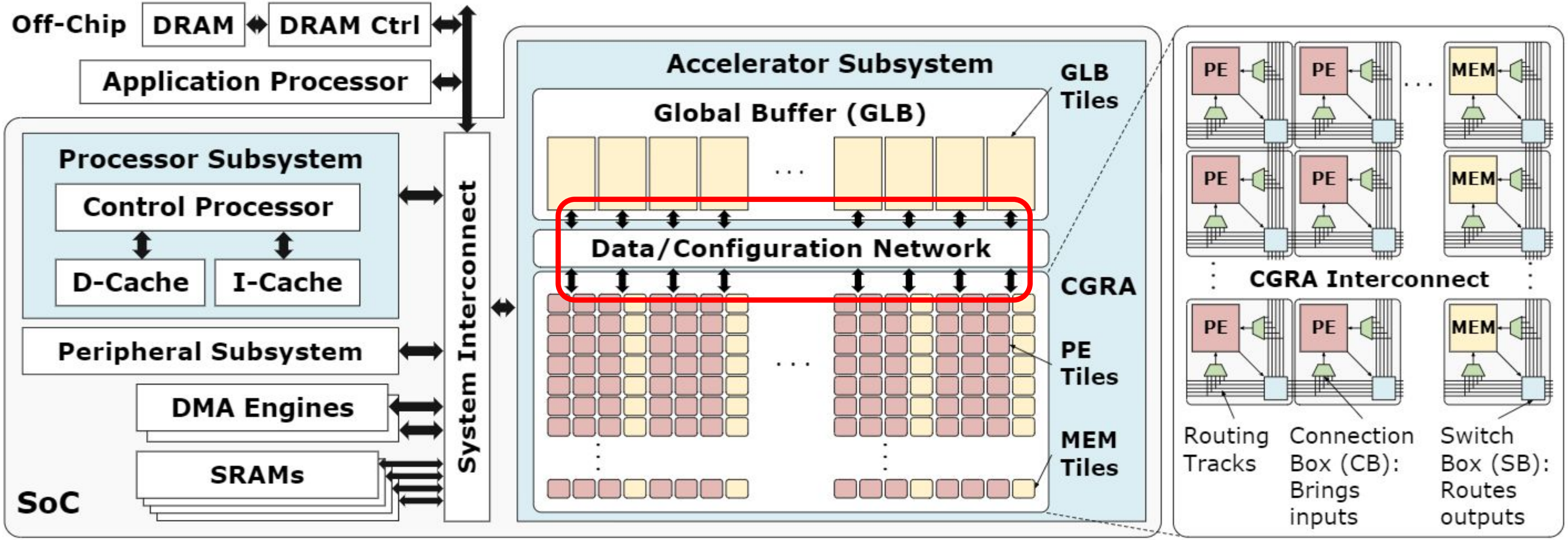


Memory Kernel

Contributions to Critical Path Delay

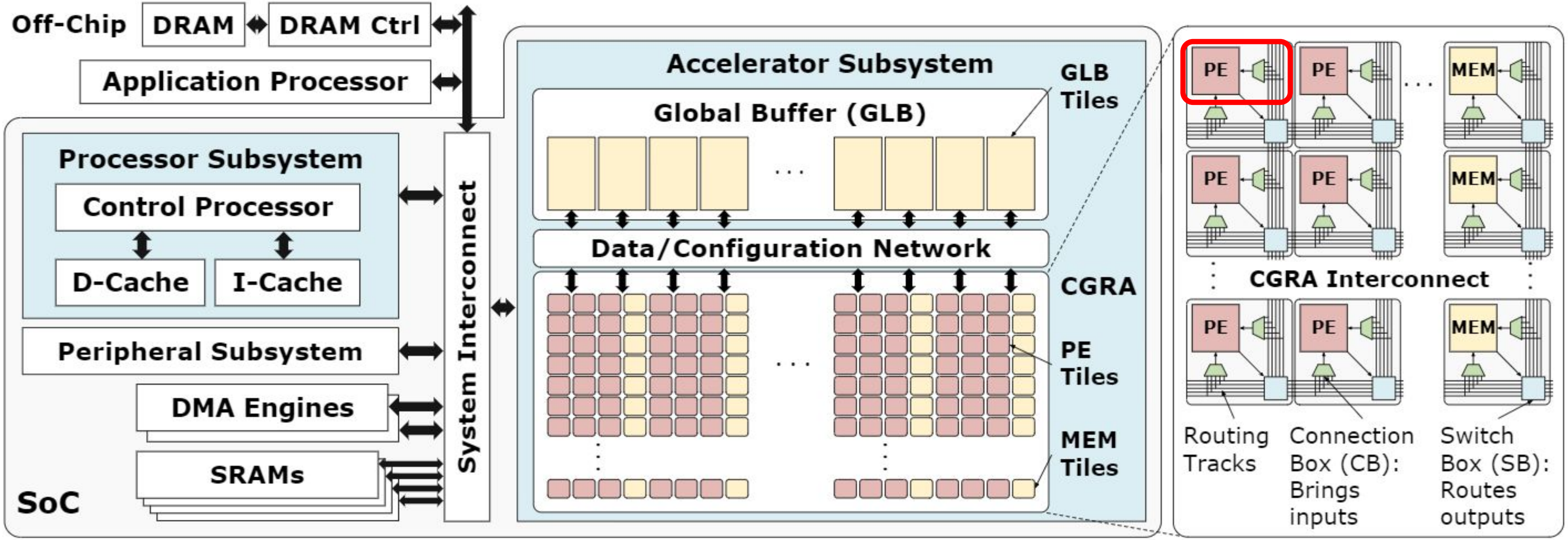


Contributions to Critical Path Delay



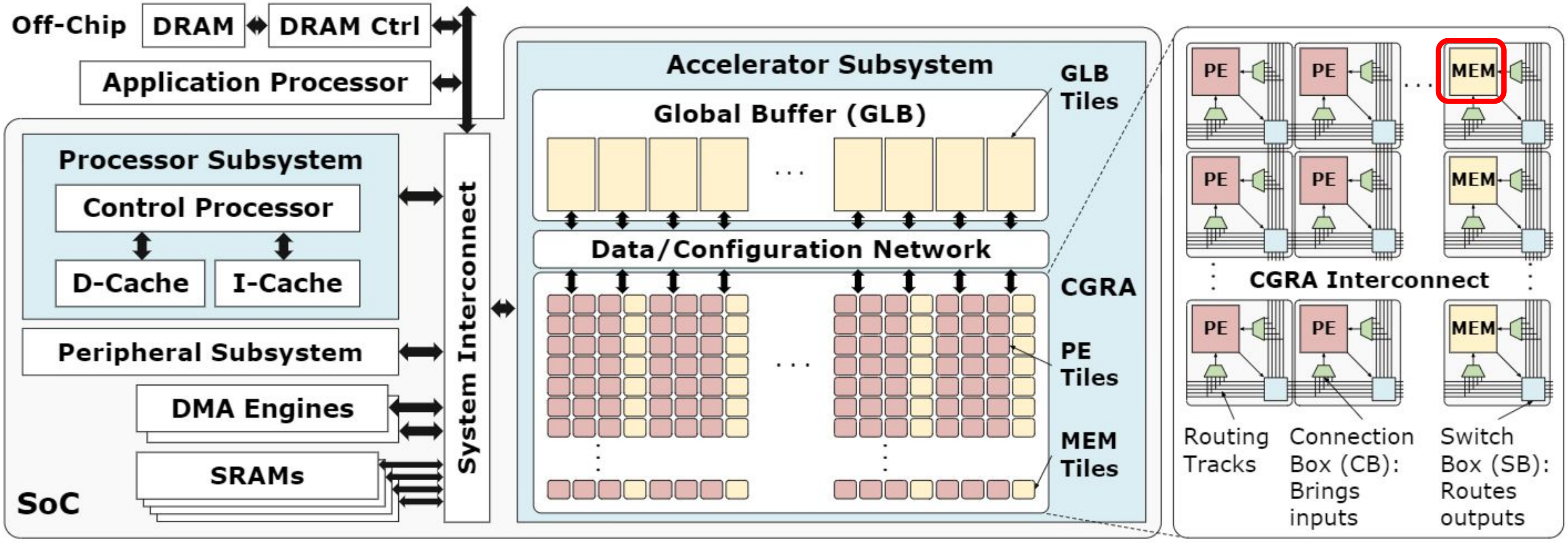
Global Buffer Delay: 1.1 ns

Contributions to Critical Path Delay



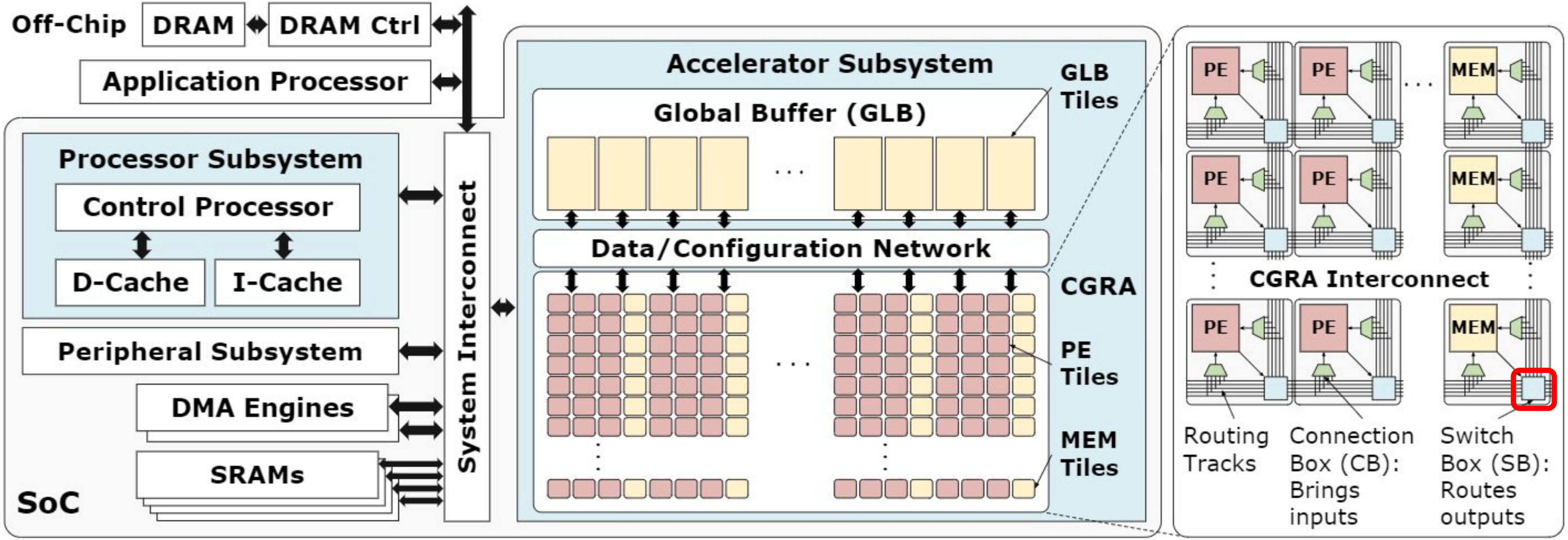
PE Delay: 0.48-0.70 ns

Contributions to Critical Path Delay



MEM Delay: 0 ns

Contributions to Critical Path Delay



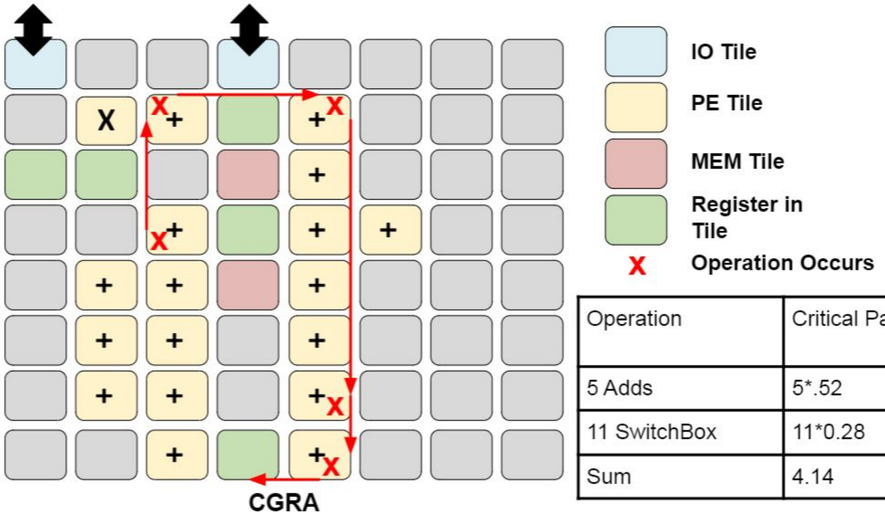
SB Delay: 0.14 ns

Critical Path Model

Contributions:

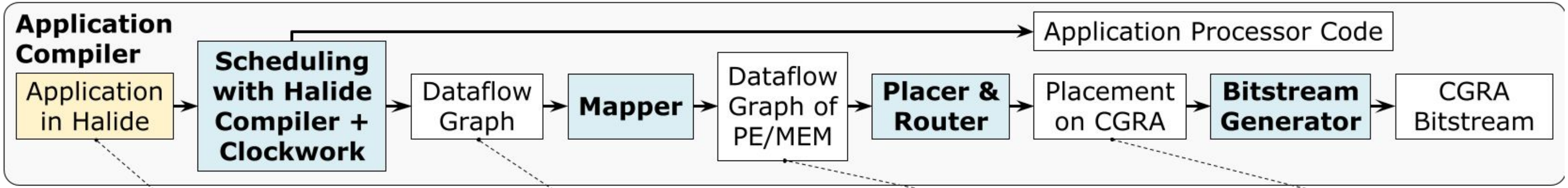
- Global Buffer: 1.1 ns
- Switch Boxes: 0.14 ns
