DSPlacer: DSP Placement for FPGA-based CNN Accelerator

Baohui Xie[1†], Xinrui Zhu[1†], Zhiyuan Lu[1], Yuan Pu[2], Tongkai Wu[1], Xiaofeng Zou[3], Bei Yu[2], Tinghuan Chen[1*]

[1]CUHK-Shenzhen [2]CUHK [3]National University of Defense Technology

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA sigda

Outline

1. Introduction
2. Algorithm
3. Experiment

Introduction

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda ACM

Introduction to DSPlacer

DSPlacer, a datapath-driven FPGA placement method, optimizes DSP placement to boost CNN accelerator clock frequency.

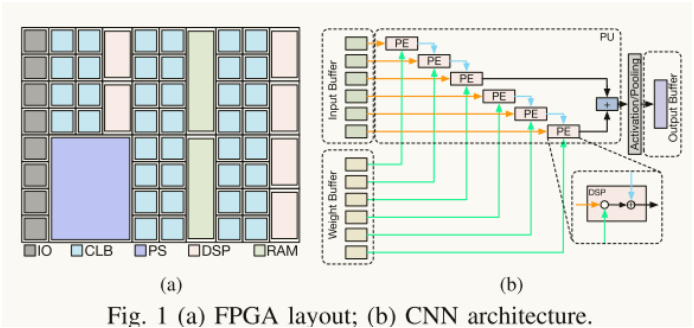


Fig. 1 (a) FPGA layout; (b) CNN architecture.

FPGA layout and CNN architecture

Problem Formulation

Objective: To maximize the clock frequency of a CNN accelerator on an FPGA by optimizing the placement of DSPs.

Input:

1. A pre-implementation netlist defines component connections.
2. DSP specifications of the target FPGA

Constraints:

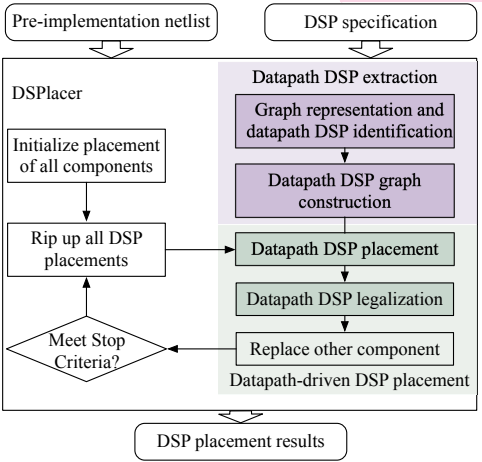
1. DSPs must be placed in legal locations.

Challenge:

Ensuring clock frequency is maximized while maintaining timing closure and minimizing the complexity of the placement algorithm.

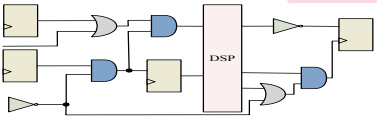
Algorithm

Algorithm: DSPlacer overview

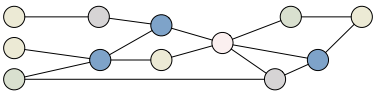


DSPlacer overview

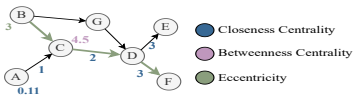
Algorithm: Datapath DSP extraction



(a)



(b)



(c)

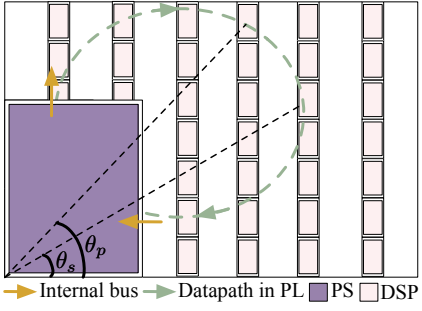
(a) Netlist; (b) Graph representation; (c) Node embedding layer.

Algorithm: Datapath-Driven DSP Placement

Key Steps:

1. Datapath DSP Placement: Assign DSP components to device locations.
2. Datapath DSP Legalization: Ensure compliance with cascade constraints.

