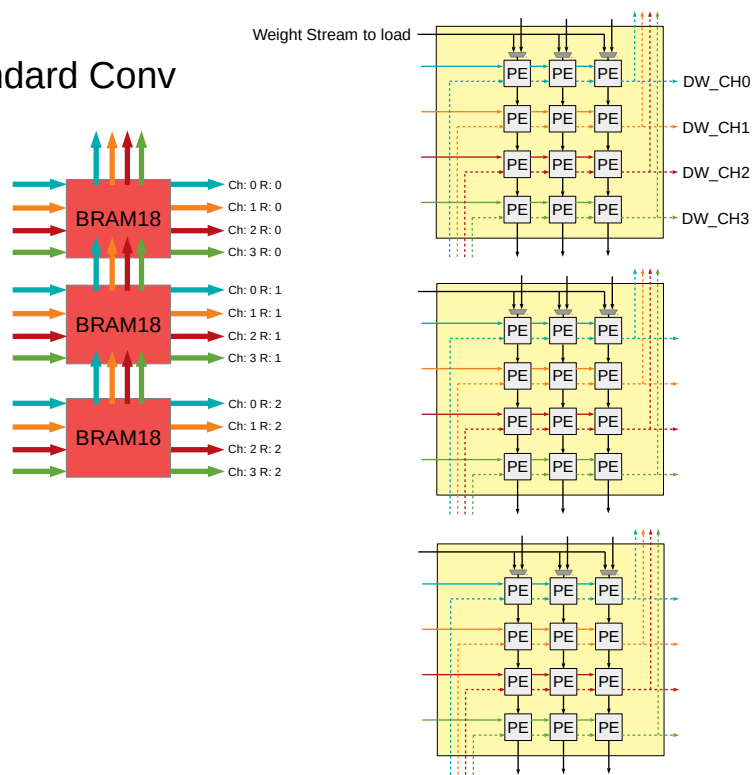
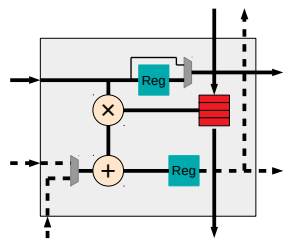


MLBlocks: Arming FPGA architectures with Dense & Low Precision units in classic column based manner

DWConv & Standard Conv



PE



Contributions:

1- Reconfigurable PE (**DSP size**, Systolic array, **Column based**)
6 times more 8x8 multiplier comparing to a DSP Block (two 8x8),

RS Data flow, High frequency, flexible data movement. Great for **SConv, DWConv, PWConv, Matrix-Matrix Multiplication**

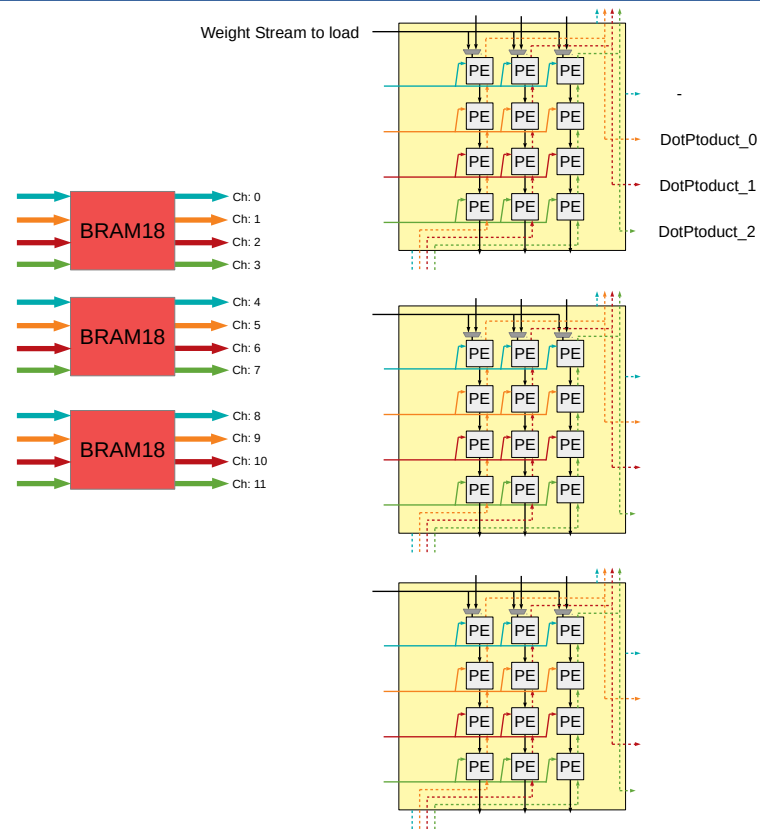
DPS – BRAM ratio 1/1 (same as Ultrascale+ arch), Low number of intermediary outputs in practice