- PEs: 0.48-0.7 ns

After about 3 hops in the interconnect, a critical path with 1 PE will be dominated by the interconnect



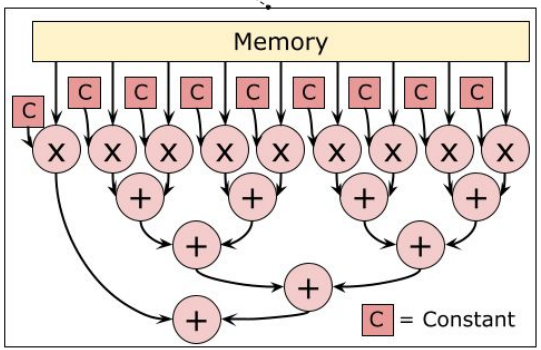
Operation	Critical Path(ns)
5 Adds	5*.52
11 SwitchBox	11*0.28
Sum	4.14

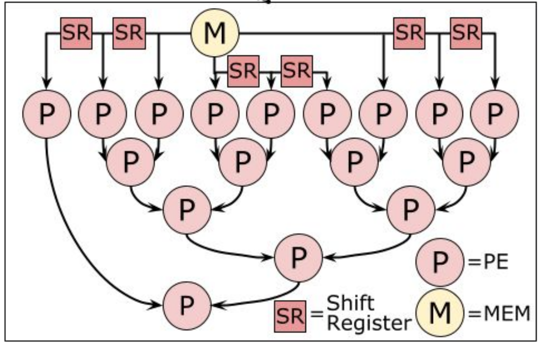
Critical Path Model

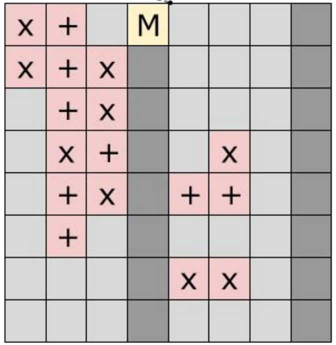


```
conv(x, y) += kernel(r.x, r.y) *
              input(x+r.x, y+r.y);
conv.in().compute_root();
conv.in()
    .tile(x,y,xo,yo,xi,yi,64,64)
    .hw_accelerate(xi, xo);
conv.update()
    .unroll(r.y, 3)
    .unroll(r.x, 3);
conv.compute_at(conv.in(), xo);
input.stream_to_accelerator();
```