- Optimize timing performance by mapping DSP components to specific locations.
- Preserve datapath structure while satisfying hardware constraints.



DSP datapath.

Algorithm: DSP Placement Formulation

$$\min_{x_{i,j}} \sum_{e\in\mathcal{E}} (\boldsymbol{x}_{e_p} - \boldsymbol{x}_{e_s})^\top (\boldsymbol{p}_x \boldsymbol{p}_x^\top + \boldsymbol{p}_y \boldsymbol{p}_y^\top)(\boldsymbol{x}_{e_p} - \boldsymbol{x}_{e_s}) \quad + \lambda \sum_{e_D\in\mathcal{E}_D} (\cos\theta_{e_{Dp}} - \cos\theta_{e_{Ds}}) \tag{1}$$

$$+ \eta \sum_{c\in C} \sum_{j=1}^{M-1} (x_{c_p,j} - x_{c_s,j+1})^2 \tag{2}$$

$$\text{s.t.} \quad \sum_{j=1}^{M} x_{i,j} = 1, \quad \forall i = 1, \cdots, N \tag{3}$$

$$\sum_{i=1}^{N} x_{i,j} \le 1, \quad \forall j = 1, \cdots, M \tag{4}$$

$$x_{c_p,j} = x_{c_s,j+1}, \quad \forall c \in C, \forall j = 1, \cdots, M-1 \tag{5}$$

$$\cos\theta_{e_{Dp}} \le \cos\theta_{e_{Ds}} \tag{6}$$

$$x_{i,j} \in \{0,1\}, \quad \forall i = 1, \cdots, N, \forall j = 1, \cdots, M \tag{7}$$

Algorithm: Formulation Linearization

We rewrite the above formulation as follows.

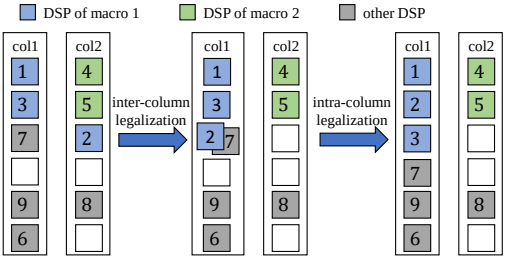
$$\begin{aligned} &\min_{x_{i,j}} \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{p=1}^{N}\sum_{q=1}^{M} a_{i,j,p,q}x_{i,j}x_{p,q} + \lambda \sum_{i=1}^{N}\sum_{j=1}^{M} c_j x_{i,j}, \\ &\text{s.t.} \sum_{j=1}^{M} x_{i,j} = 1, \quad \sum_{i=1}^{N} x_{i,j} \leq 1, \quad x_{c_p,j} = x_{c_s,j+1}, \quad \forall c \in C, x_{i,j} \in \{0,1\}. \end{aligned} \tag{8}$$

Linearization for ILP Solver Using iterative linearization:

$$a_{i,j,p,q}x_{i,j}x_{p,q} \approx \frac{1}{2}a_{i,j,p,q}(x'_{i,j}x_{p,q} + x_{i,j}x'_{p,q})$$

Algorithm: Inter - column Legalization

1. Inter-column Legalization: Minimize horizontal displacement.



Legalization

$$\min_{t_{i,j}} \sum_{i=1}^{N} \sum_{j=1}^{N_{col}} D_{col}(i,j) t_{i,j} \tag{9}$$

$$\text{s.t.} \quad \sum_{j=1}^{N_{col}} t_{i,j} = 1, \quad \sum_{i=1}^{N} t_{i,j} \le M_j \tag{10}$$

