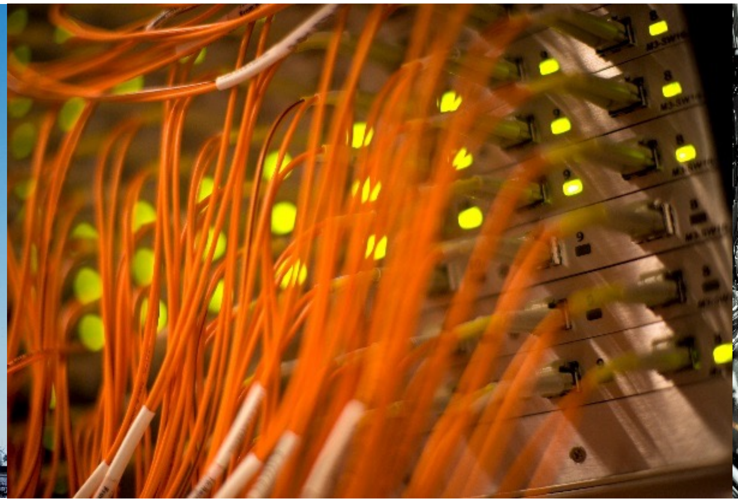
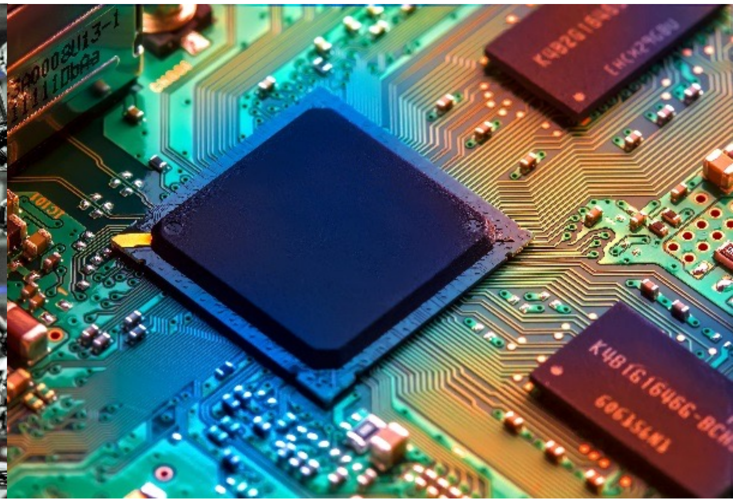
HiKonv: High Throughput Quantized Convolution With Novel Bit-wise Management and Computation

Xinheng Liu, Yao Chen, Prakhar Ganesh, Junhao Pan, Jinjun Xiong, Deming Chen



ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

ADSC
Illinois at Singapore Pte Ltd

University at Buffalo

Bio of the team

Xinheng Liu, Yao Chen, Prakhar Ganesh, Junhao Pan, Jinjun Xiong, Deming Chen

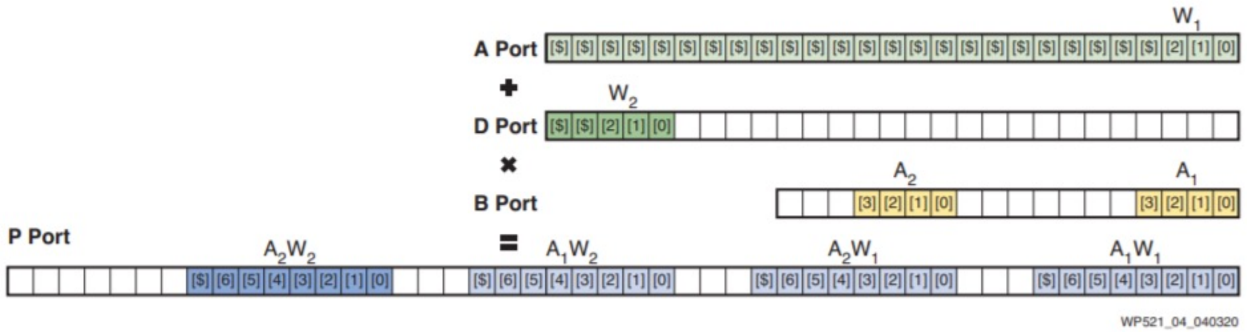
University of Illinois at Urbana-Champaign
Advanced Digital Sciences Center, Singapore
University at Buffalo

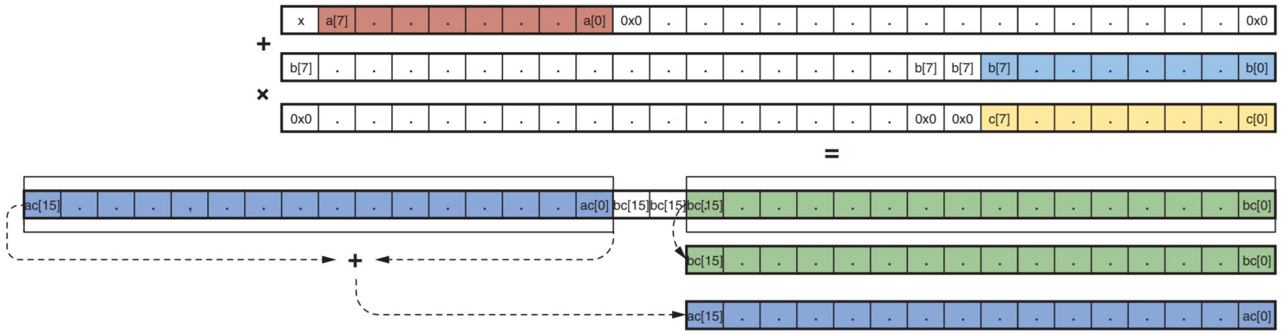
Outline

- Introduction
- Preliminary
  - 1D Convolution
- HiKonv: Multiplication for Convolution
  - Basic idea
  - Detailed bit management
  - DNN extension
- Evaluation

Introduction

- DNN quantization
  - Low-bitwidth data (e.g., 4bit or even less)
- Common hardware computation unit
  - FPGA: DSPs
  - CPU: ALUs
  - Supports large bitwidth arithmetic (16bit & above)
  - Computation wastage for low bitwidth operands
- Previous work for multiple low bitwidth computation
  - FPGA: INT4 Optimization, INT8 Optimization
  - CPU: AVX based solution for 8bit
- Our contributions:
  - **Generalize the solution for all valid quantization bitwidths, ranging from 1 bit to 8 bits**
  - **Provide theoretical foundation for achieving the maximal possible throughput**