Parameterized (for any budget limitation) – can integrate multi precision idea

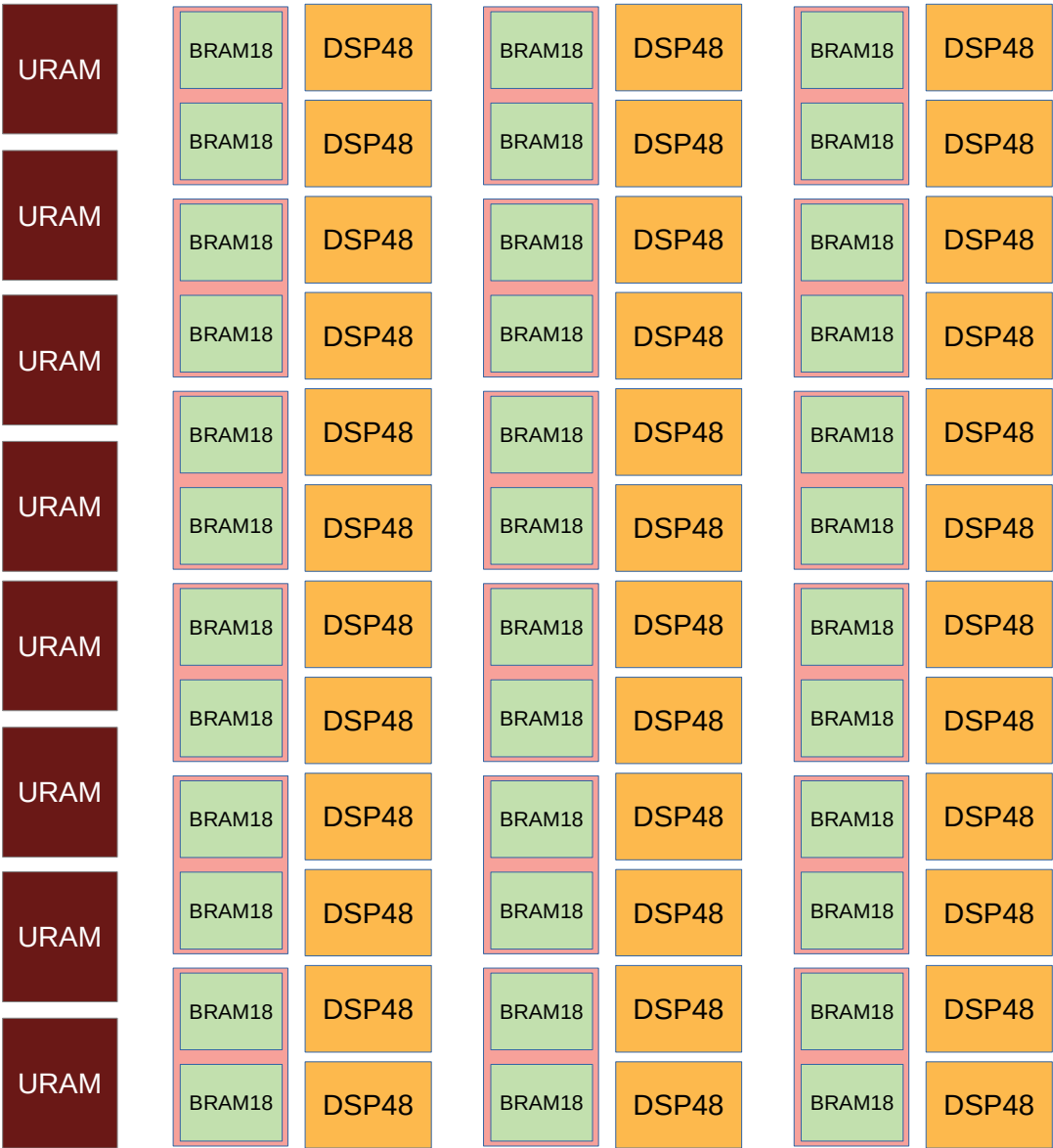
2- Compare with cascade paper (Prof. Nachiket)

3- new suggestion to use each 18KBRAM as 36bit streamer using external controller circuit (delivering 662MHz) (in cascade paper: 18bit)

PWConv & Matrix-Matrix Multiplication



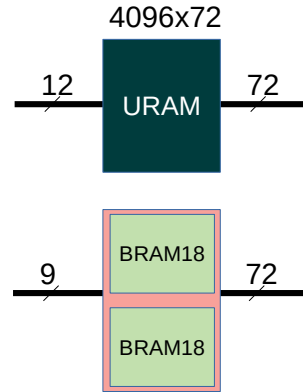
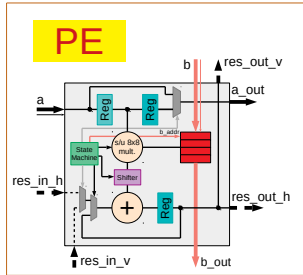
UltraScale+ architecture distribution:



Virtex 7 → **28nm** DSP48**E1** Fmax=742MHz

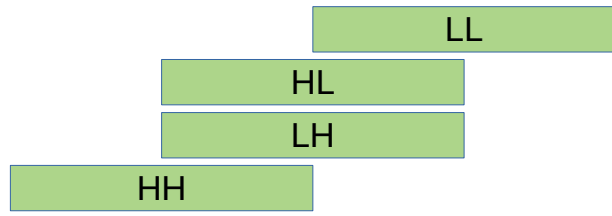
Virtex U → 20nm

Virtex UP → 16nm



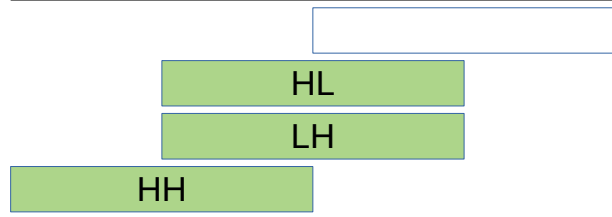
Normal

H	L
H	L



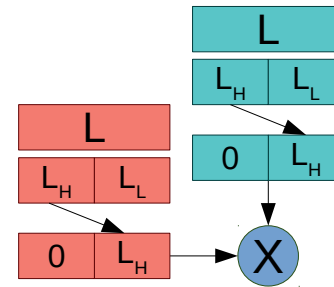
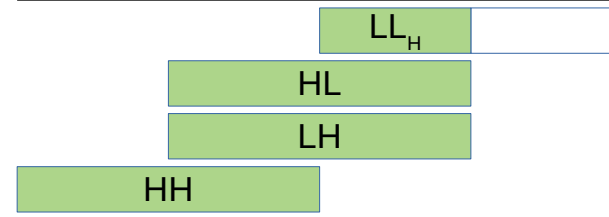
David's
idea