$$t_{c_p,j} = t_{c_s,j}, \quad \forall c \in C \tag{11}$$

$$t_{i,j} \in \{0,1\} \tag{12}$$

Algorithm: Intra-column Legalization

2. Intra-column Legalization: Minimize vertical displacement.

$$\min_{r_i} \sum_{i=1}^{N_j} |r_i - R_{col}(i)|, \tag{13}$$

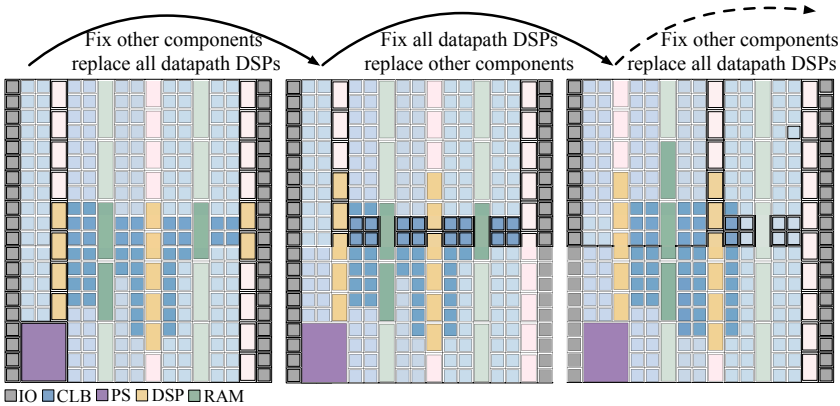
$$\text{s.t.} \quad x_i \in \{1, 2, ..., M_j\},$$

$$r_{i+1} - r_i = 1, \forall (i, i+1) \in C_j, \tag{14}$$

$$r_{i+1} - r_i \geq 1, \forall (i, i+1) \notin C_j, \tag{15}$$

Algorithm: Incremental Placement Strategy

Incremental Placement - Alternate between placing datapath DSPs and other components. - Fix positions of one group while optimizing for the other.



Incremental Datapath DSP Placement.

Experiment

Experiment setup

Development Environment

- Implemented in Python using the NetworkX library for graph transformation, and also uses Cytoscape for post-processing the DSP graph if needed.
- LEMON for solving the min-cost flow.
- Gurobi for solving the ILP.
- Utilizes Xilinx Vivado 2020.1 as the FPGA commercial tool for placement, routing, and final timing analysis.
- Our target FPGA is Xilinx Zynq UltraScale+ MPSoC ZCU104.
- We also modified AMF2.0 so that it can run in ZCU104 instead of VCU108.

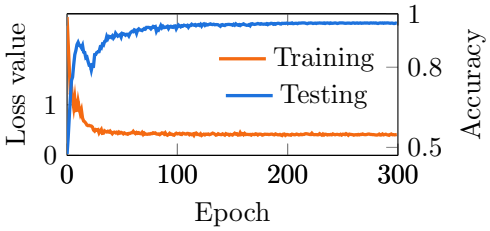
Benchmarks and Datapath DSP Extraction Comparison

Table: Benchmarks detail.

Design	#LUT	#LUTRAM	#FF	#BRAM	#DSP	DSP%	freq.(MHz)
iSmartDNN	53503	2919	55767	122	197	11%	130.0
SkyNet	43146	2748	51410	192	346	20%	150.0
SkrSkr-1	35743	3611	53887	196	642	37%	195.0
SkrSkr-2	70558	3815	64007	196	1180	68%	175.0
SkrSkr-3	70382	3791	67257	196	1431	83%	175.0

	SVM **Ward2012**	GCN
iSmartDNN	80%	88.1%
SkyNet	85%	90%
SkrSkr-1	96%	96%
SkrSkr-2	69%	95.5%
SkrSkr-3	78.7%	96.6%

(a)



(b)