Example: 3x3 Convolution

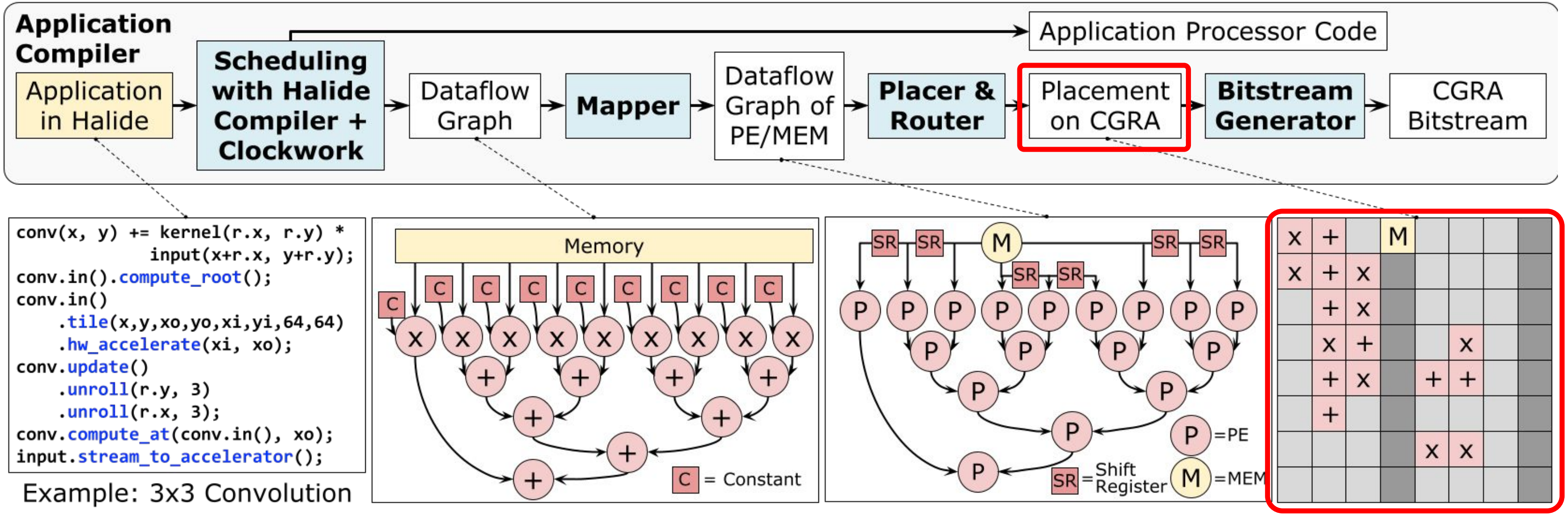






X	+		M				
X	+	X					
	+	X					
	X	+			X		
	+	X		+	+		
	+						
				X	X		

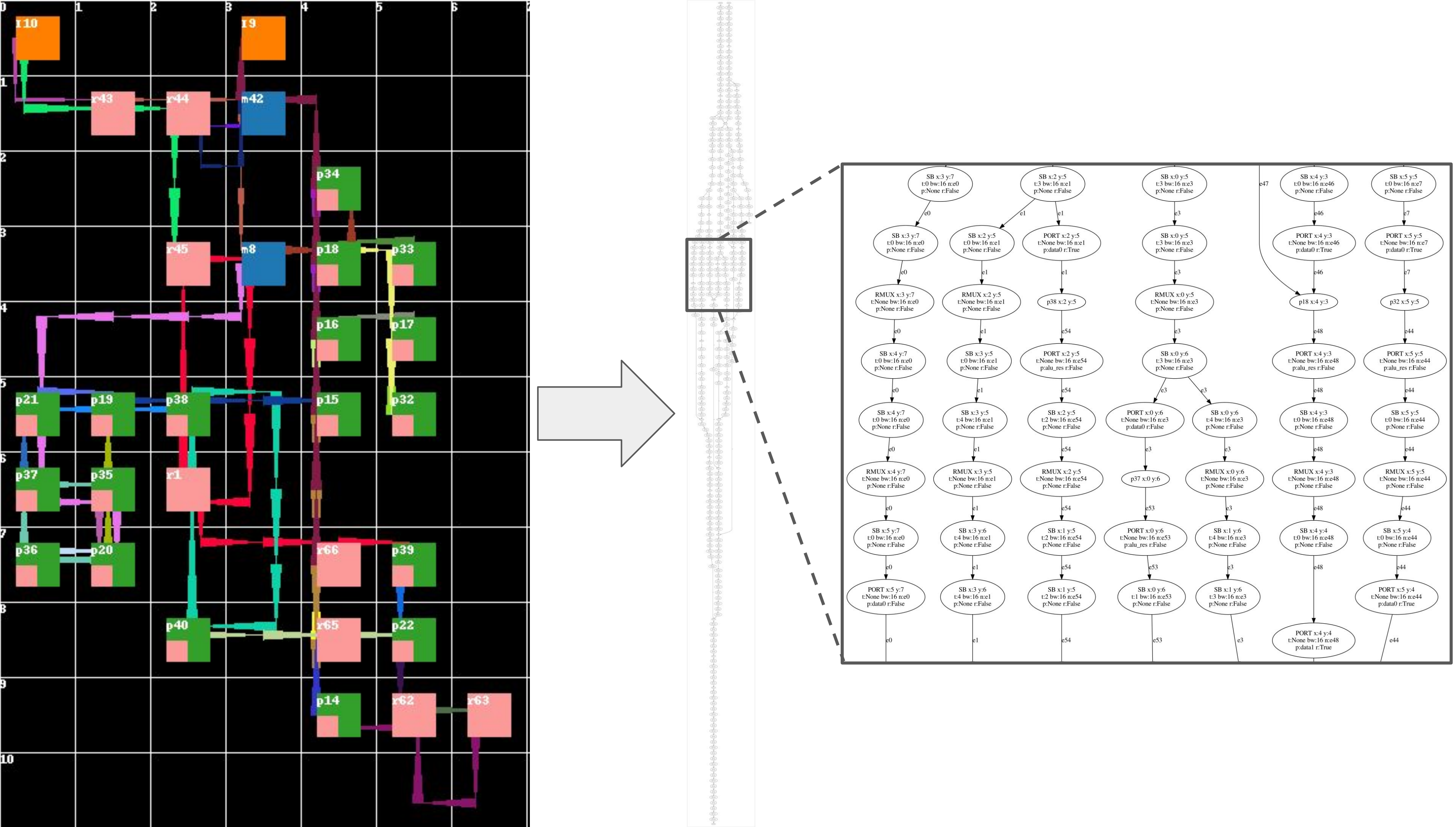
Critical Path Model



```
conv(x, y) += kernel(r.x, r.y) *
              input(x+r.x, y+r.y);
conv.in().compute_root();
conv.in()
    .tile(x,y,xo,yo,xi,yi,64,64)
    .hw_accelerate(xi, xo);
conv.update()
    .unroll(r.y, 3)
    .unroll(r.x, 3);
conv.compute_at(conv.in(), xo);
input.stream_to_accelerator();
```

Example: 3x3 Convolution

We must construct the delay model using the output of the place and route tool


0
1
2
3
4
5
6
7
8
9
10
I10
I9
r43
r44
m42
p34
r45
m8
p18
p33
p16
p17
p21
p19
p38
p15
p32
p37
p35
r1
p36
p20
r66
p39
p40
r65
p22
p14
r62
r63
SB x:3 y:7 t:0 bw:16 n:e0 p:None r:False
e0
SB x:3 y:7 t:0 bw:16 n:e0 p:None r:False
e0
RMUX x:3 y:7 t:None bw:16 n:e0 p:None r:False
e0
SB x:4 y:7 t:0 bw:16 n:e0 p:None r:False
e0
SB x:4 y:7 t:0 bw:16 n:e0 p:None r:False
e0
RMUX x:4 y:7 t:None bw:16 n:e0 p:None r:False
e0
SB x:5 y:7 t:0 bw:16 n:e0 p:None r:False
e0
PORT x:5 y:7 t:None bw:16 n:e0 p:data0 r:False
e0
SB x:2 y:5 t:3 bw:16 n:e1 p:None r:False
e1
e1
SB x:2 y:5 t:0 bw:16 n:e1 p:None r:False
e1
RMUX x:2 y:5 t:None bw:16 n:e1 p:None r:False
e1
SB x:3 y:5 t:0 bw:16 n:e1 p:None r:False
e1
SB x:3 y:5 t:4 bw:16 n:e1 p:None r:False
e1
RMUX x:3 y:5 t:None bw:16 n:e1 p:None r:False
e1
SB x:3 y:6 t:4 bw:16 n:e1 p:None r:False
e1
SB x:3 y:6 t:4 bw:16 n:e1 p:None r:False
e1
PORT x:2 y:5 t:None bw:16 n:e1 p:data0 r:True
e1
p38 x:2 y:5
e54
PORT x:2 y:5 t:None bw:16 n:e54 p:alu_res r:False
e54
SB x:2 y:5 t:2 bw:16 n:e54 p:None r:False
e54
RMUX x:2 y:5 t:None bw:16 n:e54 p:None r:False
e54
SB x:1 y:5 t:2 bw:16 n:e54 p:None r:False
e54
SB x:1 y:5 t:2 bw:16 n:e54 p:None r:False
e54
SB x:0 y:5 t:3 bw:16 n:e3 p:None r:False
e3
SB x:0 y:5 t:3 bw:16 n:e3 p:None r:False
e3
RMUX x:0 y:5 t:None bw:16 n:e3 p:None r:False
e3
SB x:0 y:6 t:3 bw:16 n:e3 p:None r:False
e3
e3
PORT x:0 y:6 t:None bw:16 n:e3 p:data0 r:False
e3
p37 x:0 y:6
e53
PORT x:0 y:6 t:None bw:16 n:e53 p:alu_res r:False
e53
SB x:0 y:6 t:1 bw:16 n:e53 p:None r:False
e53
SB x:0 y:6 t:4 bw:16 n:e3 p:None r:False
e3
RMUX x:0 y:6 t:None bw:16 n:e3 p:None r:False
e3
SB x:1 y:6 t:4 bw:16 n:e3 p:None r:False
e3
SB x:1 y:6 t:3 bw:16 n:e3 p:None r:False
e3
e47
SB x:4 y:3 t:0 bw:16 n:e46 p:None r:False
e46
PORT x:4 y:3 t:None bw:16 n:e46 p:data0 r:True
e46
p18 x:4 y:3
e48
PORT x:4 y:3 t:None bw:16 n:e48 p:alu_res r:False
e48
SB x:4 y:3 t:0 bw:16 n:e48 p:None r:False
e48
RMUX x:4 y:3 t:None bw:16 n:e48 p:None r:False
e48
SB x:4 y:4 t:0 bw:16 n:e48 p:None r:False
e48
PORT x:4 y:4 t:None bw:16 n:e48 p:data1 r:True
SB x:5 y:5 t:0 bw:16 n:e7 p:None r:False
e7
PORT x:5 y:5 t:None bw:16 n:e7 p:data0 r:True
e7
p32 x:5 y:5
e44
PORT x:5 y:5 t:None bw:16 n:e44 p:alu_res r:False
e44
SB x:5 y:5 t:0 bw:16 n:e44 p:None r:False
e44
RMUX x:5 y:5 t:None bw:16 n:e44 p:None r:False
e44
SB x:5 y:4 t:0 bw:16 n:e44 p:None r:False
e44
PORT x:5 y:4 t:None bw:16 n:e44 p:data0 r:True
e44

Critical Path Model - Evaluation

Application	Modeled (MHz)	Measured (MHz)	% Difference
Gaussian v1	158	200	-21.0%
Gaussian v2	295	280	5.4%
Gaussian v3	515	420	22.6%
Gaussian v4	793	600	32.2%
Harris v1	30	25	20.0%
Harris v2	137	160	-14.4%
Harris v3	335	300	11.7%
Harris v4	373	360	3.6%

Critical Path Model - Evaluation

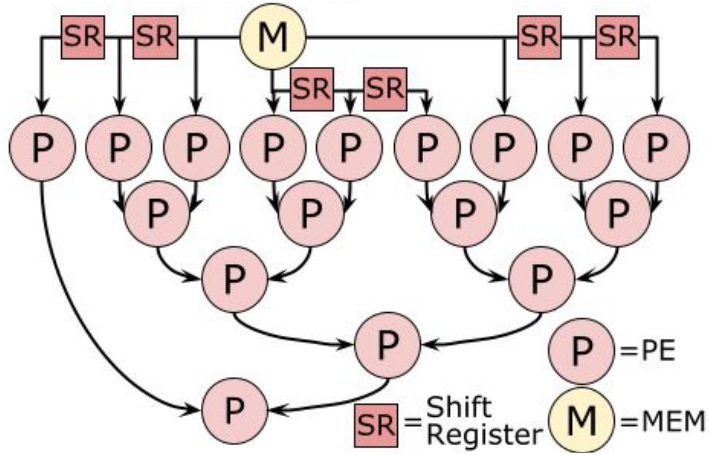
Application	Modeled (ns)	Measured (ns)	% Difference
Gaussian v1	6.33	5.00	26.6%
Gaussian v2	3.39	3.57	-5.1%
Gaussian v3	1.94	2.38	-18.4%
Gaussian v4	1.26	1.67	-24.3%
Harris v1	33.33	40.00	-16.7%
Harris v2	7.30	6.25	16.8%
Harris v3	2.99	3.33	-10.4%
Harris v4	2.68	2.78	-3.5%

Pipelining Techniques and Analysis

1. Compute Pipelining
2. Broadcast Pipelining
3. Placement Algorithm Tweaks
4. Post-PnR Pipelining
5. Register File Pipelining