Preliminary: 1D-Convolution

- The conventional 1-D discrete convolution between an N-element sequence f and a K-element kernel g (denoted as $y = F_{N,K}(f, g)$)
  - All the values are zero when indices smaller than zero or bigger than the length of the sequences

$$y[m] = (f * g)[m] = \sum_{k=0}^{K-1} f[m-k]g[k]$$

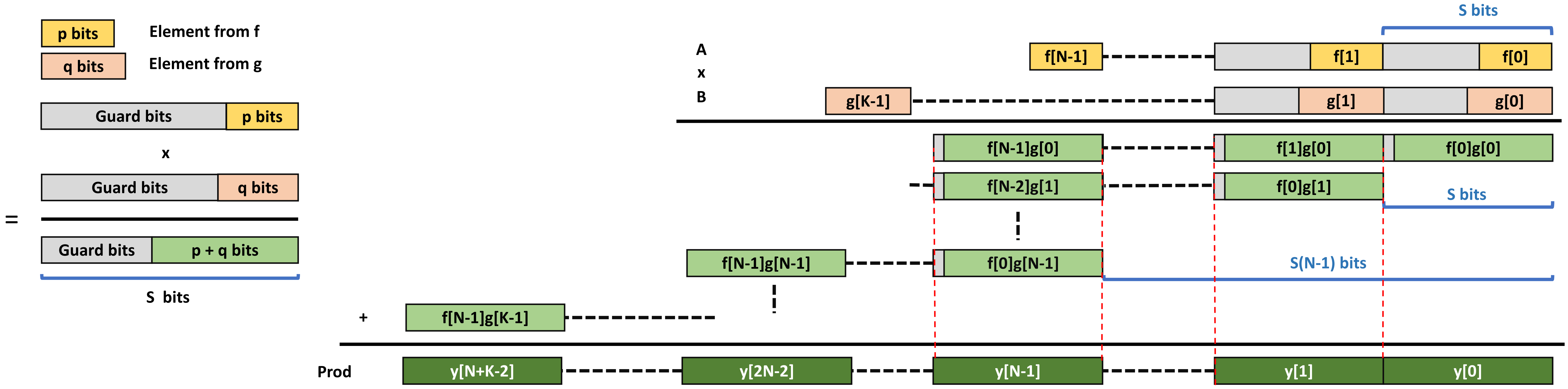
- Alternative representation (replacing $m - k$ with n)

$$y[m] = \sum_{k+n=m} f[n]g[k]$$

- y contains $N + K - 1$ non-zero elements

Multiplier for Convolution: 1-D Convolution

- **Idea**: The product of high bit-width integer multiplication can be used to perform multiple low bit-width 1D convolution operations simultaneously with proper bit management of multiplicands.
  - $P = A \times B$
  - y=[f[0]g[0], f[0]g[1]+f[1]g[0], f[0]g[2]+f[1]g[1]+f[2]g[0] .....]



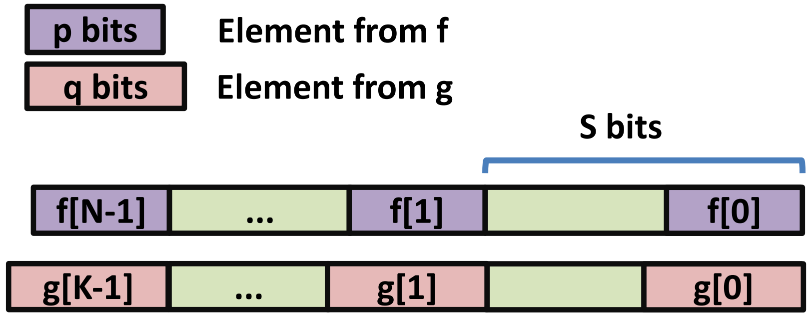
Multiplier for Convolution: low bit-width 1-D Convolution

- Multiplication: $P = A \times B$
- Input multiplicands:
  - Formularization:

$$A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}, B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$$

- Output product:
  - Formularization:

$$P = \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$$



$$P = A \times B = \left(\sum_{n=0}^{N-1} f[n] 2^{Sn}\right) \cdot \left(\sum_{k=0}^{K-1} g[k] 2^{Sk}\right)$$

$$= \sum_{m=0}^{N+K-2} \left(\sum_{n+k=m} \left(f[n] \cdot 2^{Sn} \cdot g[k] \cdot 2^{Sk}\right)\right)$$

$$= \sum_{m=0}^{N+K-2} \left(\sum_{n+k=m} \left(f[n] g[k]\right) \cdot 2^{Sm}\right)$$

$$= \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$$

Multiplier for Convolution: Bitwidth Constraints

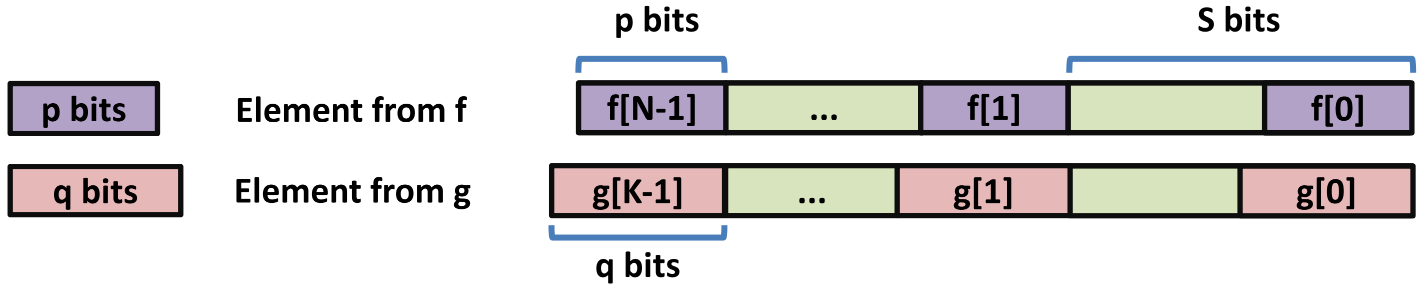
- Choice of S
  - S-bit segment should be large enough to contain each y element
  - Guard bit G_b prevents overflow from accumulation
  - $G_b = \lceil \log_2 \min(K, N) \rceil$
- Bit width constraints:
  - The packed bit width cannot exceed the multiplicands bitwidth
  - Bit_A and Bit_B : bitwidth of multiplicand A and B

$$S = \begin{cases} q + G_b, & p = 1, q \geq 1 \\ p + G_b, & q = 1, p \geq 1 \\ p + q + G_b, & otherwise \end{cases}$$