H	L
H	L

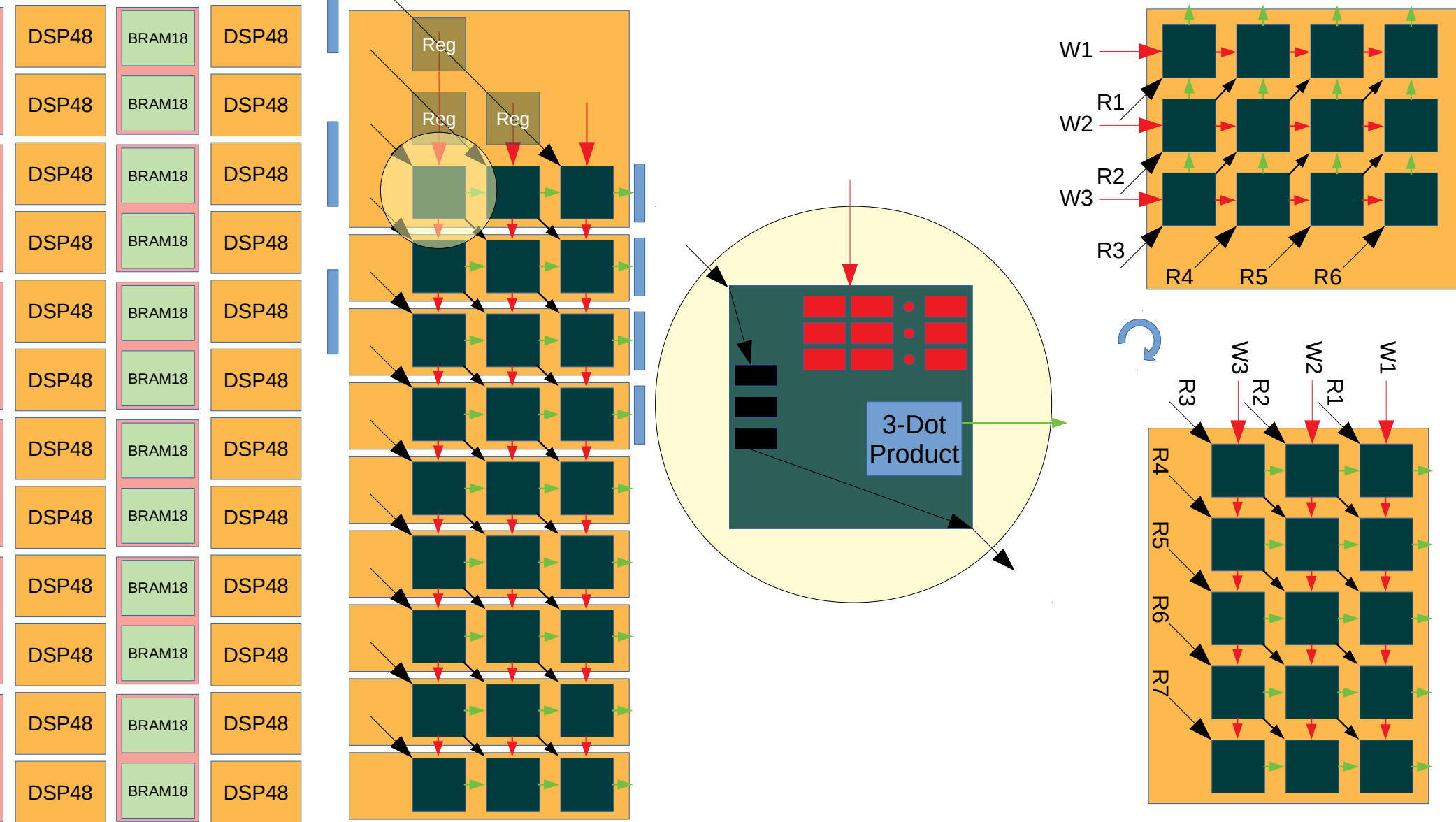


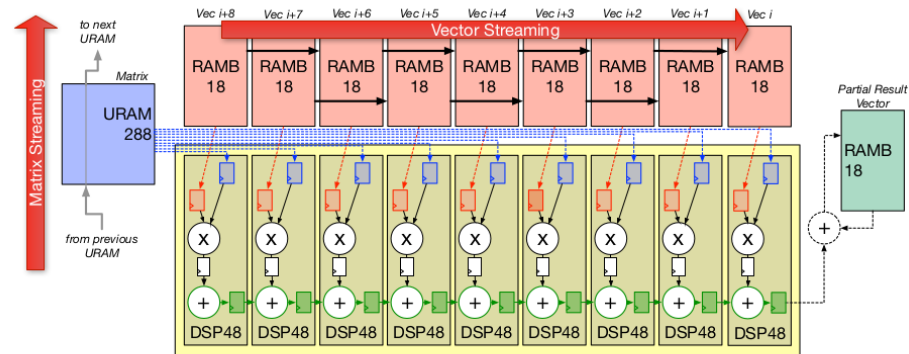
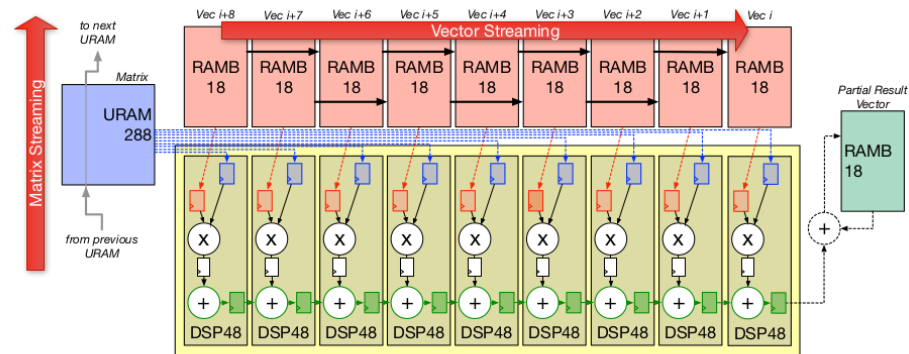
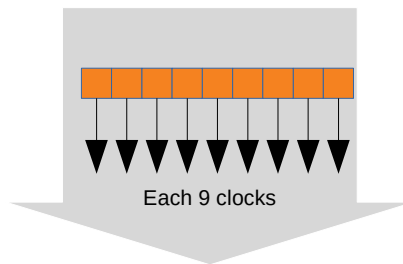
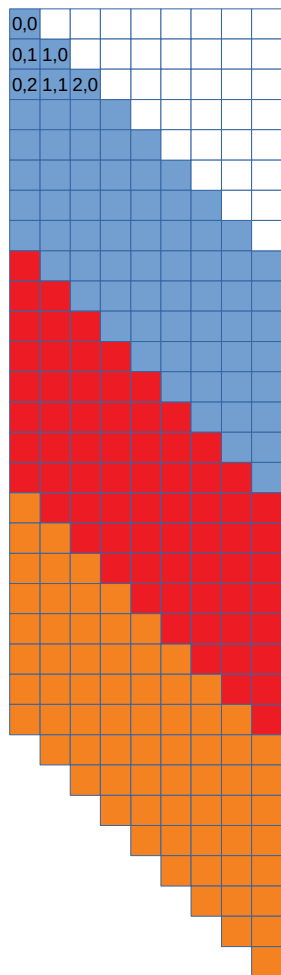
David's
idea V2

H	L
H	L



My amassing
MLBlocks world



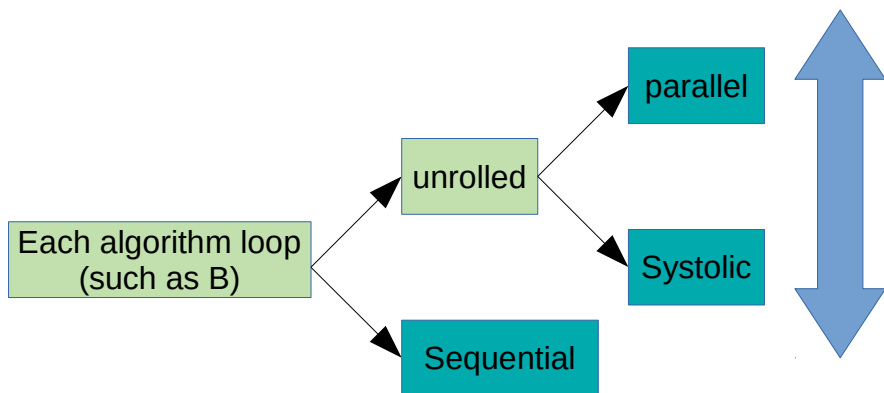


DSP48
P cascade

RAMB18
cascade

URAM288
cascade

General Purpose
Interconnect



Dis Parallel:

1- more fan in and outs (since we are talking about small Pes it is fine)

Dis Systolic:

1- tougher scheduling, rhythmic scheduling

2- prevent circuit fusions (less optimization)

$$B = B_{\text{Seq}} \times B_{\text{par}} \times B_{\text{Sys}}$$

of Physical MAC: $\times B_{\text{par}} \times B_{\text{Sys}}$

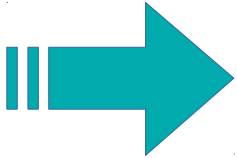
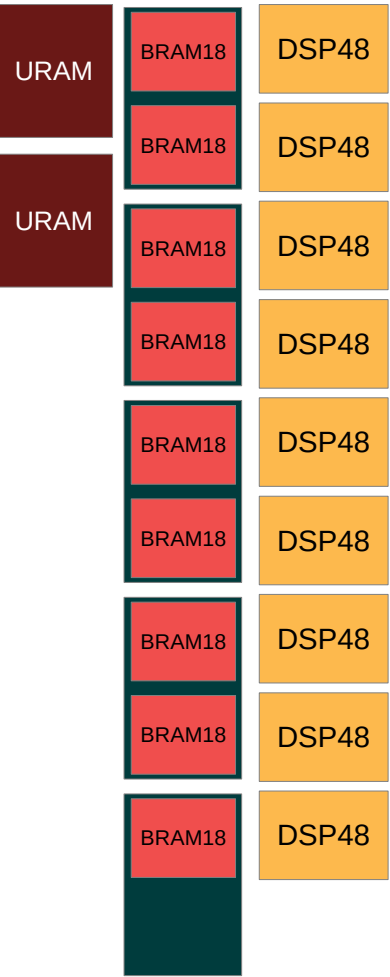
of Input: $\times B_{\text{par}}$