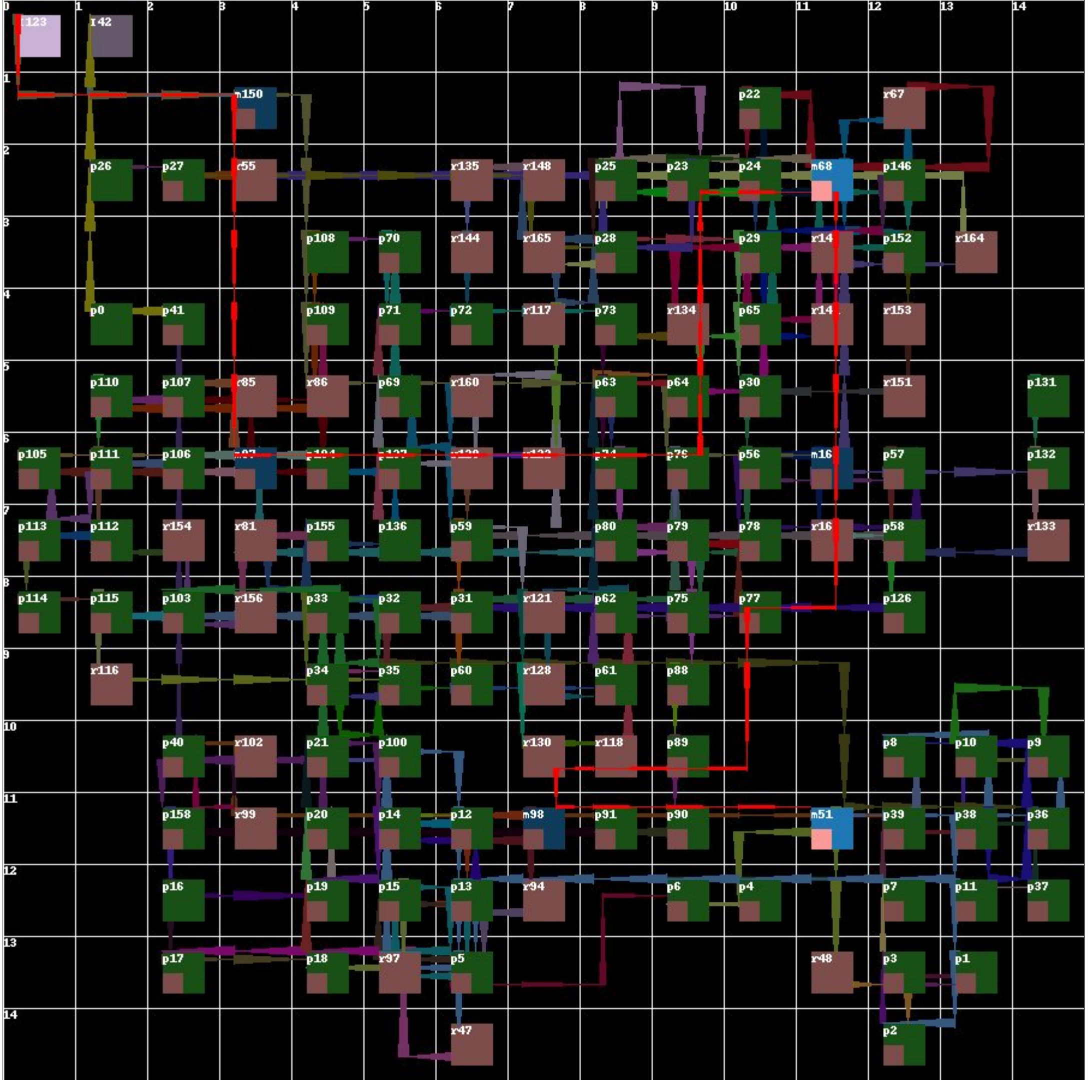
Compute Pipelining

- At compute mapping, we know how many PEs we will use and how they are connected
- Have all information needed to do branch delay matching
  - Ensures all paths from one memory to another are the same number of cycles
- All registers are added to the compute graph
  - Need to be packed into the PEs or placed onto the routing fabric

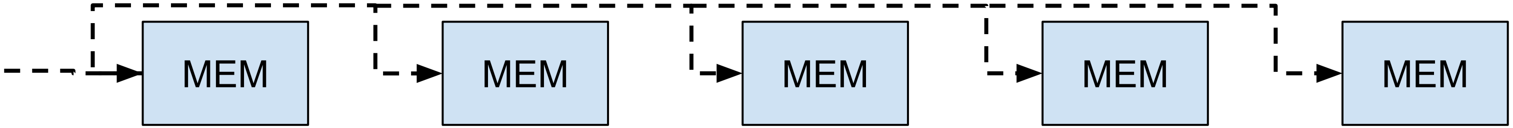


Compute Pipelining

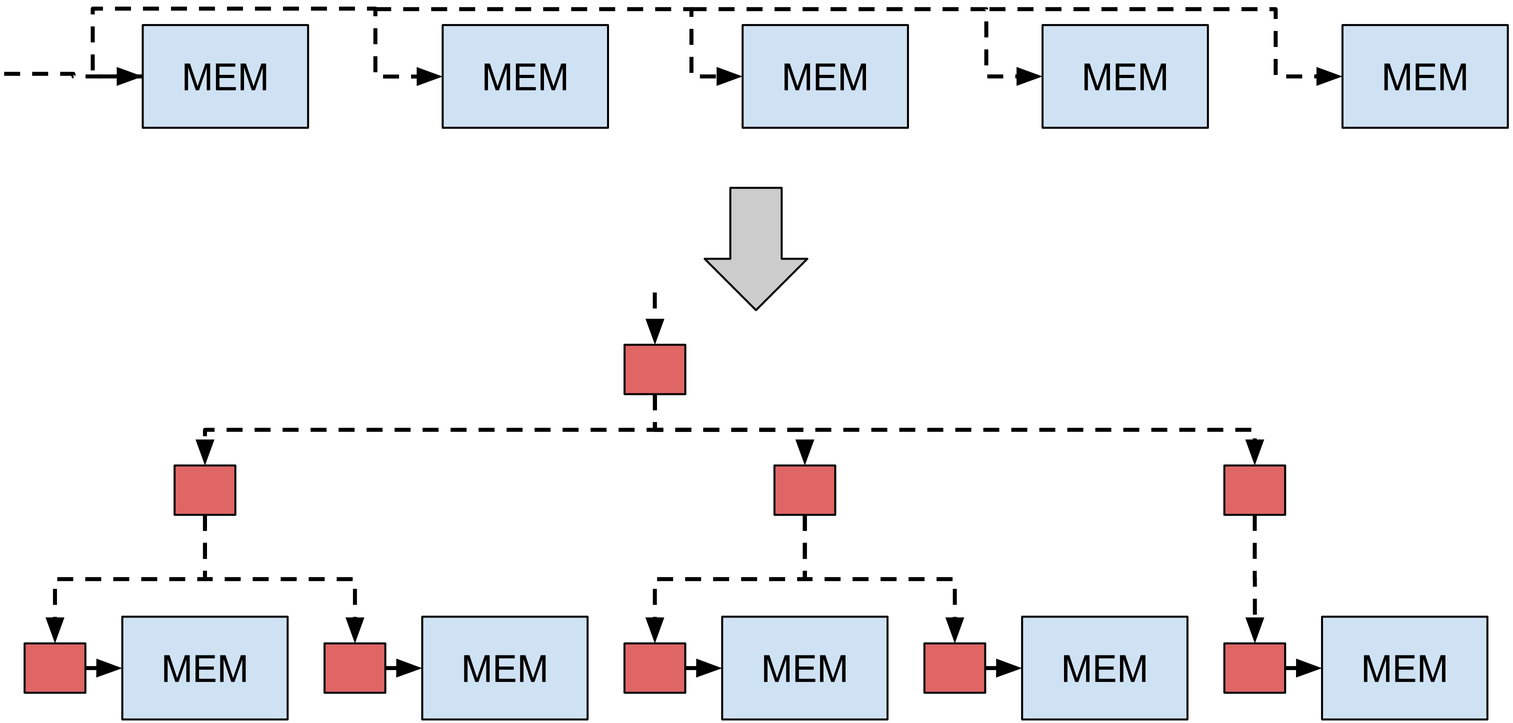
		Baseline	Compute Pipelining
Harris	Clk Freq (MHz)	26	147
	Resource Utilization (PE/Mem/Reg)	91/6/30	91/6/66
Camera Pipeline	Clk Freq (MHz)	17	32
	Resource Utilization (PE/Mem/Reg)	281/34/8	281/34/138
Gaussian	Clk Freq (MHz)	103	164
	Resource Utilization (PE/Mem/Reg)	160/8/136	160/8/160


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
r123
r42
m150
p22
r67
p26
p27
r55
r135
r148
p25
p23
p24
m68
p146
p108
p70
r144
r165
p28
p29
r14
p152
r164
p0
p41
p109
p71
p72
r117
p73
r134
p65
r14
r153
p110
p107
r85
r86
p69
r160
p63
p64
p30
r151
p131
p105
p111
p106
m97
p104
p137
r129
r122
p74
p76
p56
m16
p57
p132
p113
p112
r154
r81
p155
p136
p59
p80
p79
p78
r16
p58
r133
p114
p115
p103
r156
p33
p32
p31
r121
p62
p75
p77
p126
r116
p34
p35
p60
r128
p61
p88
p40
r102
p21
p100
r130
r118
p89
p8
p10
p9
p158
r99
p20
p14
p12
m98
p91
p90
m51
p39
p38
p36
p16
p19
p15
p13
r94
p6
p4
p7
p11
p37
p17
p18
r97
p5
r48
p3
p1
r47
p2