$$y[m] = \sum_{k+n=m} f[n]g[k]$$

$$\begin{cases} p + (N-1)S \leq Bit_A \\ q + (K-1)S \leq Bit_B \end{cases}$$

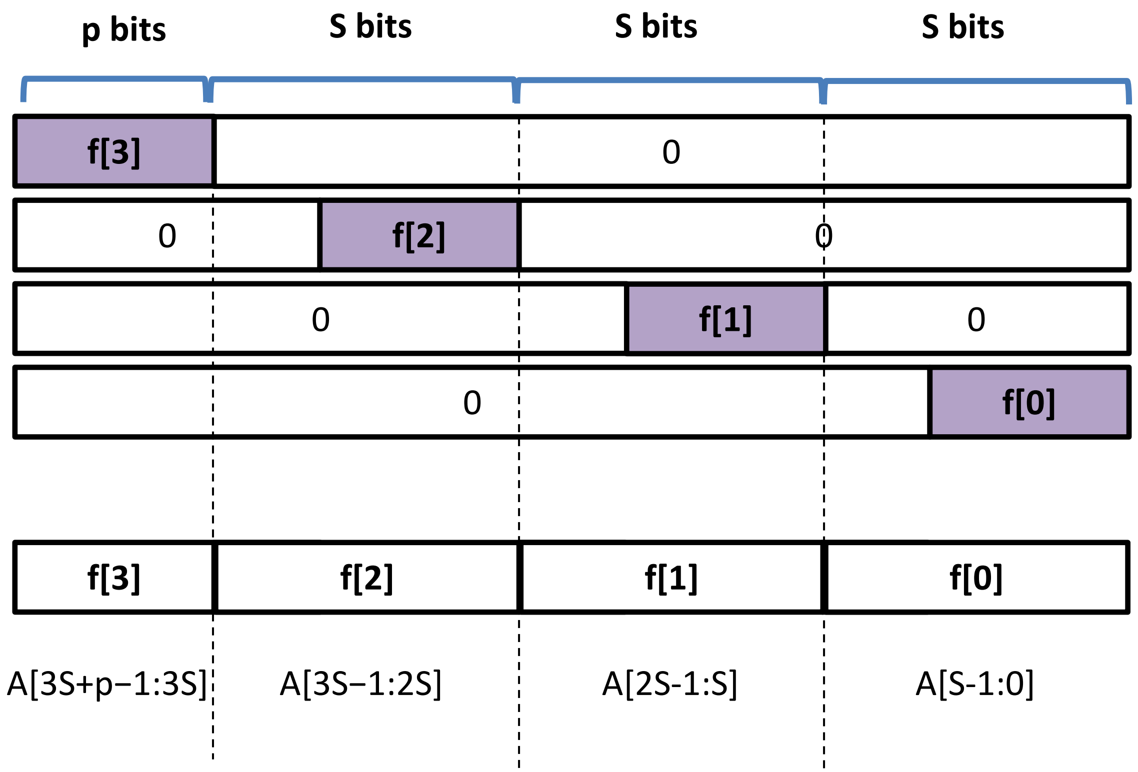
$$A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}, B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$$



Multiplier for Convolution: Bit Management

- Multiplication for convolution
  - Input Packing: $A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}, B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$
  - Output Slicing: $P = \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$
- Efficient packing and slicing
  - Unsigned f and g:
    - $A[S(n+1)-1:Sn] = f[n]$
    - $B[S(k+1)-1:Sk] = g[k]$
    - $P[S(m+1)-1:Sm] = y[m]$

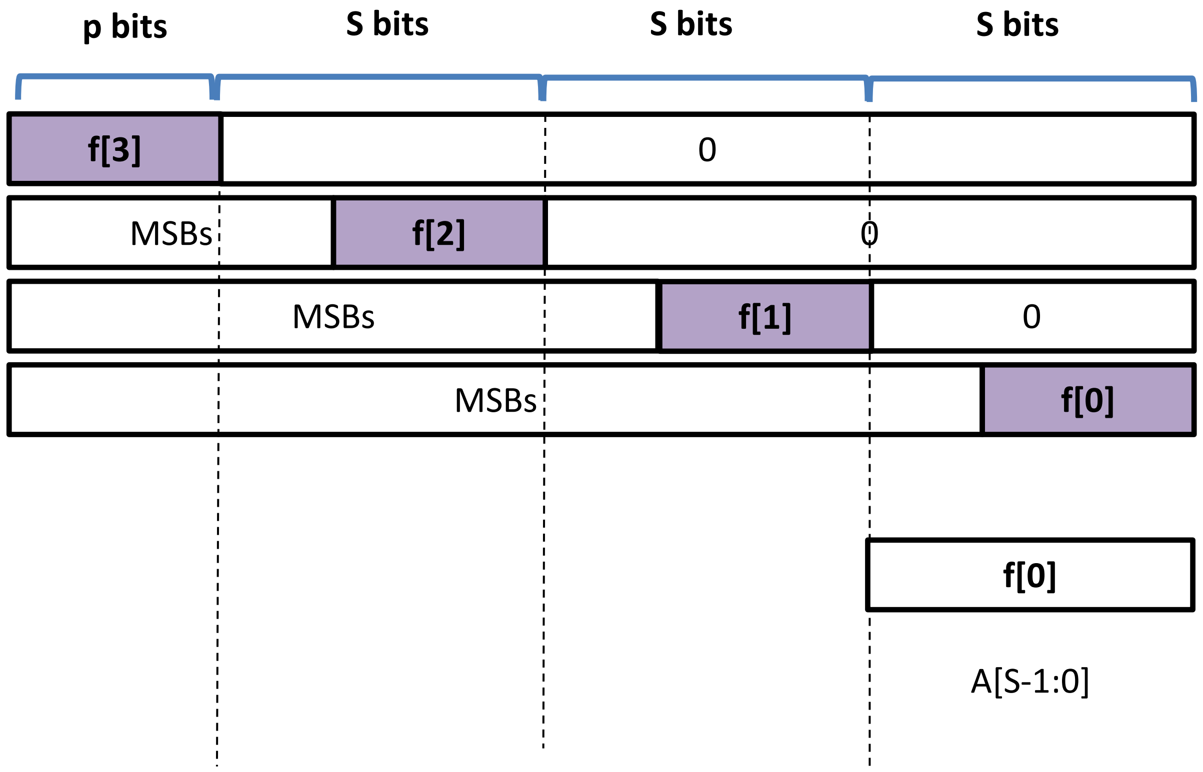
$$A = f[3] \cdot 2^{3S} + f[2] \cdot 2^{2S} + f[1] \cdot 2^{S} + f[0]$$



Multiplier for Convolution: Bit Management

- Multiplication for convolution
  - Input Packing: $A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}, B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$
  - Output Slicing: $P = \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$
- Efficient packing and slicing-
  - Unsigned f and g:
    - $A[S(n+1)-1:Sn] = f[n]$
    - $B[S(k+1)-1:Sk] = g[k]$
    - $y[m] = P[S(m+1)-1:Sm]$
  - Signed f and g:

$$A = f[3] \cdot 2^{3S} + f[2] \cdot 2^{2S} + f[1] \cdot 2^{S} + f[0]$$

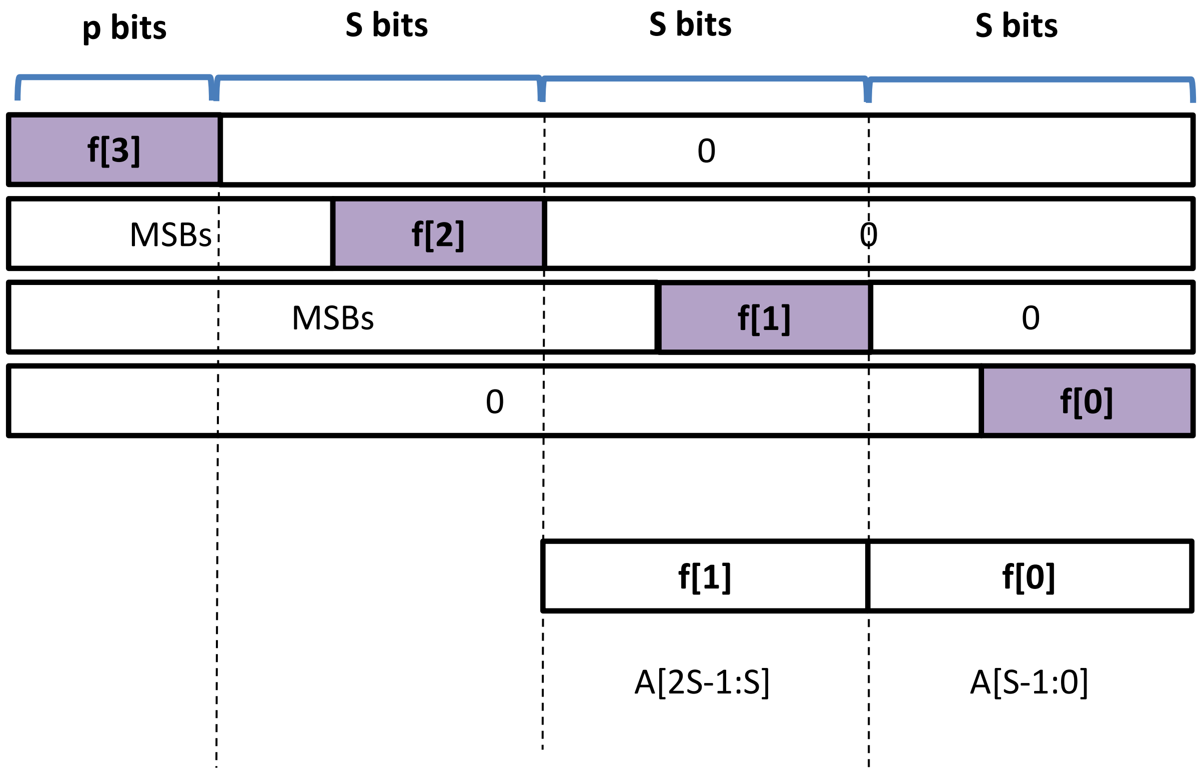


A[S-1:0]=f[0]

Multiplier for Convolution: Bit Management

- Multiplication for convolution
  - Input Packing: $A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}$, $B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$
  - Output Slicing: $P = \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$
- Efficient packing and slicing-
  - Unsigned f and g:
    - $A[S(n+1)-1{:}Sn] = f[n]$
    - $B[S(k+1)-1{:}Sk] = g[k]$
    - $y[m] = P[S(m+1)-1{:}Sm]$
  - Signed f and g:

$$A = f[3] \cdot 2^{3S} + f[2] \cdot 2^{2S} + f[1] \cdot 2^{S} + f[0]$$

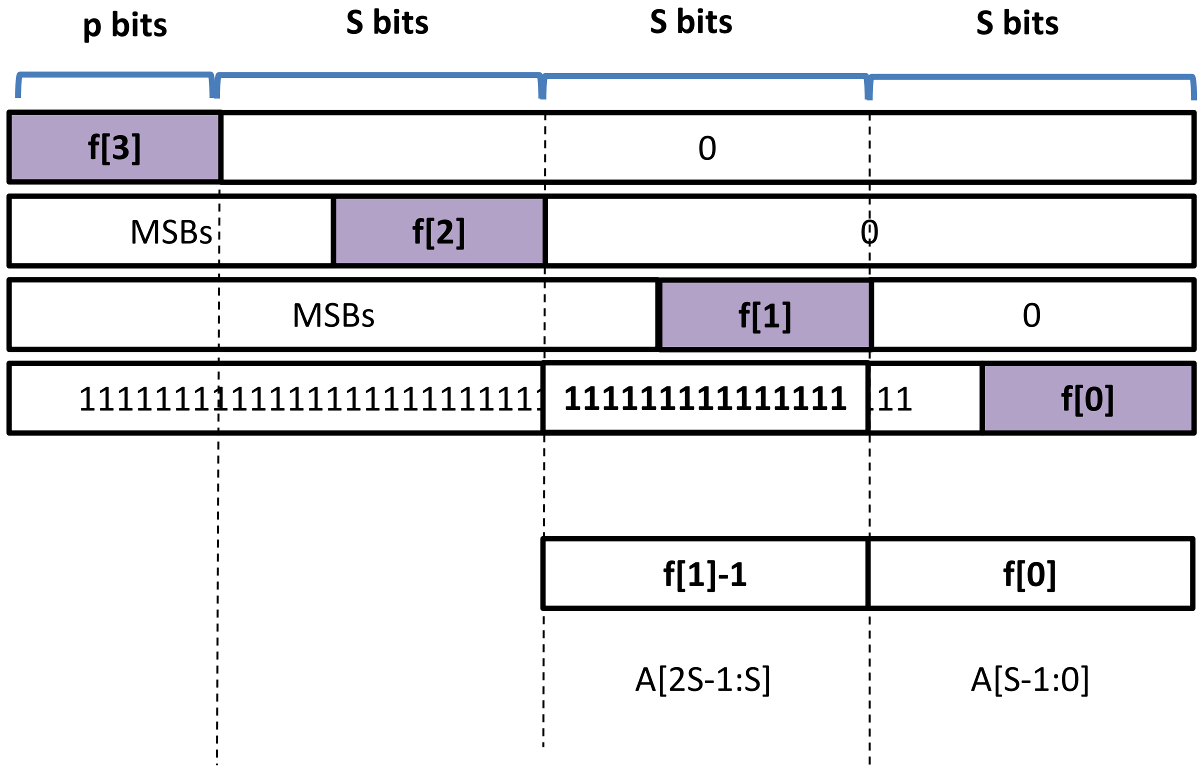


f[0]>=0 A[2S-1:S]=f[1]

Multiplier for Convolution: Bit Management

- Multiplication for convolution
  - Input Packing: $A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}, B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$
  - Output Slicing: $P = \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$
- Efficient packing and slicing-
  - Unsigned f and g:
    - $A[S(n+1)-1{:}Sn] = f[n]$
    - $B[S(k+1)-1{:}Sk] = g[k]$
    - $y[m] = P[S(m+1)-1{:}Sm]$
  - Signed f and g:

$$A = f[3] \cdot 2^{3S} + f[2] \cdot 2^{2S} + f[1] \cdot 2^{S} + f[0]$$



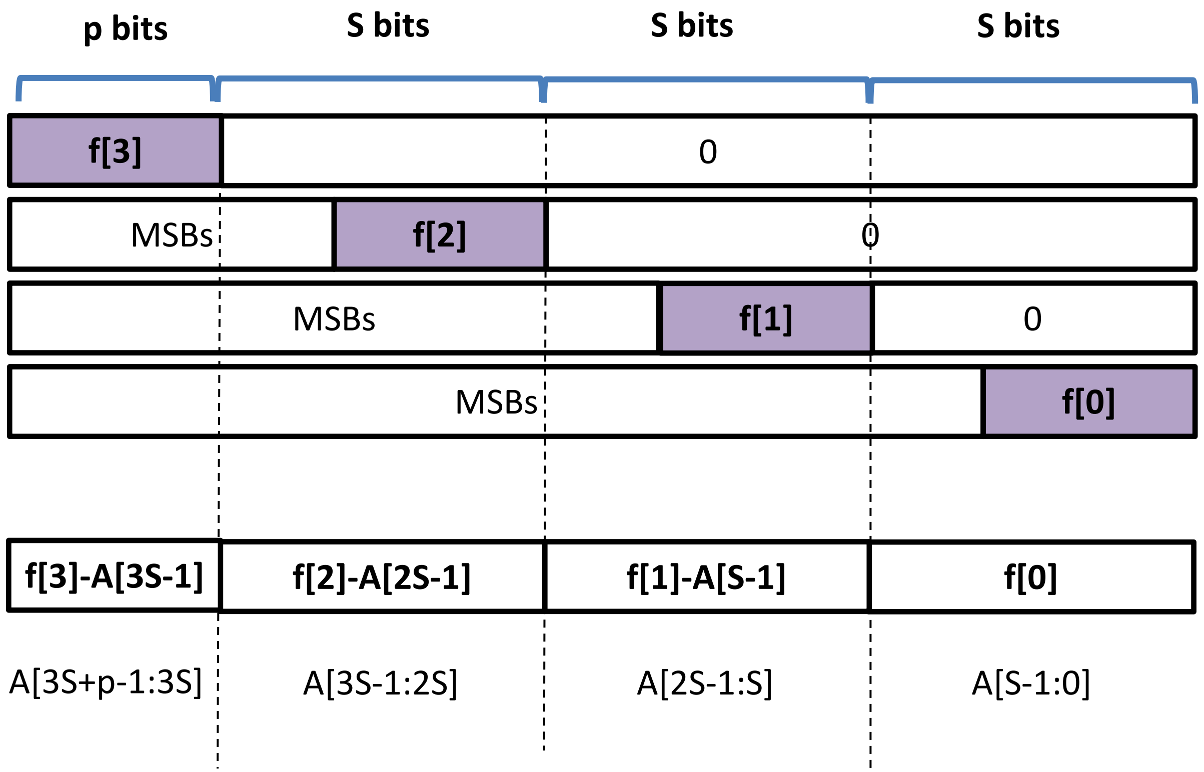
f[0]>=0 A[2S-1:S]=f[1]