Timing Result

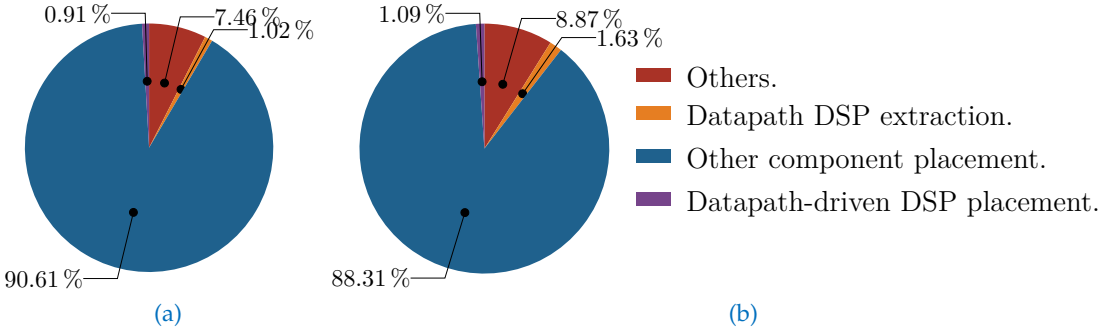
Benchmark, baseline and result

Our baseline is Vivado 2020.2 and AMF-Placer 2.0.

Benchmark	Vivado				AMF				DSPlacer			
	WNS (ns)	TNS (ns)	HPWL (um)	Runtime (s)	WNS (ns)	TNS (ns)	HPWL (um)	Runtime (s)	WNS (ns)	TNS	HPWL (um)	Runtime (s)
iSmartDNN	-0.131	-0.391	2965232	771	-0.830	-649.995	5152227	1552	**0.151**	0	3107487	865
SkyNet	-0.164	-7.765	3482155	868	-0.154	-3.557	3633692	975	**0.189**	0	3878462	940
SkrSkr-1	-0.587	**-97.799**	2488788	775	-0.883	-4043.358	3384970	699	**-0.164**	-109.042	4473110	701
SkrSkr-2	-0.597	-826.739	3700116	1091	-1.865	-16672.268	8809141	4486	**0.009**	0	4489697	2462
SkrSkr-3	-0.216	-21.417	4034489	1232	-0.786	-804.950	6104589	3329	**0.007**	0	4912782	1755
Normalize	1.325×	1.042×	0.550×	0.485×	1.658×	1.103×	1.446×	2.145×	**1.000**×	**1.000**×	**1.000**×	**1.000**×

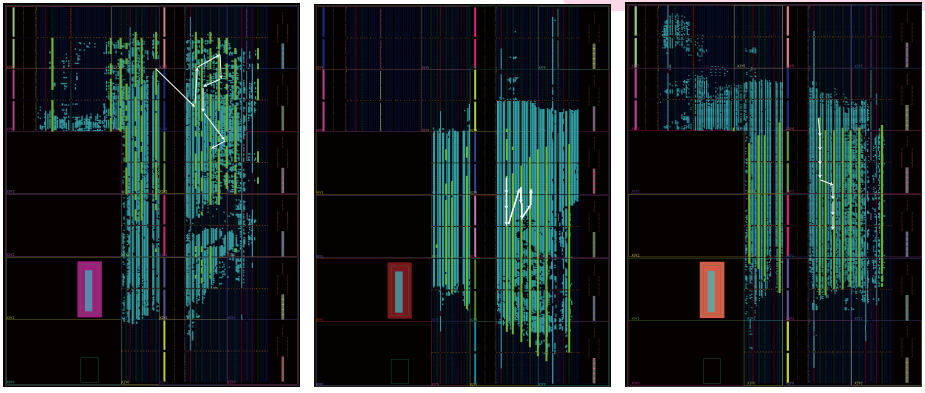
Table: Experiment Results

Runtime Profiling



Runtime profiling: (a) iSmartDNN and (b) SkyNet.

Datapath Visualizations



(a) (b) (c)

Datapath visualizations of placement layout within different design tools: (a) Vivado, (b) AMF, and (c) DSPlacer.

Conclusion

- When deploying CNNs on FPGAs, placement directly impacts clock frequency. Existing approaches suffer from scalability issues and struggle to automate placement rules.
- We proposed DSPlacer, which combines GCN techniques with optimization methods. It extracts datapath DSPs and their shortest path information to accurately identify critical DSP nodes.
- The placement optimization problem is formulated as a 0-1 integer programming problem and further simplified into a min-cost flow model to improve computational efficiency.
- Experimental results show that DSPlacer improves WNS by 32% compared to Vivado and 65% compared to AMF-Placer, significantly enhancing timing performance.

AI

Security

Systems

EDA

Design

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda ACM