Broadcast Signal Pipelining

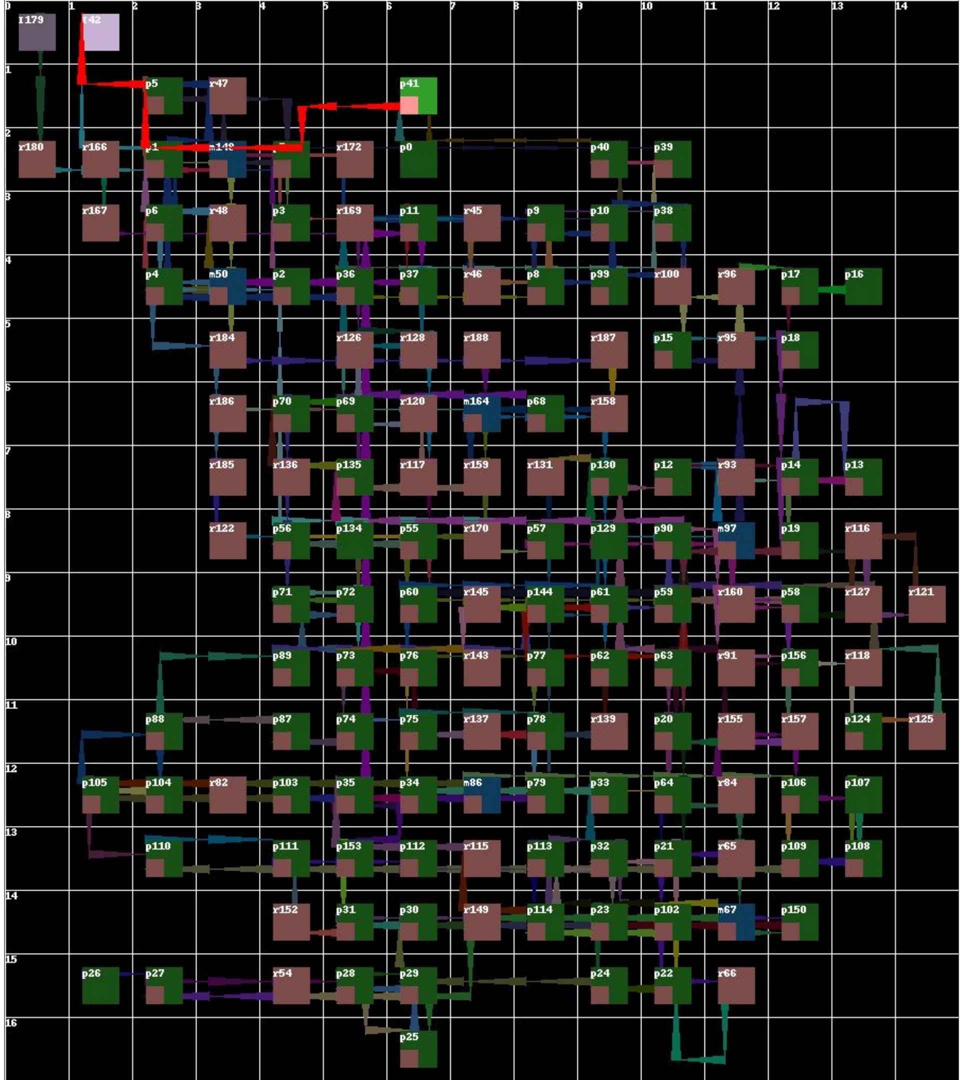


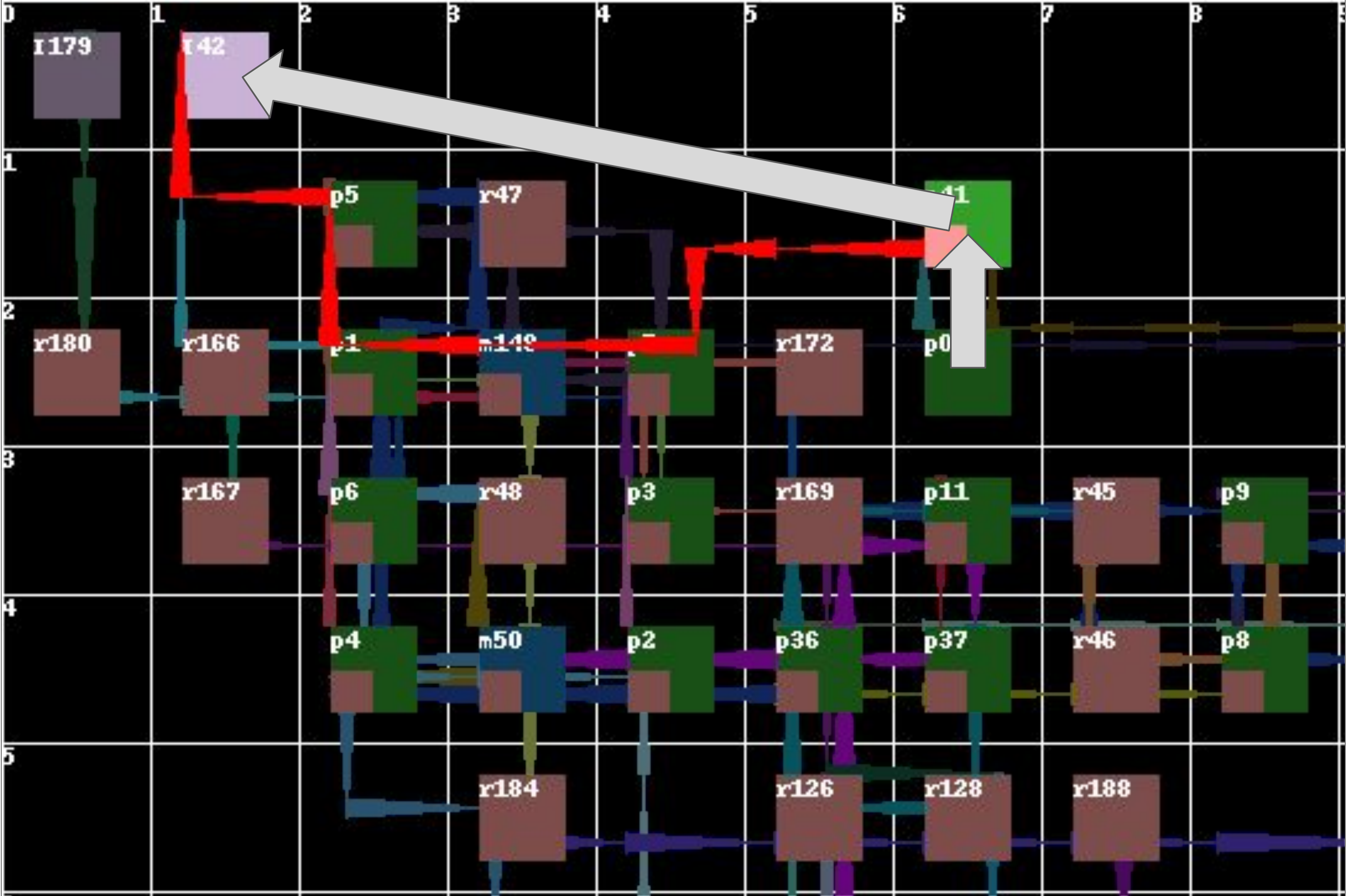
Broadcast Signal Pipelining



Broadcast Signal Pipelining

		Baseline	Compute Pipelining	Broadcast Pipelining
Harris	Clk Freq (MHz)	26	147	370
	Resource Utilization (PE/Mem/Reg)	91/6/30	91/6/66	91/6/86
Camera Pipeline	Clk Freq (MHz)	17	32	84
	Resource Utilization (PE/Mem/Reg)	281/34/8	281/34/138	281/34/206
Gaussian	Clk Freq (MHz)	103	164	291
	Resource Utilization (PE/Mem/Reg)	160/8/136	160/8/160	160/8/211

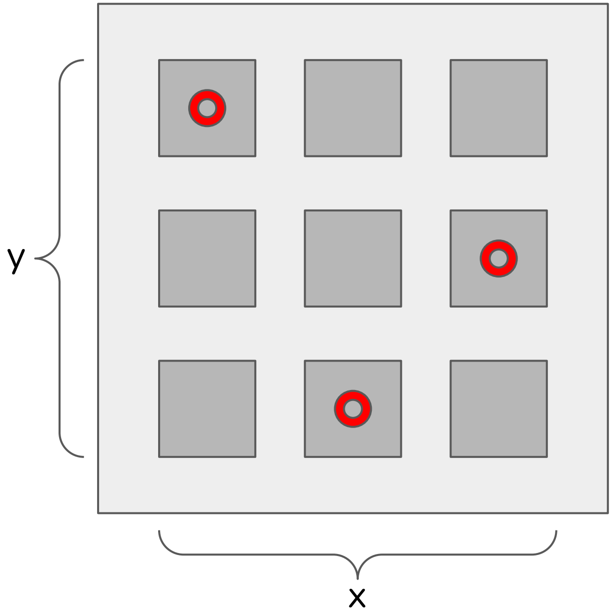

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
r179
r42
p5
r47
p41
r180
r166
p1
m149
r172
p0
p40
p39
r167
p6
r48
p3
r169
p11
r45
p9
p10
p38
p4
m50
p2
p36
p37
r46
p8
p99
r100
r96
p17
p16
r184
r126
r128
r188
r187
p15
r95
p18
r186
p70
p69
r120
m164
p68
r158
r185
r136
p135
r117
r159
r131
p130
p12
r93
p14
p13
r122
p56
p134
p55
r170
p57
p129
p90
m97
p19
r116
p71
p72
p60
r145
p144
p61
p59
r160
p58
r127
r121
p89
p73
p76
r143
p77
p62
p63
r91
p156
r118
p88
p87
p74
p75
r137
p78
r139
p20
r155
r157
p124
r125
p105
p104
r82
p103
p35
p34
m86
p79
p33
p64
r84
p106
p107
p110
p111
p153
p112
r115
p113
p32
p21
r65
p109
p108
r152
p31
p30
r149
p114
p23
p102
m67
p150
p26
p27
r54
p28
p29
p24
p22
r66
p25


0
1
2
3
4
5
6
7
8
1
2
3
4
5
r179
r42
p5
r47
41
r180
r166
p1
m149
r172
p0
r167
p6
r48
p3
r169
p11
r45
p9
p4
m50
p2
p36
p37
r46
p8
r184
r126
r128
r188

Placement Algorithm Tweaks

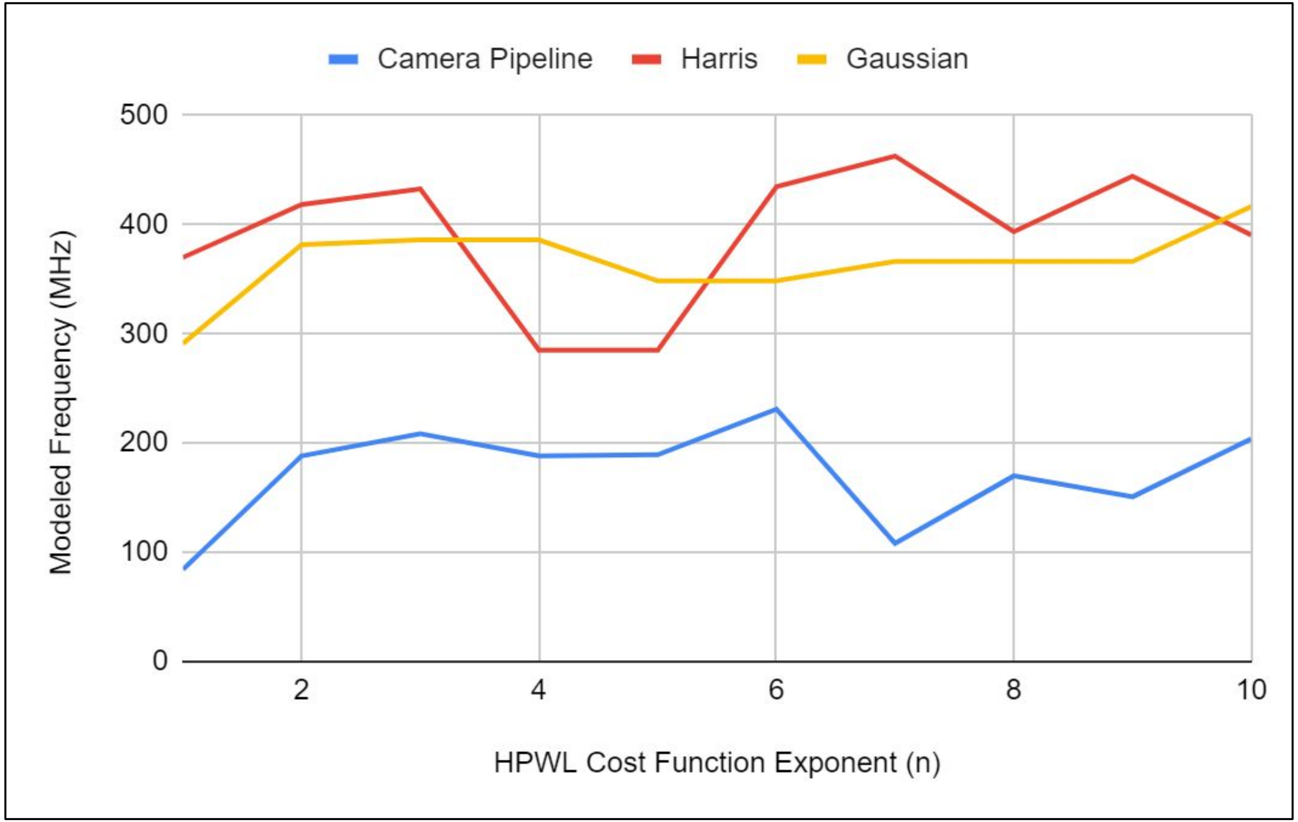
- The cost metric for placement is total wirelength
  - Generally leads to good routability