f[0]< 0 $A[2S-1:S]=f[1]_2+1111...111_2=f[1]-1$

Multiplier for Convolution: Bit Management

- Multiplication for convolution
  - Input Packing: $A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}, B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$
  - Output Slicing: $P = \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$
- Efficient packing and slicing-
  - Unsigned f and g:
    - $A[S(n+1)-1:Sn] = f[n]$
    - $B[S(k+1)-1:Sk] = g[k]$
    - $y[m] = P[S(m+1)-1:Sm]$
  - Signed f and g:
    - $A[S(n+1)-1:Sn] = \begin{cases} f[0], n = 0 \\ f[n] - A[Sn-1], n > 0 \end{cases}$
    - $B[S(k+1)-1:S] = \begin{cases} g[0], k = 0 \\ g[k] - B[Sk-1], k > 0 \end{cases}$

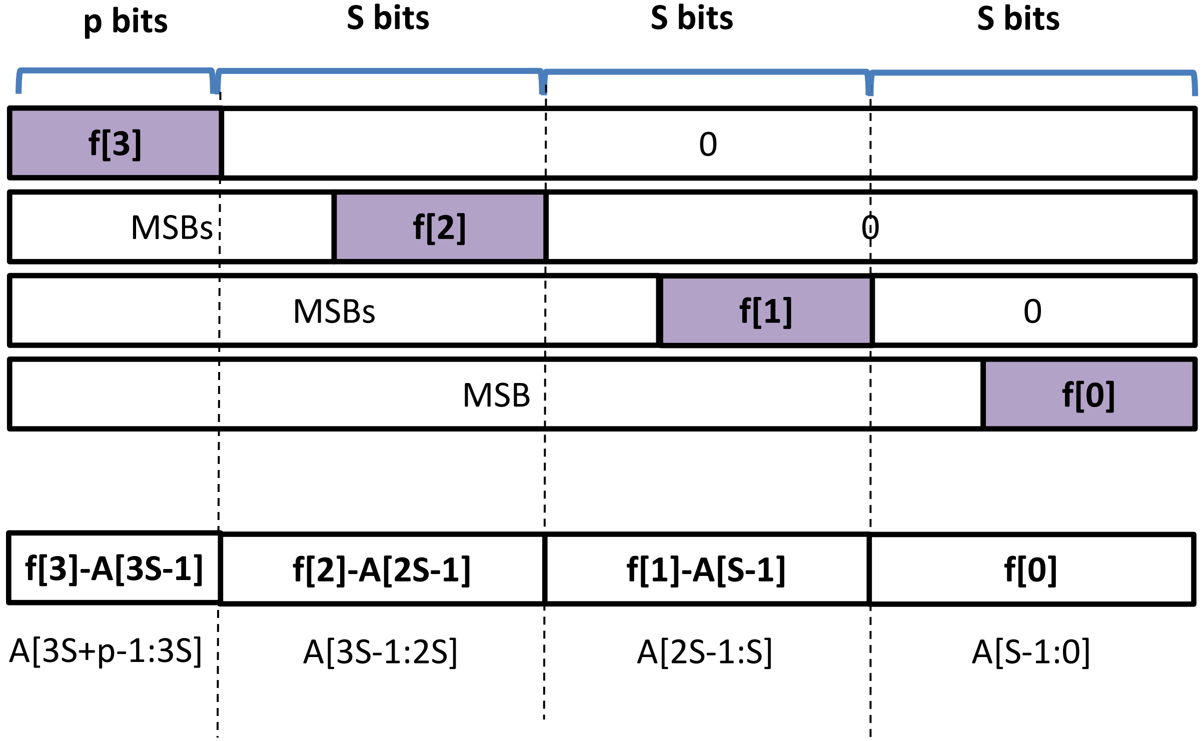
$$A = f[3] \cdot 2^{3S} + f[2] \cdot 2^{2S} + f[1] \cdot 2^{S} + f[0]$$



Multiplier for Convolution: Bit Management

- Multiplication for convolution
  - Input Packing: $A = \sum_{n=0}^{N-1} f[n] \cdot 2^{Sn}$, $B = \sum_{k=0}^{K-1} g[k] \cdot 2^{Sk}$
  - Output Slicing: $P = \sum_{m=0}^{N+K-2} y[m] \cdot 2^{Sm}$
- Efficient packing and slicing-
  - Unsigned f and g:
    - $A[S(n+1)-1:Sn] = f[n]$
    - $B[S(k+1)-1:Sk] = g[k]$
    - $y[m] = P[S(m+1)-1:Sm]$
  - Signed f and g:
    - $A[S(n+1)-1:Sn] = \begin{cases} f[0], n = 0 \\ f[n] - A[Sn-1], n > 0 \end{cases}$
    - $B[S(k+1)-1:Sk] = \begin{cases} g[0], k = 0 \\ g[k] - B[Sk-1], k > 0 \end{cases}$
    - $y[m] = \begin{cases} P[S-1:0], m = 0 \\ P[S(m+1)-1:Sm]+P[Sm-1], m > 0 \end{cases}$

