DFGC: DFG-Aware NoC Control Based on Time Stamp Prediction for Dataflow Architecture

Tianyu Liu[†‡], Wenming Li [†‡*], Zhihua Fan [†‡]

[†] State Key Lab of Processors, Institute of Computing Technology, CAS, Beijing, China
Email: {liutianyu, liwenming, fanzhihua}@ict.ac.cn

[‡] School of Computer Science and Technology, UCAS, Beijing, China

INTRODUCTION

- CGRA
- Over-serialization calls for hardware flexibility
- Dataflow Architecture
- Hardware & Software Co-Design:

Add acceptable hardware flexibility to reduce the burden of software scheduling.

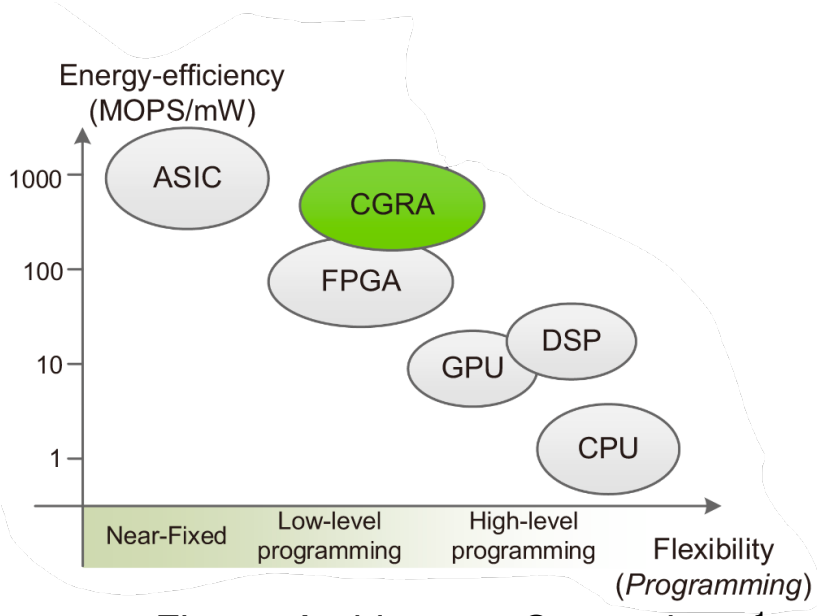

Figure: Architecture Comparison.[1]

- DFGC (DFG-aware NoC Control)
- DFGTE algorithm for rough prediction (TimeStamp)
- Dataflow control mechanism & modeling
- DFG-aware hardware design (PE & Router)

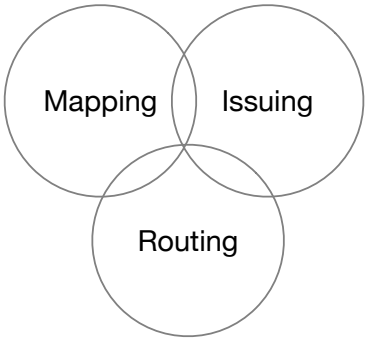

Figure: Key point in schedule.

[1] L. Liu et al., "A Survey of Coarse-Grained Reconfigurable Architecture and Design: Taxonomy, Challenges, and Applications," ACM Comput. Surv., vol. 52, no. 6, p. 118:1-118:39, /, doi: 10.1145/3357375.