- For pipelining we care about max wirelength, not total wirelength
  - Using max wirelength alone as a cost metric for placement leads to routability issues
  - Instead we can create a new cost metric that incorporates both maximum and total wirelength
  - Total wirelength^n for n = [1...10] works well



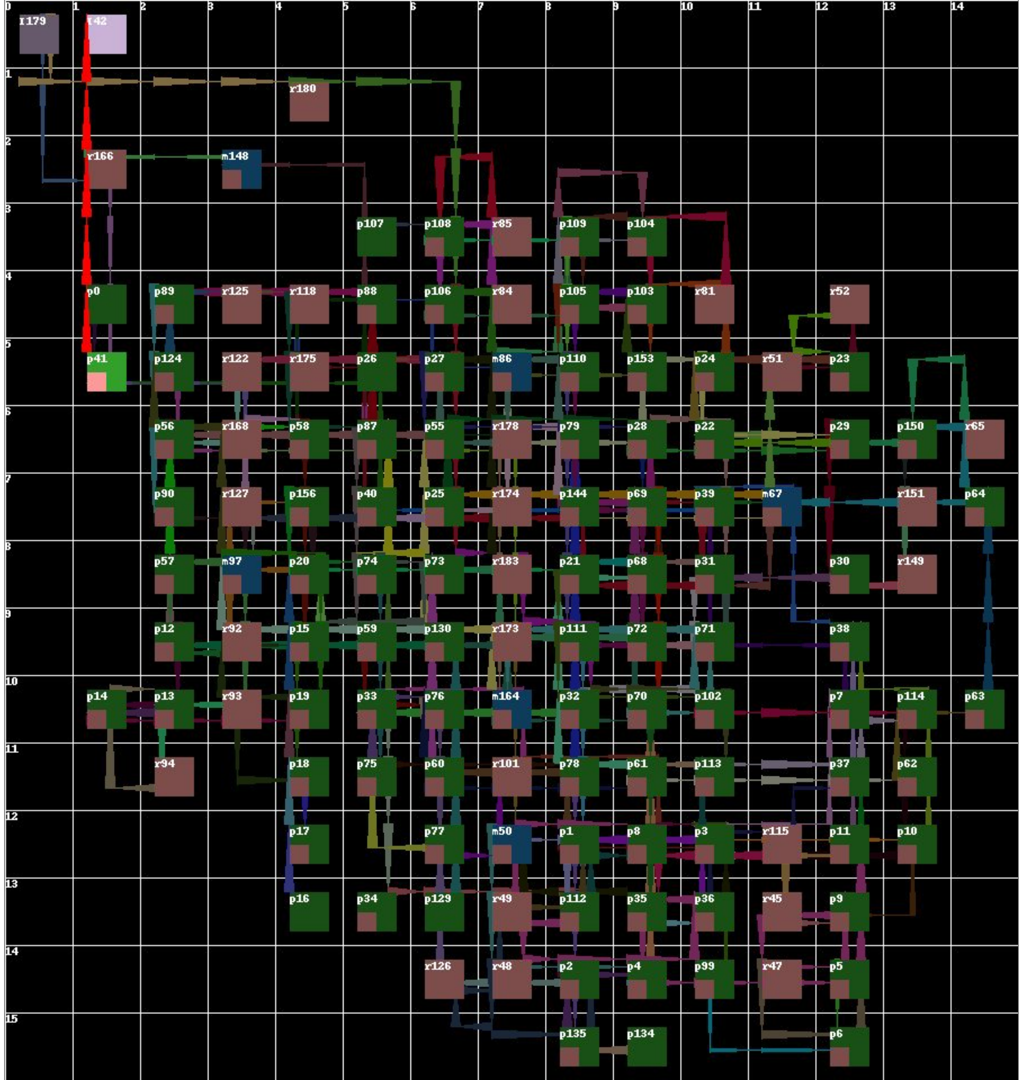
half-perimeter wirelength = x+y

Placement Algorithm Exponent Parameter Sweep



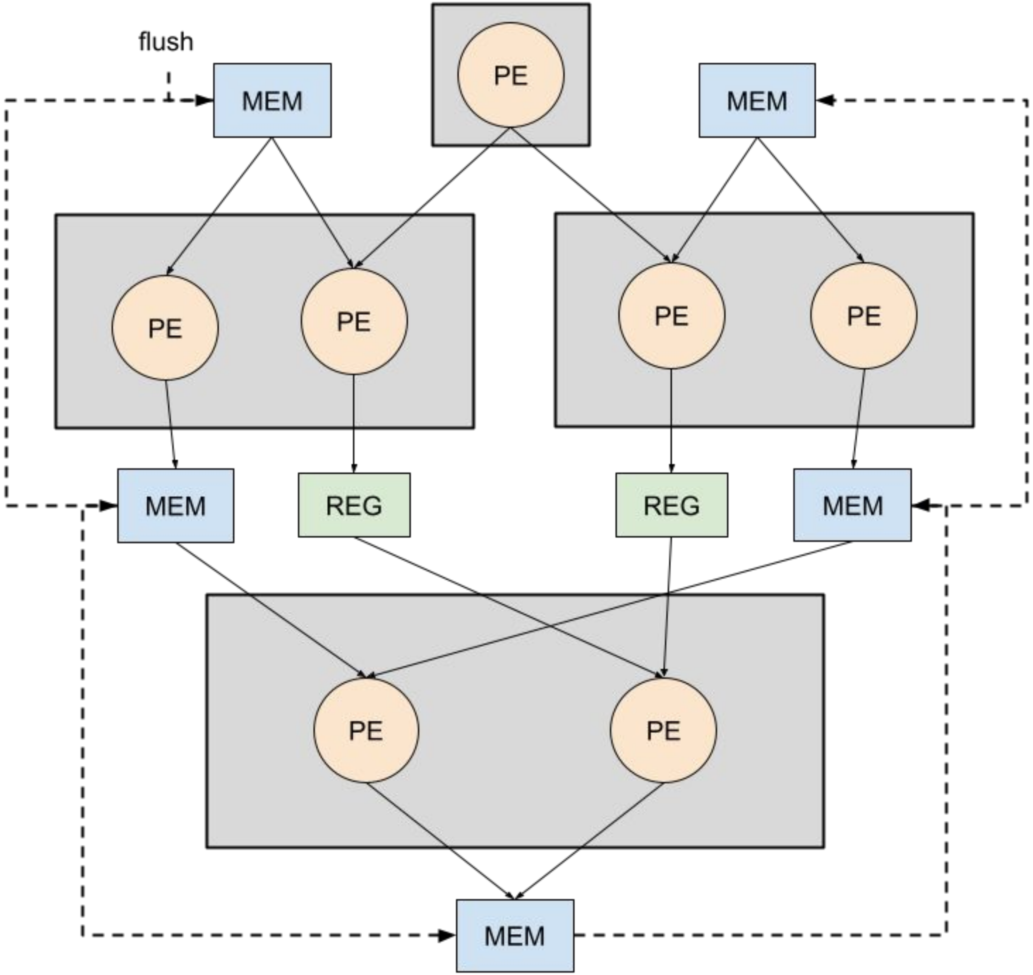
Placement Algorithm Tweaks

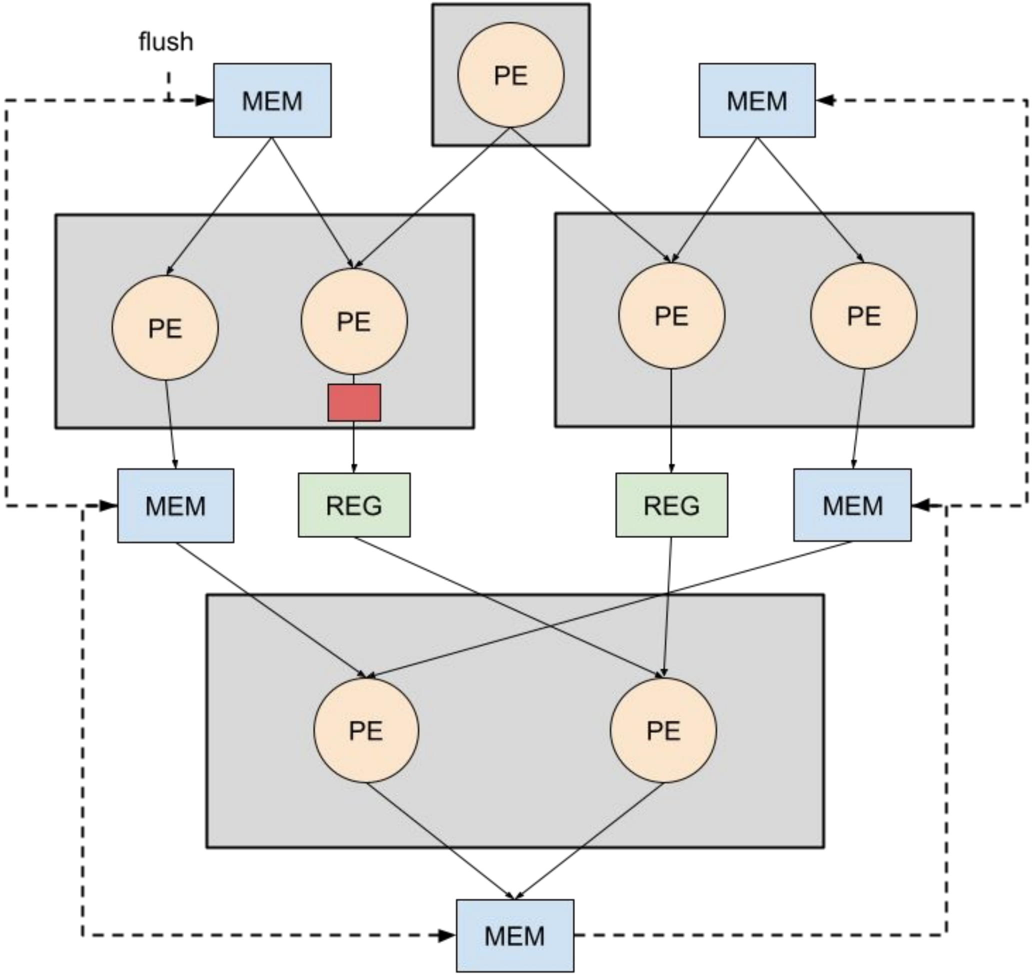
		Baseline	Compute Pipelining	Broadcast Pipelining	Placement Tweaks
Harris	Clk Freq (MHz)	26	147	370	444
	Resource Utilization (PE/Mem/Reg)	91/6/30	91/6/66	91/6/86	91/6/86
Camera Pipeline	Clk Freq (MHz)	17	32	84	231
	Resource Utilization (PE/Mem/Reg)	281/34/8	281/34/138	281/34/206	281/34/206
Gaussian	Clk Freq (MHz)	103	164	291	417
	Resource Utilization (PE/Mem/Reg)	160/8/136	160/8/160	160/8/211	160/8/211