of Output: $\times B_{\text{par}}$

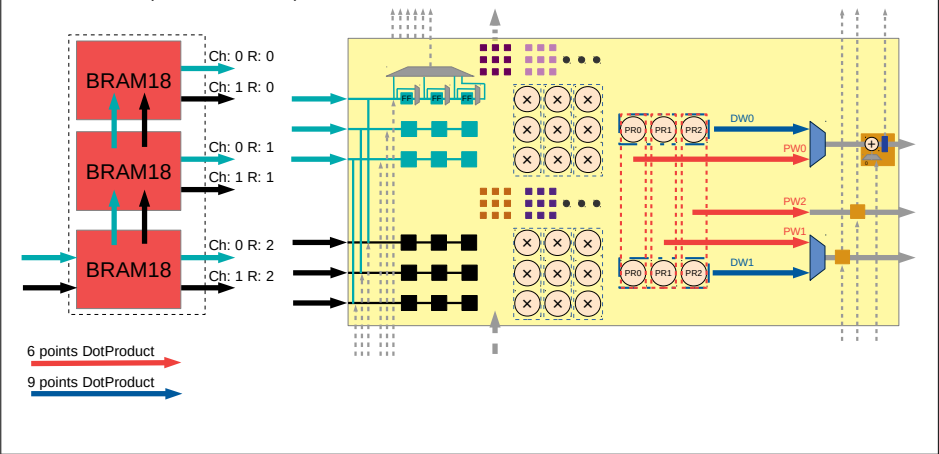
(without internal serial to parallel)

Params = {right side indexes}

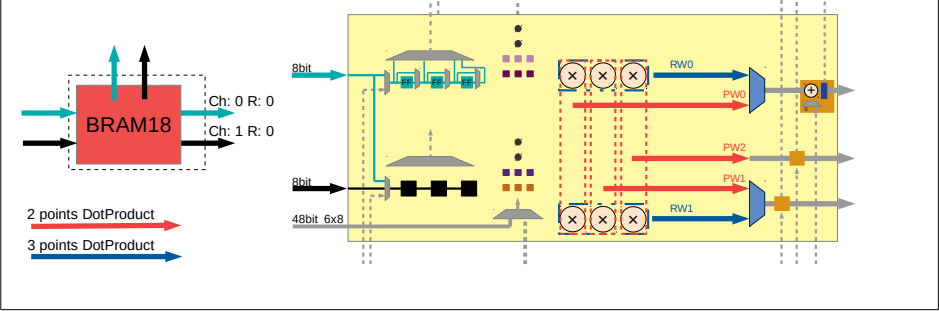
```
for paramsch_0i: 0 → sch0i
  for paramsch_1i: 0 → sch1i
    for paramsch_2i: 0 → sch2i
      for paramseqi: 0 → comp_seqi
        for paramuni: 0 → comp_uni
```

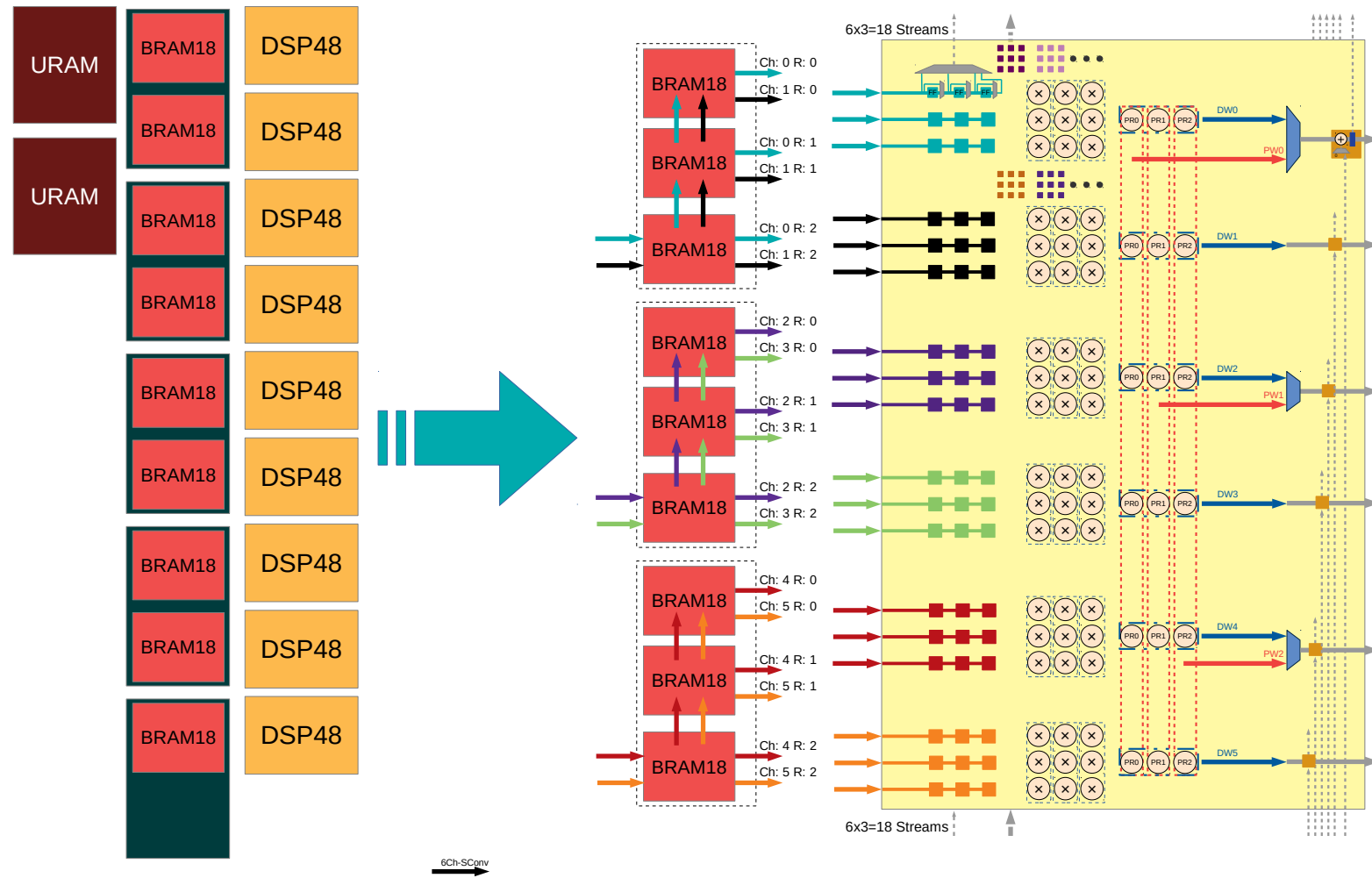


Three DSP-size (three-BRAM size)

