Design Space Exploration for Loop Nests: Unrolling, Pipelining, and Communication Granularity

(Expanded notes distilled from slides)

September 2, 2025

Abstract

These notes formalize a pragmatic design space exploration (DSE) workflow for loop–intensive programs. The *design knobs* are (i) per–stage loop unroll factors and (ii) per–edge communication granularity. We define a minimal cost model, legality constraints, and a greedy multi–stage search that iteratively balances a pipeline under an area (capacity) budget. The presentation mirrors the slides while adding definitions, symbols, and pseudocode suitable for implementation.

1 Problem statement

Consider a program decomposed into a sequence of computational stages $S = \{s_1, \ldots, s_m\}$ connected by data dependencies (edges) $E \subseteq S \times S$. Each stage is a loop nest that produces/consumes named values $v \in \mathcal{V}$ (e.g. `peak`, `feature_x`). We wish to select:

- an **unroll factor** $u_s \in \mathbb{N}$ for each stage s, and
- a **communication granularity** $g_e \in \mathbb{N}$ (bytes/msg) for each edge $e \in E$,

so as to minimise runtime (or maximise throughput) under a *capacity* (area/memory) constraint.

Notation. Let B_e be the total number of bytes transferred along edge $e = (s \to t)$ per steady–state item, and L the link–level per–message latency. Let BW be the sustained bandwidth (bytes/s) for that interconnect. The **communication time model** for e at granularity g_e is

$$T_{\text{comm}}(e, g_e) = \left\lceil \frac{B_e}{g_e} \right\rceil \left(L + \frac{g_e}{BW} \right). \tag{1}$$

Smaller g_e yields more messages (higher latency cost), larger g_e reduces messages but increases per–message payload, stressing buffers. A *valid granularity* must satisfy alignment and buffering constraints (Sec. 3.1).

For stage s we assume a simple **area–time model** parameterised by its unroll factor u_s:

$$A(s, u_s) = A_s^{\text{base}} + u_s \, A_s^{\text{unit}}, \tag{2}$$

$$T_{\text{comp}}(s, u_s) = \frac{I_s}{\min(u_s, P_s)} \cdot II_s + T_s^{\text{setup}}. \tag{3}$$

Here I_s is the iteration count, II_s the initiation interval (cycles) when pipelined, P_s the available parallel resources, and A_s^{unit} the per–replica area cost when unrolled. Equation (3) captures the first–order speedup from unrolling/pipelining until resources saturate.

The **pipeline time** for design $d = \{u_s\}, \{g_e\}$ is

$$T(d) \;=\; \max_{s \in S} \Big(T_{\text{comp}}(s, u_s) + \sum_{e \in \text{in}(s)} T_{\text{comm}}(e, g_e) \Big). \tag{4}$$

The **capacity constraint** is

$$\sum_{s \in S} A(s, u_s) \;+\; \sum_{e \in E} \text{Buf}(e, g_e) \;\le\; A_{\max}, \tag{5}$$

where $\text{Buf}(e, g_e)$ is the buffer area required by granularity g_e and chosen depth (Sec. 3.1).

1.1 Per–stage read/write demand and inter–stage communication

For each value v used or defined by a stage s we define the *read/write demand* per item:

$$R_s(v), \quad W_s(v) \quad \text{[bytes]}. \tag{6}$$

For each edge $e = (s \to t)$ that moves value v we define the *communication demand*

$$CED_{s \to t}(v) \;=\; W_s(v) \;=\; R_t(v)\,, \tag{7}$$

and accumulate $B_e = \sum_v CED_{s \to t}(v)$.[1]

2 Step 1: Pipeline and communication analysis

Goal. Partition the loop nest(s) into stages S, build the dependency graph E, and compute the per–stage read/write demand and per–edge communication demand (Sec. 1.1). This step is independent of the eventual choice of u_s and g_e. It provides the payloads B_e used later by Eq. (1).

Outcome. A typed, acyclic stage graph with:

1. $R_s(v), W_s(v)$ for all stage/value pairs (s, v),
2. B_e for all edges $e = (s \to t)$,
3. base performance/area parameters $\{A_s^{\text{base}}, A_s^{\text{unit}}, II_s, P_s, T_s^{\text{setup}}\}$.