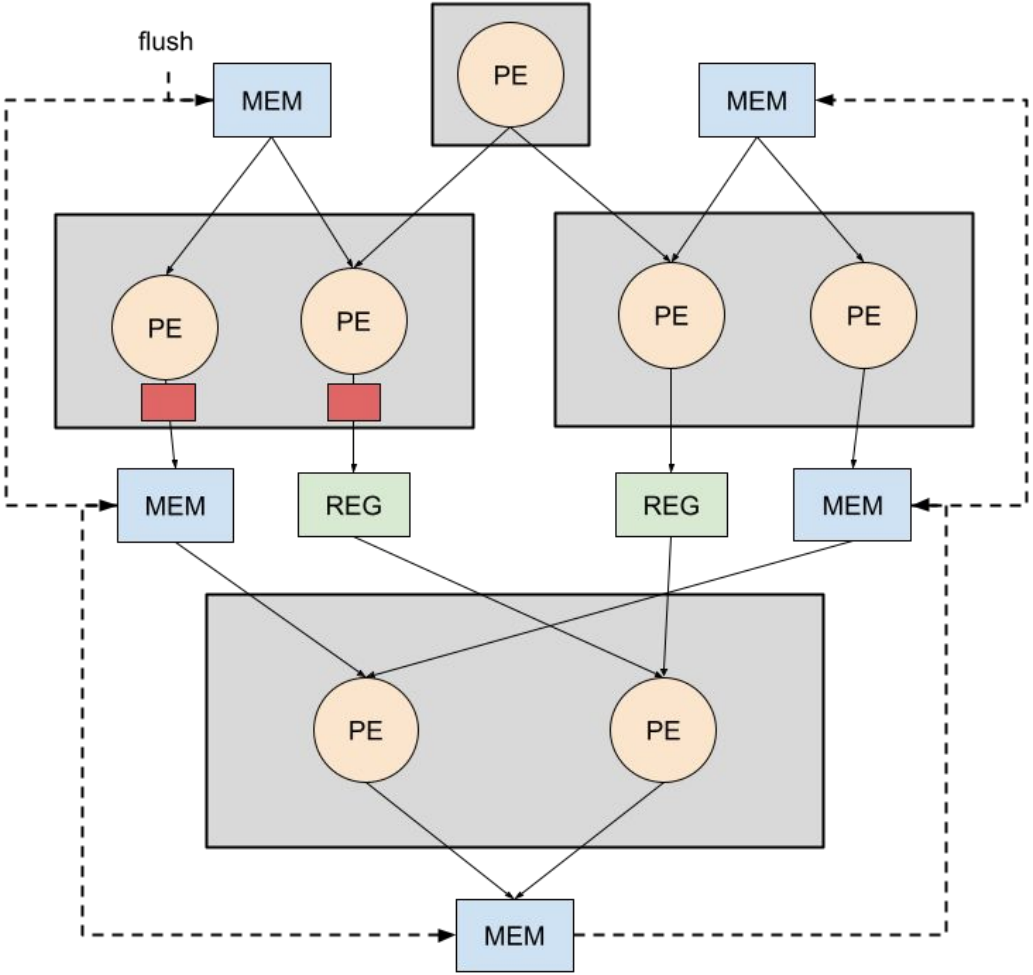

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
r179
r42
r180
r166
m148
p107
p108
r85
p109
p104
p0
p89
r125
r118
p88
p106
r84
p105
p103
r81
r52
p41
p124
r122
r175
p26
p27
m86
p110
p153
p24
r51
p23
p56
r168
p58
p87
p55
r178
p79
p28
p22
p29
p150
r65
p90
r127
p156
p40
p25
r174
p144
p69
p39
m67
r151
p64
p57
m97
p20
p74
p73
r183
p21
p68
p31
p30
r149
p12
r92
p15
p59
p130
r173
p111
p72
p71
p38
p14
p13
r93
p19
p33
p76
m164
p32
p70
p102
p7
p114
p63
r94
p18
p75
p60
r101
p78
p61
p113
p37
p62
p17
p77
m50
p1
p8
p3
r115
p11
p10
p16
p34
p129
r49
p112
p35
p36
r45
p9
r126
r48
p2
p4
p99
r47
p5
p135
p134
p6

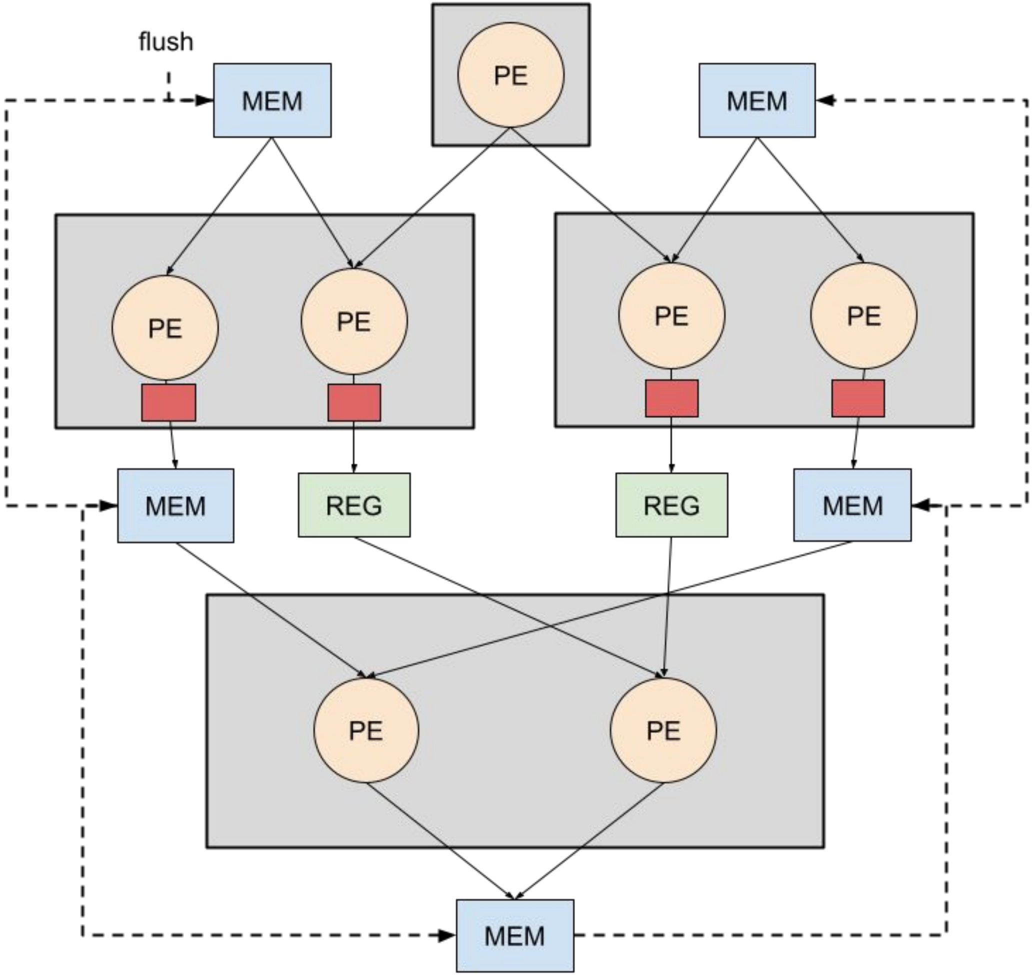
Post Place and Route Pipelining

- Iteratively break the critical path determined by the STA tool
- Re-analyze and determine new critical path
- Continue until there are no more registers available to use on the interconnect
  - Adding pipelining registers to a placement result is not always possible because branch delay matching is required