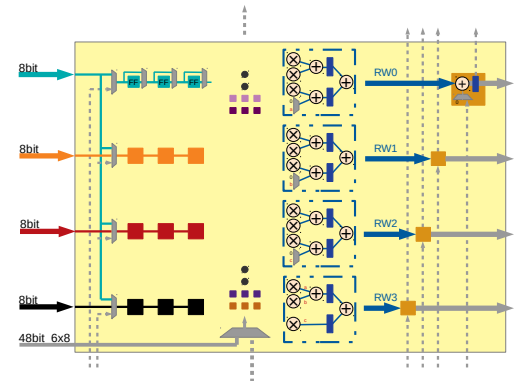
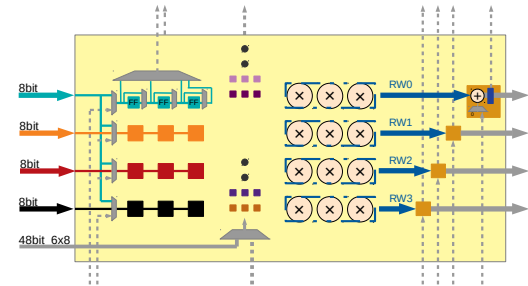
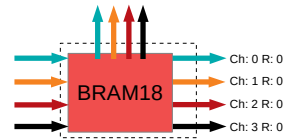
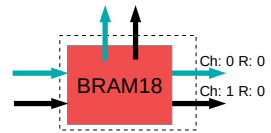


DSP-size (BRAM-size) (simplified PIR-DSP)

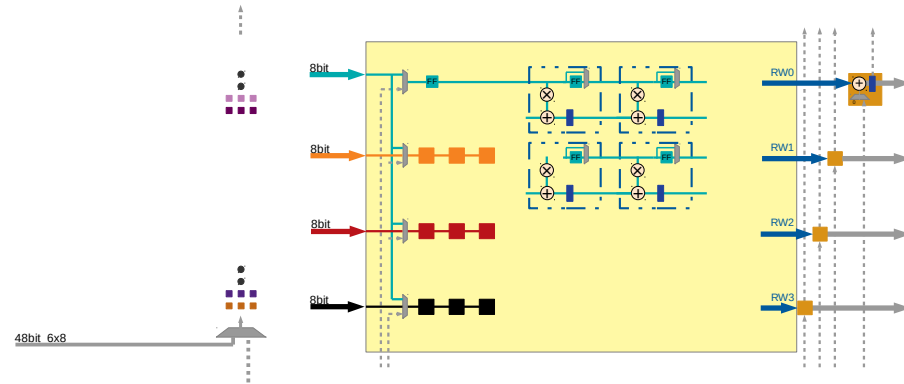
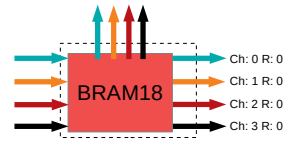
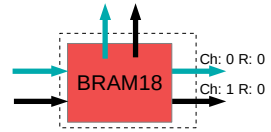




DSP-size (BRAM-size)



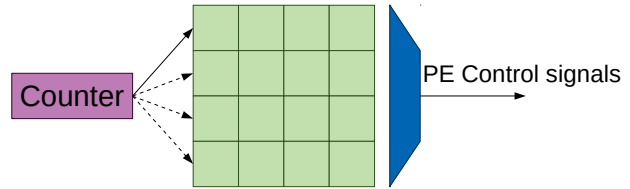
DSP-size (BRAM-size)