3 Step 2: Single–stage solutions in isolation

For each stage s:

1. Enumerate *feasible unroll factors* $u_s \in \mathcal{U}_s$ that obey functional constraints (e.g. divisor of loop bounds or vector width) and local resource limits ($A(s, u_s) \le A_{\max}^{\text{local}}$).
2. Derive the *valid granularity set* for every incident edge e (Sec. 3.1).
3. Record the per–stage Pareto set $\mathcal{P}_s = \{(u_s, A(s, u_s), T_{\text{comp}}(s, u_s))\}$.

[1] The slides abbreviated these with `RDAD` (read/write demand) and `CED` (communication edge demand).

Algorithm 1 Greedy DSE for unroll factors and granularities

Input: stage set S, edges E, per–stage Pareto sets $\mathcal{P}_s$, legal granularity sets $\mathcal{G}_e$, capacity $A_{\max}$
Initialise design $d \leftarrow$ pick max u_s from $\mathcal{P}_s$ and largest $g_e \in \mathcal{G}_e$ that fits capacity
Evaluate $T(d)$ via (4); keep a visited set $\mathcal{V}$ of $(\{u_s\}, \{g_e\})$
repeat
    Identify bottleneck stage $s^\star \leftarrow \arg\max_s T_s$ and congested edge $e^\star$ on its cut
    Generate neighbors $\mathcal{N}(d)$ by:
    - decreasing $u_{s^\star}$ to the next Pareto point in $\mathcal{P}_{s^\star}$,
    - modifying $g_{e^\star}$ to the nearest legal value in $\mathcal{G}_{e^\star}$ (up or down).
    Filter $\mathcal{N}(d)$ by capacity (5) and $\notin \mathcal{V}$
    if $\mathcal{N}(d)$ is empty **then break**
    $d' \leftarrow \arg\min_{c \in \mathcal{N}(d)} \big(T(c), -\mathrm{eff}(c)\big)$ ▷ lexicographic: time then efficiency
    if $T(d') < T(d)$ **then**
        $d \leftarrow d'$; $\mathcal{V} \leftarrow \mathcal{V} \cup \{d\}$
    elsebreak
until converged
return d

3.1 Granularity legality & buffers

Let $e = (s \to t)$ carry B_e bytes per item and element size b bytes. A granularity g_e is valid iff:

(i) *alignment*: $g_e \equiv 0 \pmod{b}$,

(ii) *bounded message count*: $1 \le \lceil B_e / g_e \rceil \le M_{\max}$,

(iii) *buffer feasibility*: $\mathrm{Buf}(e, g_e) = D_e \cdot g_e \cdot \alpha \le A^{\mathrm{buf}}_{\max}(e)$,

where D_e is the chosen FIFO depth (in messages) and α converts bytes to "area" units for that memory. The legal set is

$$\mathcal{G}_e \;=\; \Big\{ g \in \mathbb{N} \;:\; \text{(i)–(iii) hold} \Big\}. \tag{8}$$

In practice one snaps g to powers of two or cache line sizes.

4 Step 3: Multi–stage heuristic search

Idea. Start from an initial design and make small, legal moves that *relieve the bottleneck*: reduce the unroll factor of the slowest stage (decrease area, possibly increase time), or adjust g_e along the busiest edge to reduce message overhead or buffer usage.

Balanced pipeline heuristic. Call a pipeline *balanced* if stage times are within a factor β:

$$\frac{\max_s T_s}{\min_s T_s} \le \beta \quad (\beta \approx 1.1\text{–}1.3). \tag{9}$$

Moves should reduce the imbalance while respecting capacity (5).

Efficiency and benefit. Besides runtime $T(d)$ we track

$$\mathrm{benefit}(d) = \frac{T_{\mathrm{baseline}}}{T(d)}, \qquad \mathrm{efficiency}(d) = \frac{\mathrm{benefit}(d)}{\sum_s A(s, u_s) + \sum_e \mathrm{Buf}(e, g_e)}. \tag{10}$$

The slide criteria $\mathit{time}(d_i) < \mathit{time}(d_{i-1})$, $\mathit{benefit}(d_i) > \mathit{benefit}(d_{i-1})$, $\mathit{efficiency}(d_i) > \mathit{efficiency}(d_{i-1})$ are enforced by the acceptance rule in Alg. 1.