$$A = f[3] \cdot 2^{3S} + f[2] \cdot 2^{2S} + f[1] \cdot 2^{S} + f[0]$$



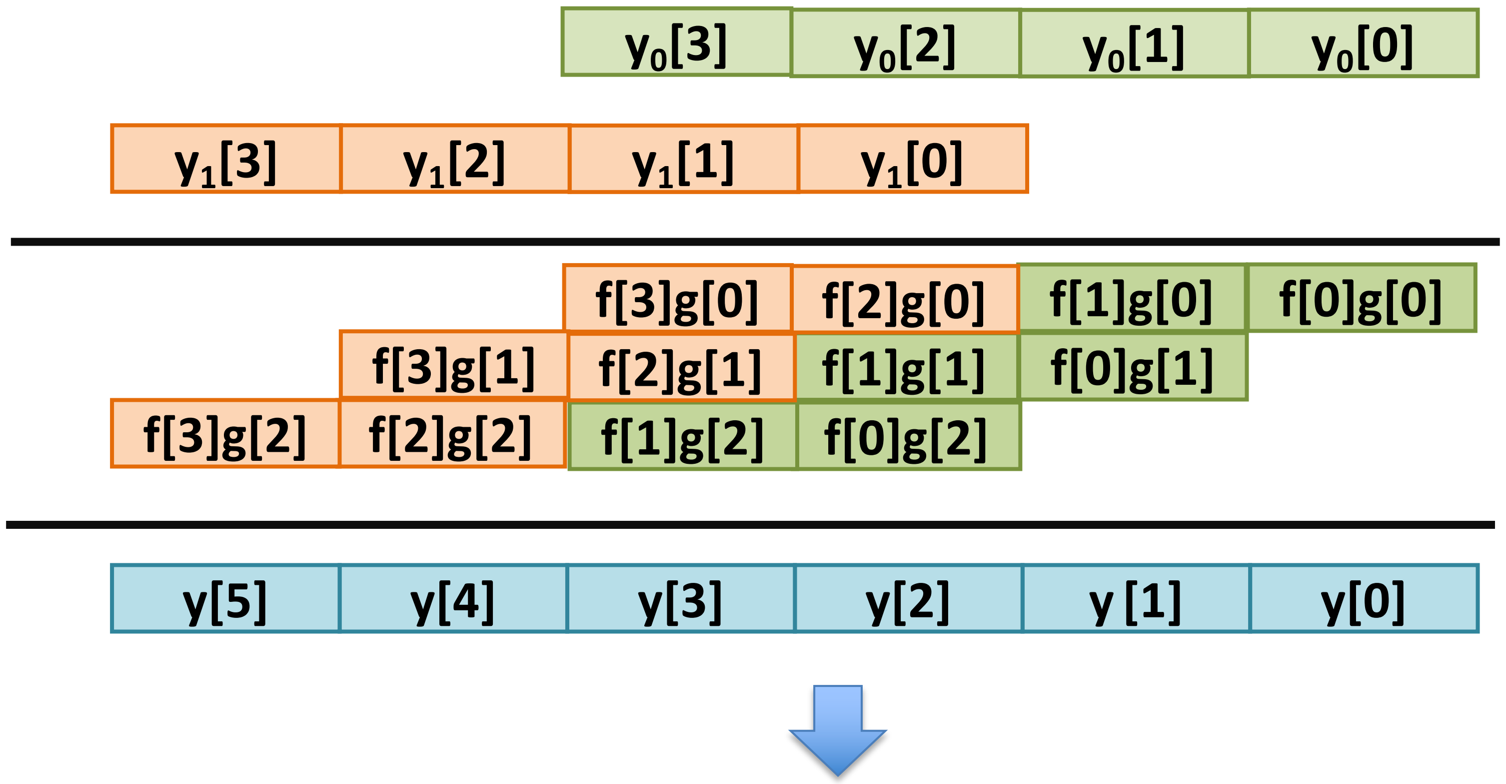
$$P[S(m+1)-1:Sm] = \begin{cases} y[0], m = 0 \\ y[m] - P[Sm-1], m > 0 \end{cases}$$

$$y[m] = \begin{cases} P[S-1:0], m = 0 \\ P[S(m+1)-1:Sm]+P[Sm-1], m > 0 \end{cases}$$

Note: the most significant slice has a different format

1D Convolution Extension: Split and Accumulation

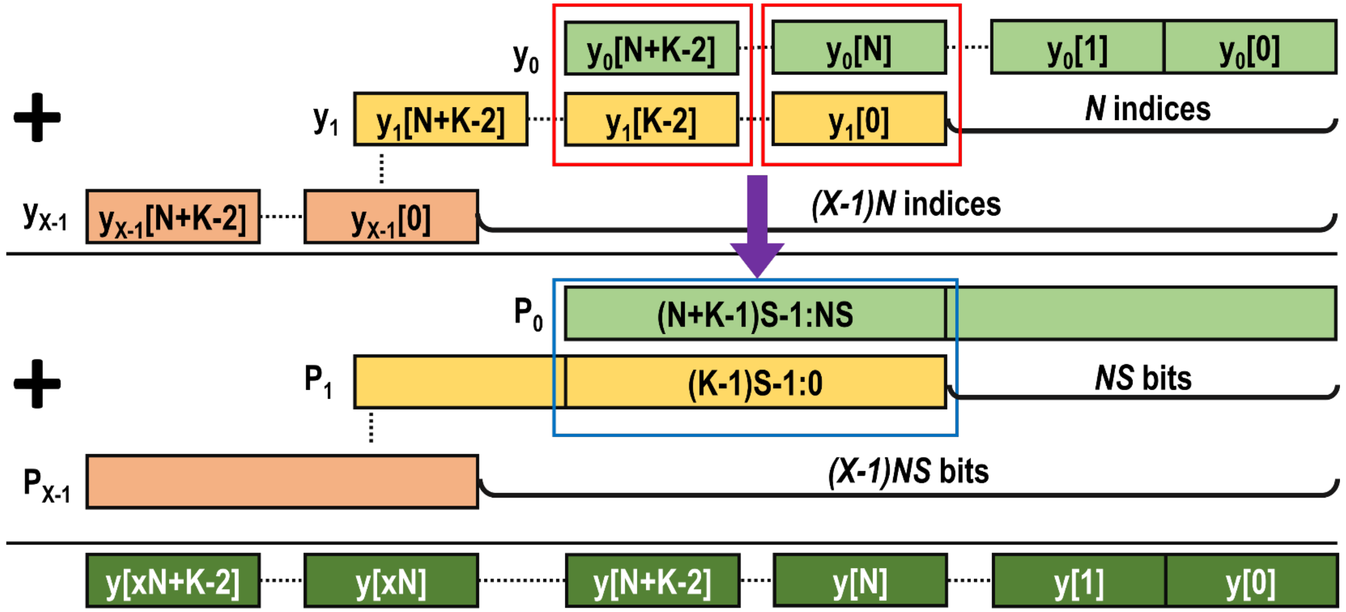
- **Idea:**
  - Partition the original sequence into multiple subsequences
  - Compute 1D convolution for each subsequence
  - Accumulate the subsequence results to produce the final convolution solution
- Example:
  - 4-element sequence f and 3-element sequence g
  - $f \rightarrow f_{0,1} | f_{2,3}$
  - $y_0 = F_{2,3}(f_{0,1}, g), y_1 = F_{2,3}(f_{2,3}, g)$
  - $y = F_{4,3}(f, g)$ can be composed based on the elements in y_0 and y_1