BACKGROUND

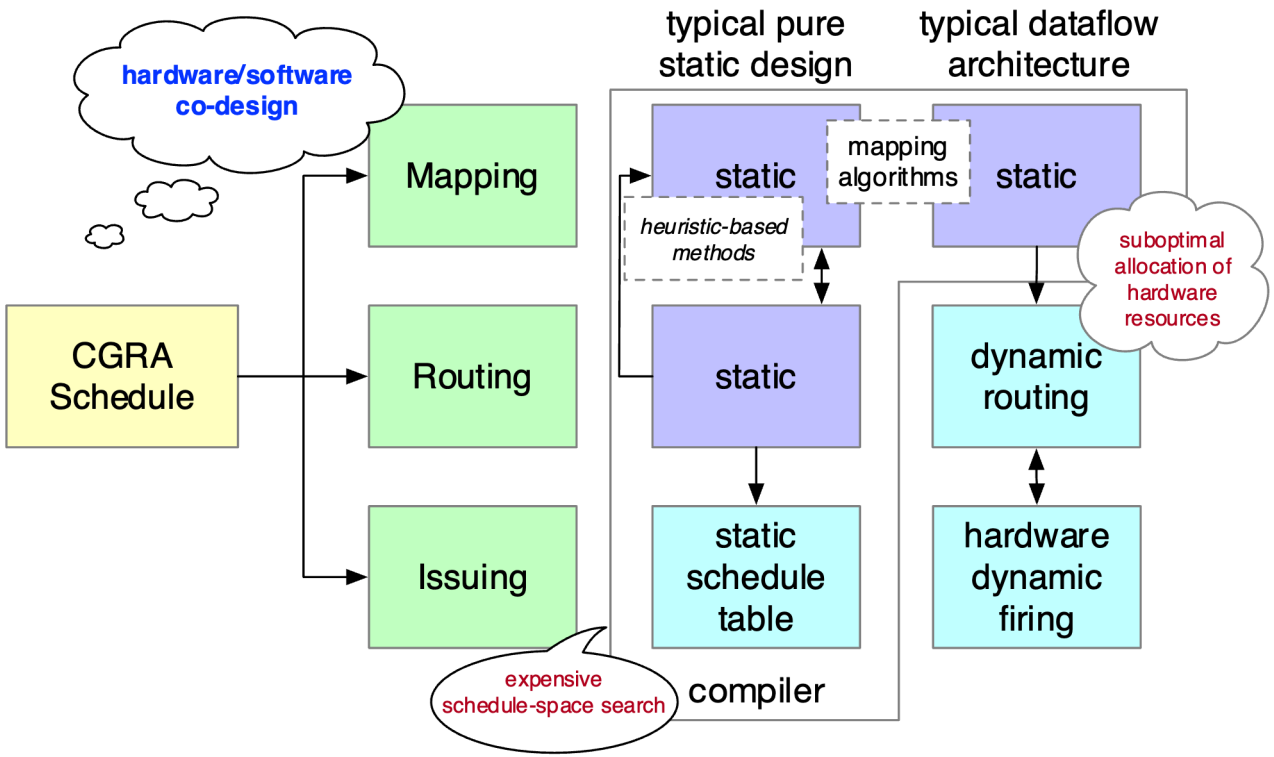

Figure: Scheduling on CGRA.

- CGRA design & schedule method
- Hardware/Software Co-Design
- Dataflow Architecture
- DFGC (DFG-aware NoC Control)

Our solution:

1. Effectively modeling and predicting on DFG (dataflow graph).
2. Optimizing hardware decisions via DFG-aware design.

DFGC DESIGN:

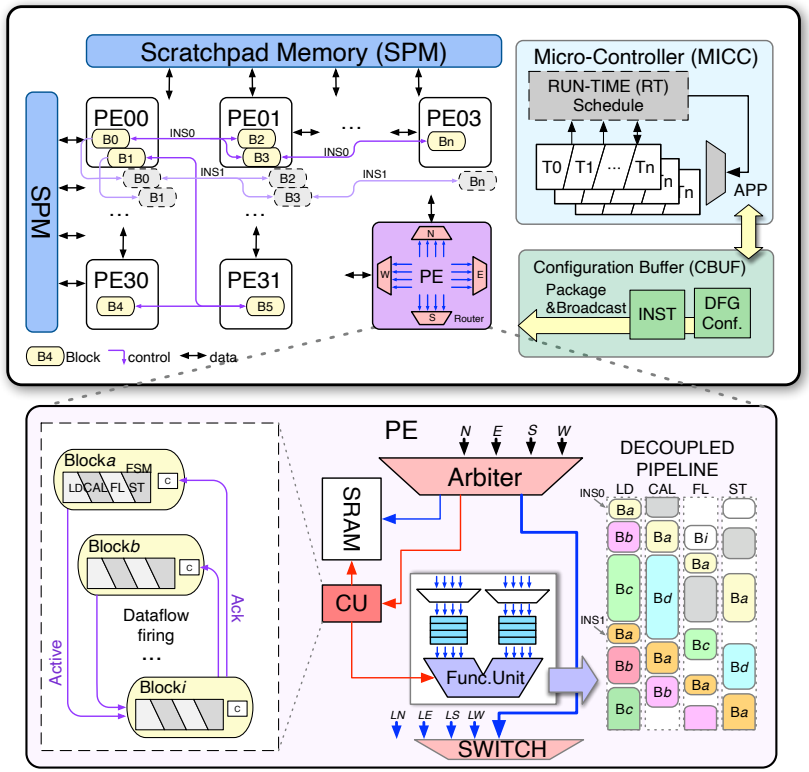

Figure: DFGC control mechanism diagram.