5 Step 4: Termination and acceptance

Stop when (i) no legal neighbor improves T, (ii) capacity is tight, or (iii) a balanced solution is reached. The final design must satisfy

$$\text{Capacity: } \sum_s A(s, u_s) + \sum_e \text{Buf}(e, g_e) \le A_{\max}, \tag{11}$$

$$\text{Dominance: } T(d) \text{ minimal among visited designs,} \tag{12}$$

$$\text{Pareto sanity: no component uses a dominated point from } \mathcal{P}_s. \tag{13}$$

6 Implementation commentary (from slides to practice)

Transform pipeline. A practical flow is:

Front-end	Parse C/Python to an IR with explicit loops and arrays.
Scalar & loop opts	Constant folding, LICM, simplification.
DSE(1)	Select unroll factors (per stage).
Code transform	Apply unroll-and-jam; expose pipeline structure.
Analysis	Compute R/W demand and inter-stage payloads B_e.
DSE(2)	Choose granularities g_e and re-balance.
Synthesis/Estimation	Produce scheduled C/VHDL + timing/area estimates.

Multiple loop nests. When several loop nests feed each other, the stage graph is hierarchical. Apply Step 1 per nest, then fuse into a global DAG. DSE proceeds on the fused graph with stage-local Pareto sets and global capacity.

7 Worked micro-example (illustrative)

Suppose three stages $S_1 \to S_2 \to S_3$ produce/consume arrays of 32 bit values with per-item payloads $B_{1\to2}$=64 kB and $B_{2\to3}$=8 kB. On a link with L=200 ns and BW=6 GB/s, Eq. (1) gives:

$$T_{\text{comm}}(1{\to}2, g) = \left\lceil \frac{65536}{g} \right\rceil \left(200\,\text{ns} + \frac{g}{6 \cdot 10^9}\right),$$
$$T_{\text{comm}}(2{\to}3, g) = \left\lceil \frac{8192}{g} \right\rceil \left(200\,\text{ns} + \frac{g}{6 \cdot 10^9}\right).$$

Choosing g=4 kB yields roughly 16 and 2 messages respectively; increasing to 8 kB halves message count at the expense of buffer space. The heuristic will usually increase g on the 1→2 edge first (it dominates latency), provided buffers fit.

8 What to measure during experiments

- **Area vs. time staircase.** For each stage, plot u_s vs. $T_{\text{comp}}(s, u_s)$ and mark feasible points under local budgets.
- **End-to-end throughput.** Compare the initial (no unrolling) design to the "fully unrolled" design; show where capacity is exceeded.
- **Sensitivity to g_e.** Sweep g_e on the heaviest edge to visualise the latency/buffer trade-off.

9 Checklist for implementing the slides' algorithm

1. Build a stage/edge graph from code; compute R/W and B_e.
2. Calibrate $A_s^{\text{base}}, A_s^{\text{unit}}, II_s, P_s, T_s^{\text{setup}}$.
3. Generate $\mathcal{U}_s$, $\mathcal{G}_e$ and per–stage Pareto sets $\mathcal{P}_s$.
4. Run Alg. 1 with capacity $A_{\max}$; log visited designs.
5. Report $\{u_s\}$, $\{g_e\}$, area breakdown, $T(d)$, benefit, efficiency.

Reproducibility tip. Keep the model symbolic where possible (e.g. B_e as $M \times N \times$bytes), then substitute numbers for specific benchmarks (DCT, vision kernels, histogram).

Acknowledgement of provenance

These notes were prepared by translating a set of slides titled *"Design Space Exploration (DSE)"* (dated March 31, 2006) into a structured, implementation-ready document. The symbols and equations here are a faithful formalisation of the slide content (unroll factors, pipelined stages, and communication granularities) with standard cost models added where the slides were schematic.