$y = F_{N,K}(f, g)$

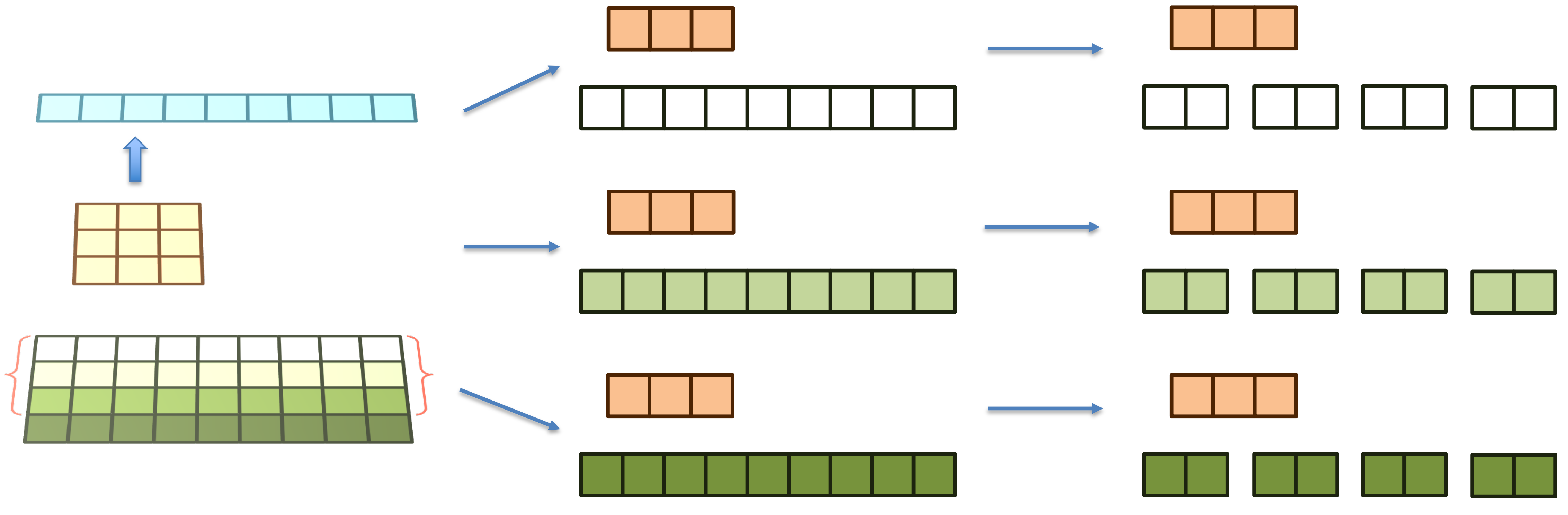
1D Convolution Extension: Theorem to Generalize the Technique

- **Theorem**: Given and an XN-element sequence f and a K-element filter g, the 1D convolution output $y = F_{XN,K}(f, g)$ can be computed by following computation step:
  - Sequence split: $f_x = f[xN: (x+1)N - 1]$.
  - 1D convolution: $y_x = F_{N,K}(f_x, g)$
  - $y_x \rightarrow y_x[n - xN]$
  - $y[n] = \sum_{x=0}^{X-1} y_x[n - xN]$



2D DNN Convolution Extension

- DNN convolution layers have convolution pattern and can be built upon our 1D convolution techniques



2D DNN Convolution Extension

- DNN convolution formula:

$$O[c_o][h][w] = \sum_{c_i=0}^{C_i-1} \sum_{k_h=0}^{K-1} \sum_{k_w=0}^{K-1} I[c_i][h+k_h][w+k_w] W[c_o][c_i][k_h][k_w]$$

- **Theorem**: For a DNN convolution, the output feature-map can be computed by $F_{N,K}$ 1-D convolution with the following equation:

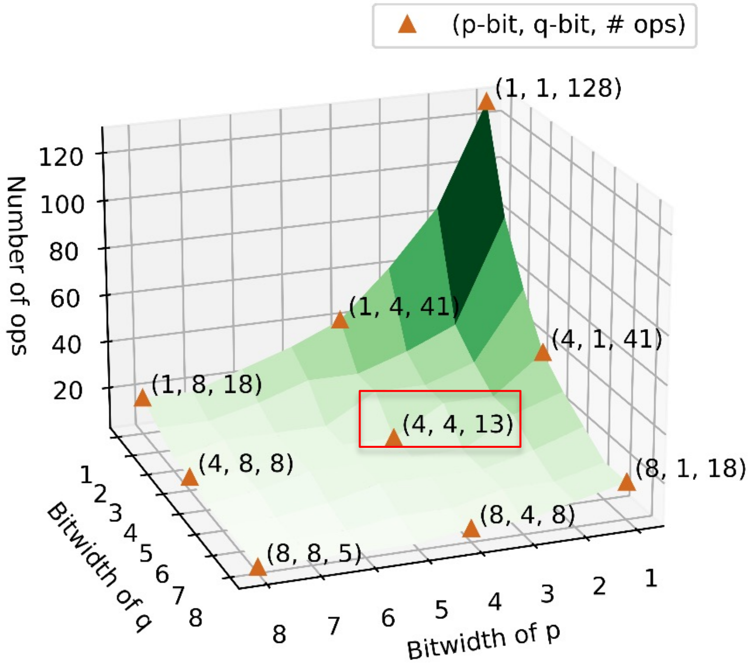
$$O[c_o][h][w] = \sum_{c_i=0}^{C_i-1} \sum_{k_h=0}^{K-1} \sum_{x=0}^{\lceil \frac{W_i}{N} \rceil - 1} y_{c_i,c_o,h,k_h,x}[w - xN + K - 1]$$

Where

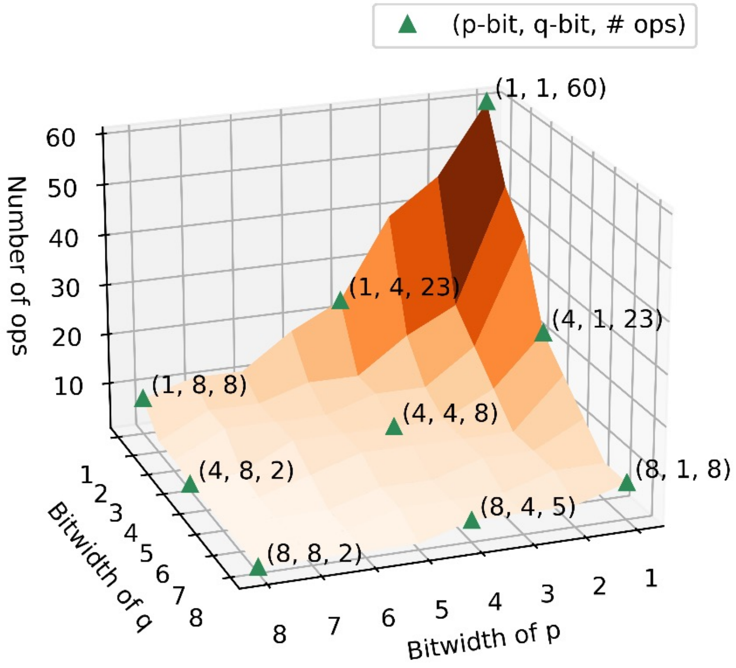
$$\begin{cases} y_{c_i,c_o,h,k_h,x} = F_{N,K}(f, g) \\ f = I[c_i][h+k_h][xN:(x+1)N-1] \\ g = W[c_o][c_i][k_h][K-1:0] \end{cases}$$

Evaluation: Single Multiplication Unit Throughput

- Evaluation computation unit
  - CPU : 32 bit multiplier
  - FPGA: 27x18 bit multiplier
- Maximum N,K with bitwidth constraint
  - $p + (N-1)S \leq Bit_A$
  - $q + (K-1)S \leq Bit_B$
- Evaluation throughput
  - Maximum number of effective operations (add or multiplication) in convolution within each multiplication