- dataflow dynamic firing
- Instance

Algorithm 1: DFG *TimeStamp* Extraction Algorithm

Input : $\mathcal{G}$ - DFG, $\mathcal{P}$ - PE array, $\mathcal{A}$ - mapping($\mathcal{G} \rightarrow \mathcal{P}$)
Output: $\mathcal{T}$ - *TimeStamp* of ($\mathcal{A}$, $\mathcal{B}$ - blocks, $\mathcal{P}$)

```
1  N ← nodes(G), B ← Instance_expansion(N, G)
2  T ← null, pCosts← null
3  for block i ∈ sorted (B, G) do
4      ranked_costs ← null
5      if i.children is not empty then
6          for block c ∈ i.children do
7              cost ← calPathCost (c, B, pCosts)
8              ranked_costs.append(c, cost)
9          tp ← ranked_costs.Sort
10         T ← T+ tp            ▷ merge duplicate item.
11 Sort_Normalization(T)
12 return T
13 Function calPathCost (block n, B, pCosts)
14     if n ∈ pCosts then
15         return pCosts[n]
16     if n.children is empty then
17         pCosts[n] ← n.execLatency
18         return n.execLatency
19     mcost ← 0
20     for block c ∈ n.children do
21         cost ← calPathCost (c, B, pCosts)
22         mcost ← max(mcost, cost+transLatency(n,c))
23     pCosts[n] ← n.execLatency + mcost
24     return pCosts[n]
```

Figure: DFGTE Algorithm.

- Block dependency
- Predict latency (exec + routing)
- Generate TimeStamp
- Higher TimeStamp have higher priority

DFGC DESIGN:
DFGC Control Scheme

- [compiler]
- DFG loop splitting, DFG mapping and TimeStamp prediction

- [hardware]
- BSC manage local block status.
- Block execution
- Message packeting & transmission
- Bypass on NoC under guide of TP
- Hardware-autonomous routing
- TP update

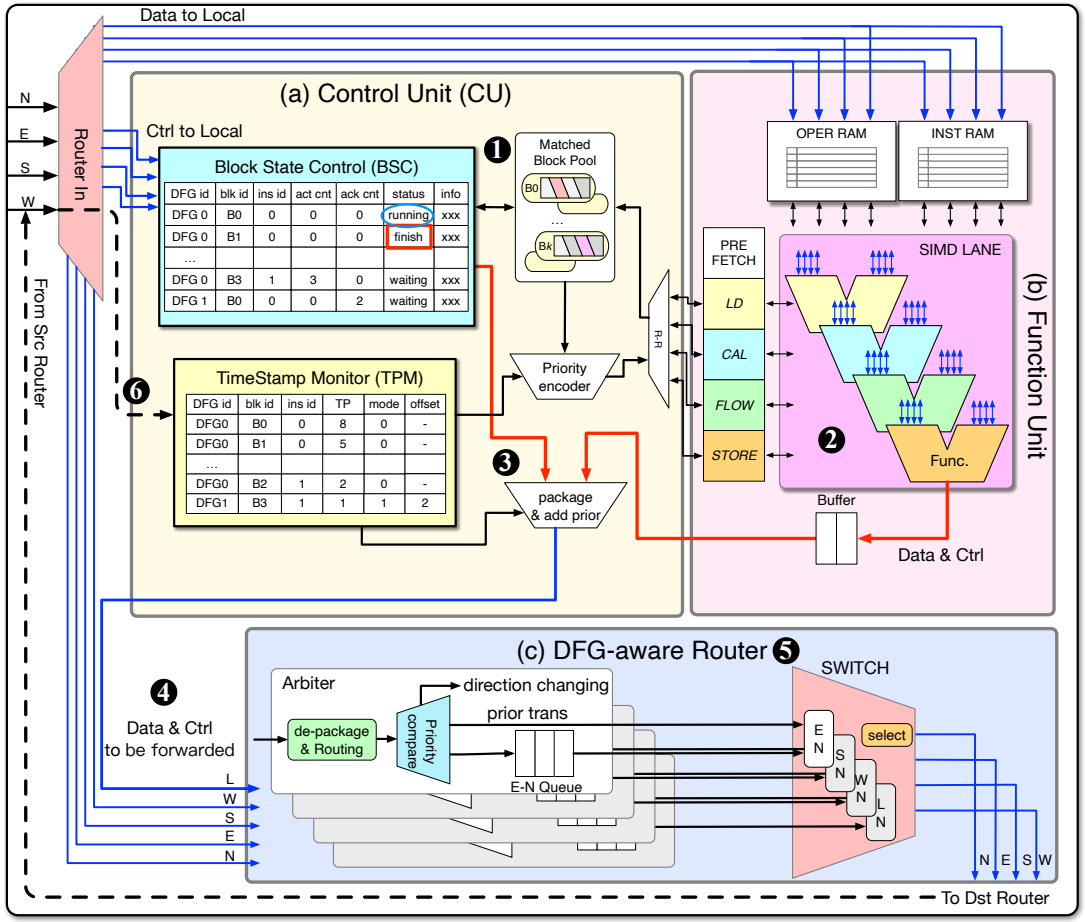

Figure: DFGC PE and NoC hardware design.