Primary Results

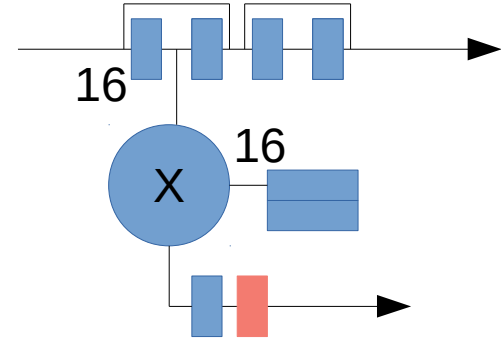
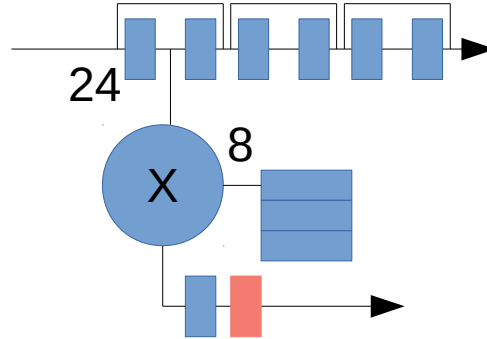
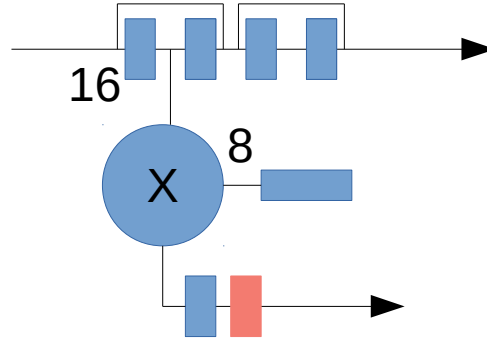
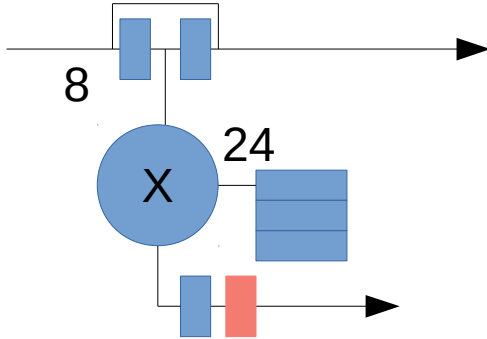
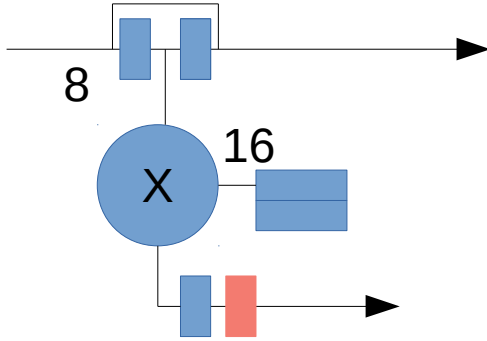
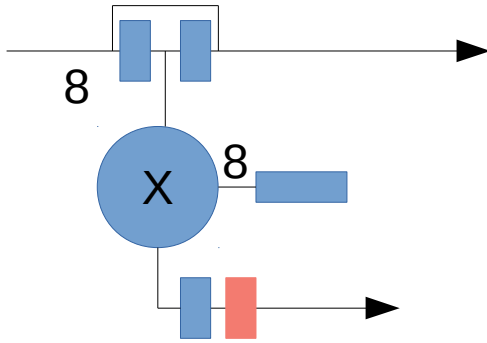
Size	Experiment	Computations (A,B)	A_D	B_D	ACC_D	SHIFTER	Area
DSP48		Dual8x8,27x18					7958
4x3	HP_all+R	8x8,8x16,16x8,24x8,8x24,16x16	6	6	2	HP	18340
4x3	HP_all	8x8,8x16,16x8,24x8,8x24,16x16	6	3	1	HP	13797
4x3	HP_most+R	8x8,8x16,16x8,24x8,16x16	6	4	2	HP	16778
4x3	HP_most	8x8,8x16,16x8,24x8,16x16	6	2	1	HP	13107
4x3	HP_semi+R	8x8,8x16,16x8,16x16	4	4	2	HP	16232
4x3	HP_semi	8x8,8x16,16x8,16x16	4	2	1	HP	12536
4x3	HP_apx+R	8x8,8x16,16x8,16x16apx	4	4	2	HP_apx	14318
4x3	HP_apx	8x8,8x16,16x8,16x16apx	4	2	1	HP_apx	10786
4x3	BYPASS+R	8x8+reuse	2	2	2	BYPASS	10721
4x3	BYPASS	8x8	2	1	1	BYPASS	6445
3x3	HP_all+R	8x8,8x16,16x8,24x8,8x24,16x16	6	6	2	HP	13760
3x3	HP_all	8x8,8x16,16x8,24x8,8x24,16x16	6	3	1	HP	10346
3x3	HP_most+R	8x8,8x16,16x8,24x8,16x16	6	4	2	HP	12641
3x3	HP_most	8x8,8x16,16x8,24x8,16x16	6	2	1	HP	9570
3x3	HP_semi+R	8x8,8x16,16x8,16x16	4	4	2	HP	12206
3x3	HP_semi	8x8,8x16,16x8,16x16	4	2	1	HP	9140
3x3	HP_apx+R	8x8,8x16,16x8,16x16apx	4	4	2	HP_apx	10686
3x3	HP_apx	8x8,8x16,16x8,16x16apx	4	2	1	HP_apx	8161
3x3	BYPASS+R	8x8+reuse	2	2	2	BYPASS	8062
3x3	BYPASS	8x8	2	1	1	BYPASS	4825
2x3	HP_all+R	8x8,8x16,16x8,24x8,8x24,16x16	6	6	2	HP	9246
2x3	HP_all	8x8,8x16,16x8,24x8,8x24,16x16	6	3	1	HP	6944
2x3	HP_most+R	8x8,8x16,16x8,24x8,16x16	6	4	2	HP	8460
2x3	HP_most	8x8,8x16,16x8,24x8,16x16	6	2	1	HP	6447
2x3	HP_semi+R	8x8,8x16,16x8,16x16	4	4	2	HP	8173
2x3	HP_semi	8x8,8x16,16x8,16x16	4	2	1	HP	6160
2x3	HP_apx+R	8x8,8x16,16x8,16x16apx	4	4	2	HP_apx	7186
2x3	HP_apx	8x8,8x16,16x8,16x16apx	4	2	1	HP_apx	5544
2x3	BYPASS+R	8x8+reuse	2	2	2	BYPASS	5440
2x3	BYPASS	8x8	2	1	1	BYPASS	3267

412MHz without pipeline



MLBlobk - PEFlex

For reusement:
Each extra reusement requires one more results register



Why Dot product comparing to Systolic?