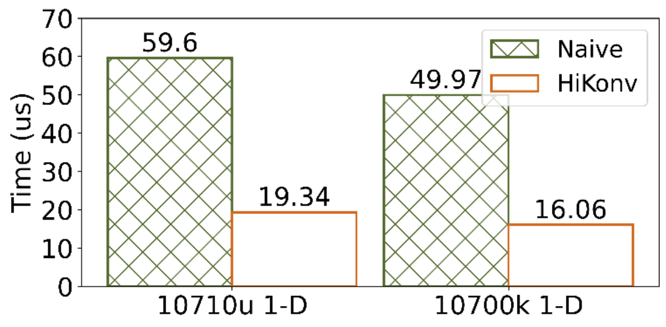
CPU: A = 32 bits, B = 32 bits



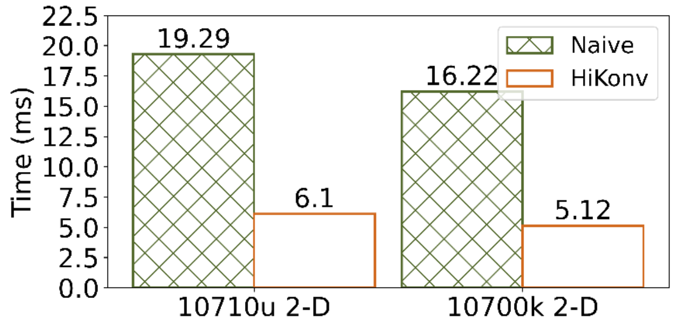
FPGA: A = 27 bits, B = 18 bits

Evaluation: General Purpose Processors

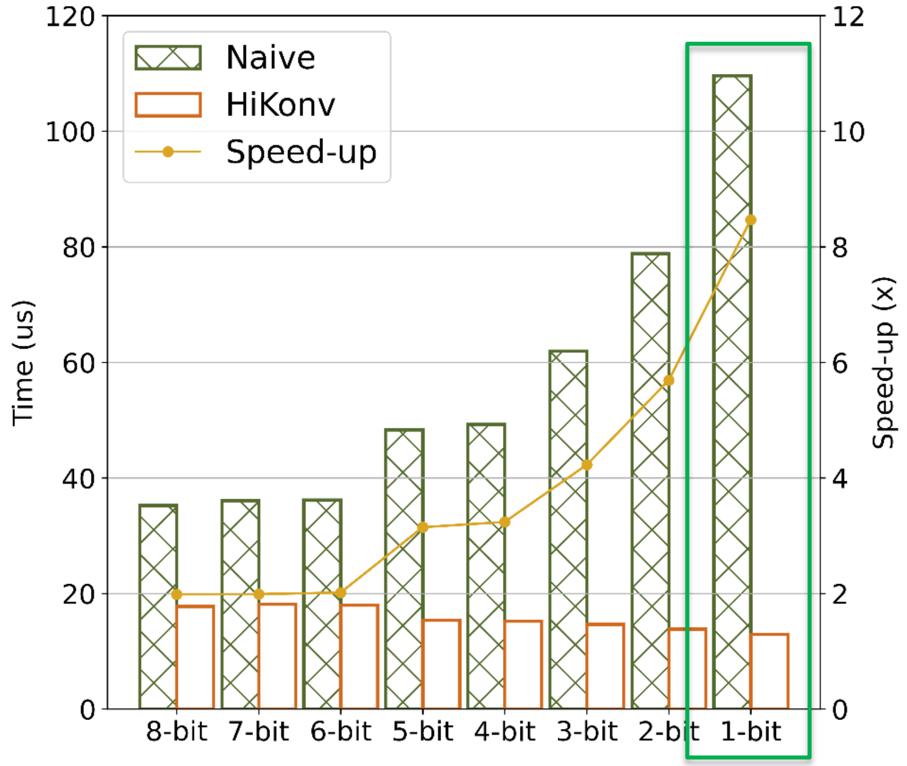
- Test platform
  - Intel Core i7-10700K CPU and i7-10710U CPU
- Test case
  - 1D and 2D convolution
    - 32bit multiplier, unsigned 4-bit data
    - K=3,N=3, S=10
  - 1D convolution with different bitwidth
- ~3x faster than the baseline algorithm



1-D Convolution



2-D Convolution



Speedup for different bitwidths.

ECE ILLINOIS

Evaluation: Reconfigurable Computation Device

- Platform:
  - Xilinx Ultra96 MPSoC platform
- BNN testcase:
  - 1bit weight and 1bit feature map
  - Same performance
  - LUTs to DSP ratio: 43.7~76.6

Table I: Comparison of Resource util. of binary convolution

# of Concurrent MACs		336	576	960	1536	3072
BNN-LUT	LUT	3371	4987	7764	12078	23607
BNN-HiKonv	LUT	2672	2536	3369	3587	9319
	DSP	16	32	64	128	256
	DSP Thro.	21	18	15	12	12
	LUT/DSP	43.7	76.6	68.7	65.4	55.8

Evaluation: Reconfigurable Computation Device

- Low Bitwidth DNN testcase
  - 4bit CNN model
  - DACSDC 2020 Winner **Ultranet**
  - ~ 2.37X better performance
  - ~2.61X DSP efficiency

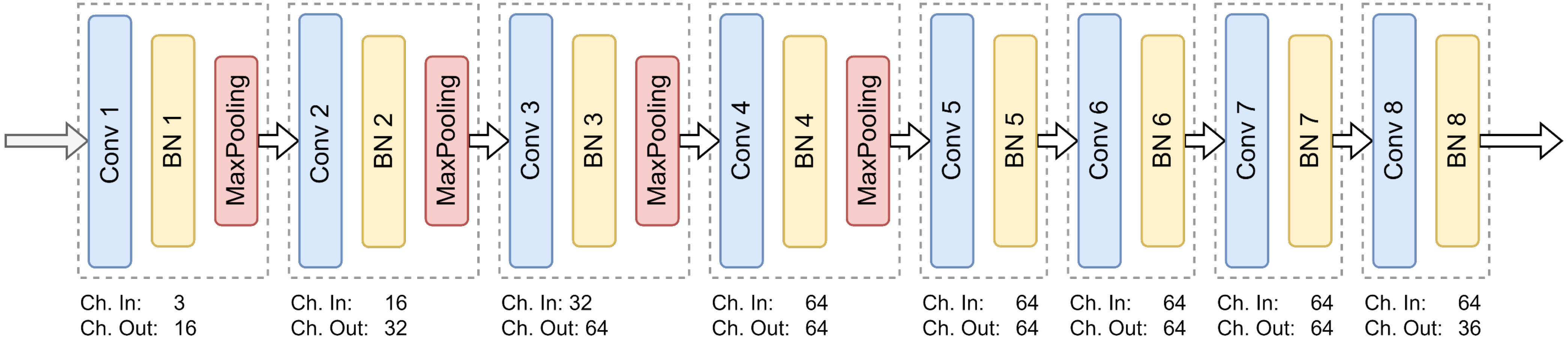


Table II: UltraNet resource and performance.

	LUT	DSP	fps	DSP Eff. (Gops/DSP)
UltraNet	4.3k	360	248	0.289
UltraNet-HiKonv	4.8k	327	401/588	0.514/0.753
			2.37X ↑	2.61X ↑

Conclusion

- Proposed a general technique, Hikonv, with theoretical guarantees for using a single multiplier unit to process multiple low-bitwidth convolution operations in parallel for significantly higher computation throughput with flexible bitwidths.
- HiKonv supports both the 1D convolution and DNN convolutions
- Achieved 3.17x throughput improvement on CPU solutions and 2.37x performance improvements on FPGA solutions.

Thank You!
Q & A