EVALUATION

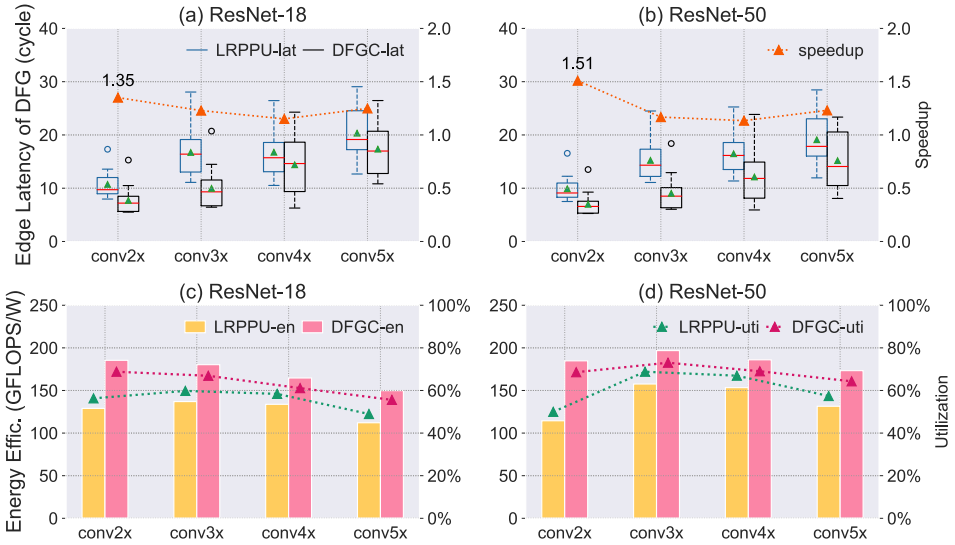

Figure: Performance & Energy effic. versus LRPPU[1].

Versus LRPPU:

1. Gmean 1.25x speedup and in convx of ResNet-18,50. (up to 1.51x)
2. DFG edge latency reduction of 1.42x and 1.45x
3. Energy effic. Improve 1.33x and 1.35x.

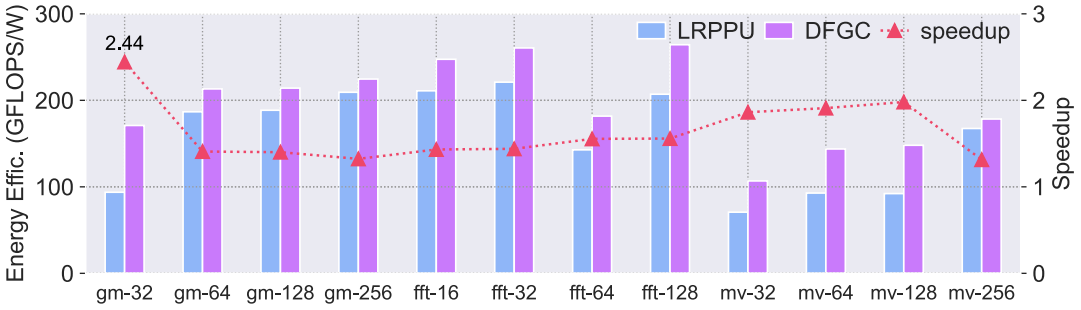

Figure: Performance and energy improvement.
Reach gmean energy efficiency of 189 GFLOPS/W in small-scale GEMM, FFT, and MV computations.

Versus GPU&DSP:

- 5.9x V100, 4.2x Jetson, 8.9x DSP (energy effic.)

Versus CGRA:

- 1.8x HyCube, 2x REVEL (energy effic.)

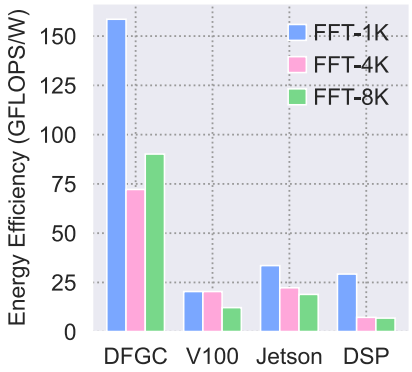

Figure: Energy effic. compared to GPU and DSP.

[1] X. Wu, *et al.*, "LRP: Predictive output activation based on SVD approach for CNN s acceleration," in *DATE*, 2022.

CONCLUSION

- The significance of hardware/software co-design
- DFGC fills the gap between software mapping and hardware scheduling on high flexible architectures such as dataflow.
- The possibility of using actual data transmission condition statistics on PE array to guide the formation of optimal mapping solutions.

Q&A