- Circuit fusion and optimisation
- Both have same unrolling factor
- Efficient pipelining rather than structured pipelines
- Register replacing and retiming
- Systolic are designed for better scaling (we are focusing on PE design which is small size)
 - If we talk for inside a PE ==> dot-product is better
 - If we explore PE-PE structure ==> Systolic manner
- Vector unit or systolic? Number of IO is much better in Systolic arrays
- My MLBlock benefit both Systolic-array and dot-product based accelerators.
 - Without using Systolic interconnections: MLBlock = a dot-product unit. Great for Supertile
 - Using interconnections: MLBlock = a column based systolic array structure Great for TPU/SeanFPT like

IO requirements

- Types:
 - Stream (Windowing) vs RAM:
 - Reasonable windowing
 - # of IOs
 - Less out
 - Maximum input

For every parameter in a given algorithm

$$P = P_{\text{Comp_Sch}} \times P_{\text{Comp_PE}}$$

$$P_{\text{Comp_Sch}} = \prod P_{\text{Sch-}i}$$

$$P_{\text{Comp_PE}} = P_{\text{Comp_Un}} \times P_{\text{Comp_Seq}}$$

- IO aspect of a PE

$$P_{\text{Comp_Un}} \times P_{\text{Comp_Seq}} = P_{\text{IO_Un}} \times P_{\text{IO_Seq}}$$

$P_{\text{IO_Seq}}$: dictate the clock cycles for fully recharge

$P_{\text{IO_Seq}}$ should be

For every parameter in a given algorithm

Required Multipliers = $\prod P_{\text{comp_PE}}$

$$P_{\text{Comp_Un}} \times P_{\text{Comp_Seq}} = P_{\text{IO_Un}} \times P_{\text{IO_Seq}}$$

- $P_{\text{IO_Seq}}$: clock cycles for fully recharge
- $P_{\text{Comp_Seq}}$:
-

Algorithm 1 CONV layer: simple seven nested loops.

```
for b = 0 until B do
  for k = 0 until K do
    for c = 0 until C do
      for y = 0 until Y do
        for x = 0 until X do
          for fy = 0 until FY do
            for fx = 0 until FX do
              O[b][k][x][y] += I[b][c][x+fx][y+fy]
                           ×W[k][c][fx][fy]
```

Each index affect everyplace if it is used there. example:

“b” is used in “O” and “I”. Thus having n times unrolling “b” requires:

- 1) n times more output result signals
- 2) n times more input signals
- 3) as it is used for “I” in multiplication. It requires n times more Multipliers.

4) There is no effect on “W”

Streaming does not affect anything.

Streaming + Windowing reduces the IOs by saving them inside

Who can be windowed? The elements of input's or weight's indexes which includes more than one elements (each element has the chance to be choosen).

X, Fx, Y, Fy all are the candidates. Selecting one or more is acceptable. Since they are in “I”, they just affect the “I” requirements. The affects of selecting a variable to be windowed is the added shift registers and in adition to unrolled regisres which can be relaised by the algorithm and loop orders.

Extracting from Algorithm

Index	I	W	O	Mult
B	n	1	n	n
K	1	n	n	n
C	n	n	1	n
Y	n	1	n	n
X	n	1	n	n
Fy	n	n	n	1
Fx	n	n	n	1

For every parameter in a given algorithm

Precision

Maybe 9x9 is better than 8x8

- 1) It is Xilinx style
- 2) Fit BRAM well and URAM in a good shape (URAM's width = $72 = 8 \times 9$. So both 8 and 9 are great.
- 3) Then supporting 27×18 is available (keep in mind that Acc width is pain full. 45 bits at least).
 - 1) A size: 3
 - 2) B size: 2 x reuse
 - 3) Acc size: reuse x (at least 45)

Reusement

Reusement factor for different layers (for input act)

1) Standard: KKC

2) DW: C

3) PW: C

4) FC: C

5) Mat-Mat Mult: C

Which PE arch to pick?

- Highest utilisation rate.
- Power will be managed by scheduling (Stanford paper): Power analysis on PE structure is generally non significant
- Scalability
- Flexible precision
- Reusement
- Less partial outputs
- Don't trust on batch parallelism since it is not the case for embedded designs
- Limited number of multipliers

Ideas

- 1) Top-based model → for different algorithm → ASIC PE
- 2) Benchmarking by Ideal PE archs(from 1) of current architectures (How well archs can implement the Ideal PEs)
- 3) How Synthetic arch can be generated?

Intel Architecture

Same as Xilinx
DSP is bigger, BRAM is bigger as well

DSP 28nm:

TABLE II: Area of enhanced DSP blocks and overhead of supporting different modes compared to the baseline.

DSP Block	Post-Synth. Area (μm^2)	Post-P&R Area (μm^2)	Area Ratio
Baseline DSP Block	8404	9875	1.00
Add 9×9 Mult.	8368	10320	1.04
Add 9×9 MAC	8810	10384	1.05
Add 4×4 Mult.(1)-max reuse	9571	—	—
Add 4×4 Mult.(2)-min reuse	9104	10752	1.09
Add 4×4 Mult.(3)-mid reuse	8909	11651	1.18
Add 4×4 MAC using C2	9543	11887	1.20
Add 4×4 MAC using C5	9389	11108	1.12

	intel® Agilex™ F-Series							intel® Agilex™ i-Series	
M20K	1900	2844	3792	5568	7110	11616	13272	11616	13272
DSP18x19	2300	3280	4592	8000	9020	12500	17056	12500	17056
DSP27x27	1150	1640	2296	4000	4510	6250	8528	6250	8528
ratio	1.21	1.15	1.21	1.44	1.27	1.08	1.29	1.08	1.29
DSP/M20K	0.61	0.58	0.61	0.72	0.63	0.54	0.64	0.54	0.64
BRAM18	4033								
DSP27x18	9024								
ratio									
DSP/M20K	2.24								

Roofline model