flush
MEM
PE
MEM
PE
PE
PE
PE
MEM
REG
REG
MEM
PE
PE
MEM

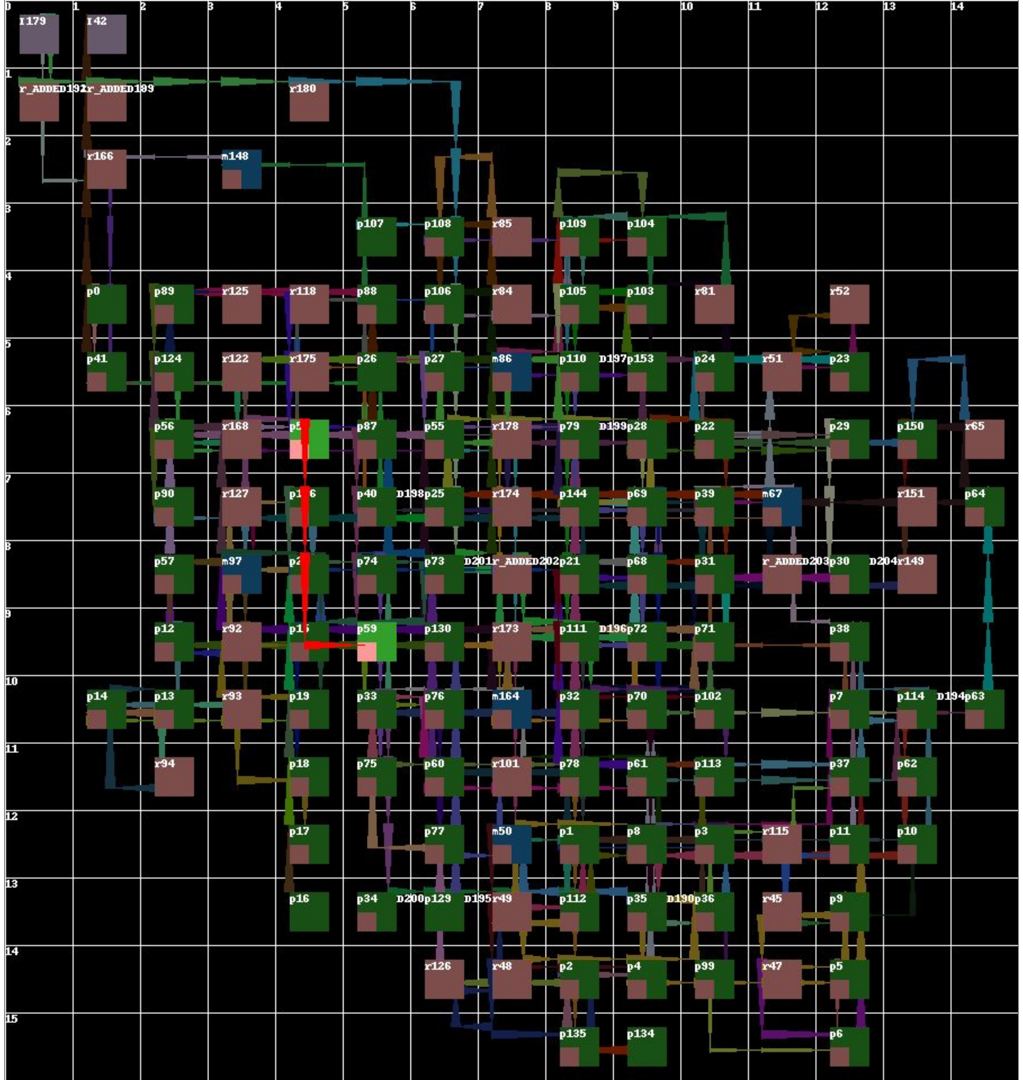

flush
MEM
PE
MEM
PE
PE
PE
PE
MEM
REG
REG
MEM
PE
PE
MEM


flush
PE
MEM
MEM
PE
PE
PE
PE
MEM
REG
REG
MEM
PE
PE
MEM


flush
MEM
PE
MEM
PE
PE
PE
PE
MEM
REG
REG
MEM
PE
PE
MEM

Post Place and Route Pipelining

		Baseline	Compute Pipelining	Broadcast Pipelining	Placement Tweaks	PnR Pipelining
Harris	Clk Freq (MHz)	26	147	370	444	706
	Resource Utilization (PE/Mem/Reg)	91/6/30	91/6/66	91/6/86	91/6/86	91/6/102
Camera Pipeline	Clk Freq (MHz)	17	32	84	231	231
	Resource Utilization (PE/Mem/Reg)	281/34/8	281/34/138	281/34/206	281/34/206	281/34/206
Gaussian	Clk Freq (MHz)	103	164	291	417	417
	Resource Utilization (PE/Mem/Reg)	160/8/136	160/8/160	160/8/211	160/8/211	160/8/211

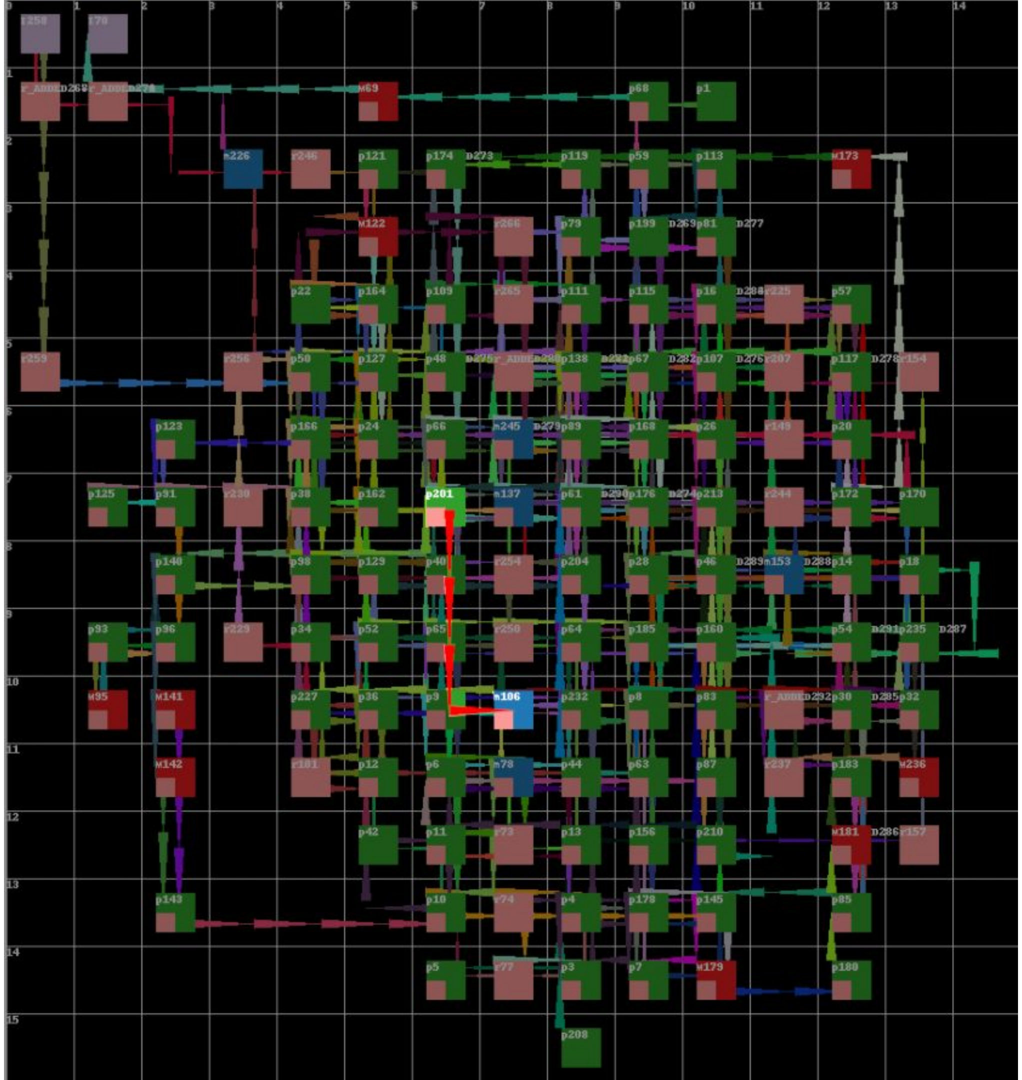

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
I179
I42
r_ADDED192
r_ADDED189
r180
r166
m148
p107
p108
r85
p109
p104
p0
p89
r125
r118
p88
p106
r84
p105
p103
r81
r52
p41
p124
r122
r175
p26
p27
m86
p110
D197
p153
p24
r51
p23
p56
r168
p5
p87
p55
r178
p79
D199
p28
p22
p29
p150
r65
p90
r127
p1 6
p40
D198
p25
r174
p144
p69
p39
m67
r151
p64
p57
m97
p2
p74
p73
D201
r_ADDED202
p21
p68
p31
r_ADDED203
p30
D204
r149
p12
r92
p1
p59
p130
r173
p111
D196
p72
p71
p38
p14
p13
r93
p19
p33
p76
m164
p32
p70
p102
p7
p114
D194
p63
r94
p18
p75
p60
r101
p78
p61
p113
p37
p62
p17
p77
m50
p1
p8
p3
r115
p11
p10
p16
p34
D200
p129
D195
r49
p112
p35
D190
p36
r45
p9
r126
r48
p2
p4
p99
r47
p5
p135
p134
p6

Register File Pipelining

- We have a register file in every PE in the CGRA
- If we write and read every cycle to the same address, we can use this reg file as a pipeline register
- If we write and read to offset incrementing addresses, we can use it as a variable length pipeline register chain

Register File Pipelining

		Baseline	Compute Pipelining	Broadcast Pipelining	Placement Tweaks	PnR Pipelining	With Pipelining Ponds
Harris	Clk Freq (MHz)	26	147	370	444	706	706
	Resource Utilization (PE/Mem/Reg)	91/6/30	91/6/66	91/6/86	91/6/86	91/6/102	91/6/84
Camera Pipeline	Clk Freq (MHz)	17	32	84	231	231	264
	Resource Utilization (PE/Mem/Reg)	281/34/8	281/34/138	281/34/206	281/34/206	281/34/206	281/68/76
Gaussian	Clk Freq (MHz)	103	164	291	417	417	471
	Resource Utilization (PE/Mem/Reg)	160/8/136	160/8/160	160/8/211	160/8/211	160/8/211	160/8/187


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
r258
r70
r_ADDED267
r_ADDED271
M69
p68
p1
m226
r246
p121
p174
D273
p119
p59
p113
M173
M122
r266
p79
p199
D269
p81
D277
p22
p164
p109
r265
p111
p115
p16
D284
r225
p57
r259
r256
p50
p127
p48
D275
r_ADDED280
p138
D281
p67
D282
p107
D276
r207
p117
D278
r154
p123
p166
p24
p66
m245
D279
p89
p168
p26
r149
p20
p125
p91
r230
p38
p162
p201
m137
p61
D290
p176
D274
p213
r244
p172
p170
p140
p98
p129
p40
r254
p204
p28
p46
D289
m153
D288
p14
p18
p93
p96
r229
p34
p52
p65
r250
p64
p185
p160
p54
D291
p235
D287
M95
M141
p227
p36
p9
m106
p232
p8
p83
r_ADDED
D292
p30
D285
p32
M142
r181
p12
p6
m78
p44
p63
p87
r237
p183
M236
p42
p11
r73
p13
p156
p210
M181
D286
r157
p143
p10
r74
p4
p178
p145
p85
p5
r77
p3
p7
M179
p180
p208

Future Directions - Hardware Changes

- Using ready-valid signaling would make branch delay matching easier
  - Rarely need to add registers to branch delay match
  - FIFOs at tile inputs can be used to buffer data that arrives early

- Optimize global buffer to array path
  - Current path is by far the longest in the array
  - We would never be able to run applications at 1 GHz

Future Directions - Compiler Changes

- Make the scheduler more adaptable to different delays
  - Our pipelining tools are currently over constrained
  - The limiting factor when adding registers post-pnr is branch delay matching

- Improve the place and route tool
  - There is too much variability in the max wirelength from run to run
  - In the current PnR tool, routability as the primary concern
  - We can add more emphasis on reducing maximum wirelength
  - Fix the global placement algorithm and add congestion estimation