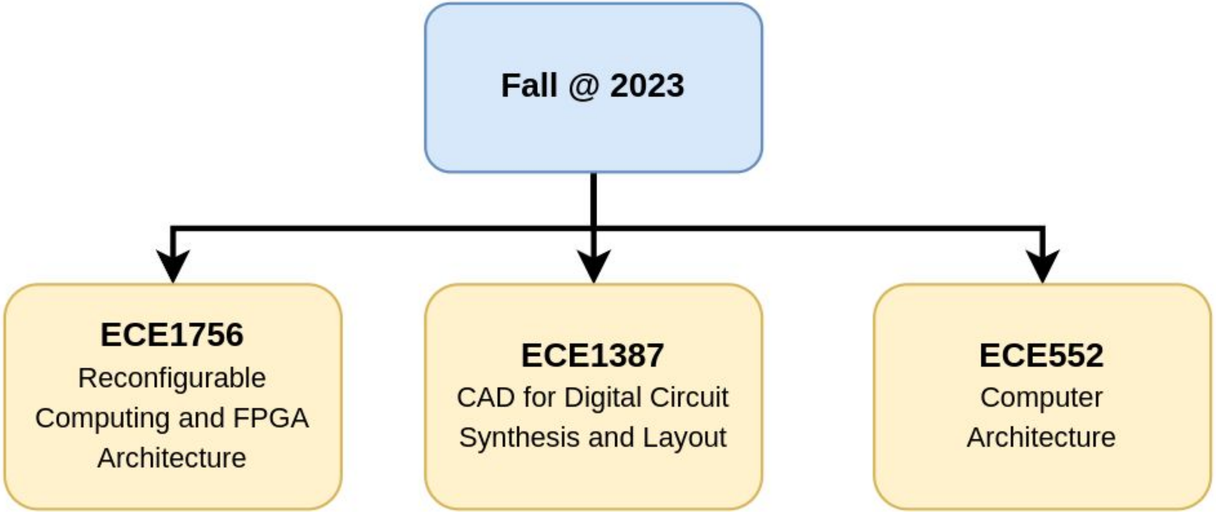
UNIVERSITY OF TORONTO

MASc @ ECE - Update

Student : **Omkar Bhilare**
Advisor : **Prof. Jason Anderson**

Index

■ Courses:

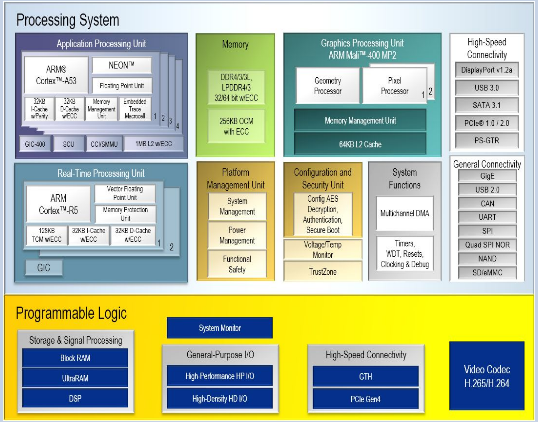


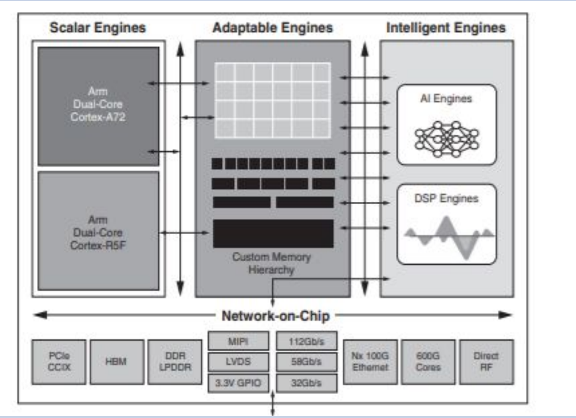
Fall @ 2023

ECE1756
Reconfigurable Computing and FPGA Architecture

ECE1387
CAD for Digital Circuit Synthesis and Layout

ECE552
Computer Architecture





ECE1756: FPGA Course

ECE1756: FPGA Course

- **Goal:** A digital circuit that computes the exponential function e^x for 16-bit fixed-point input values,

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \dots$$

- **Ideally requires 5 Multipliers and 5 Adders:**

$$y = ((((a5x + a4)x + a3)x + a2)x + a1)x + a0$$

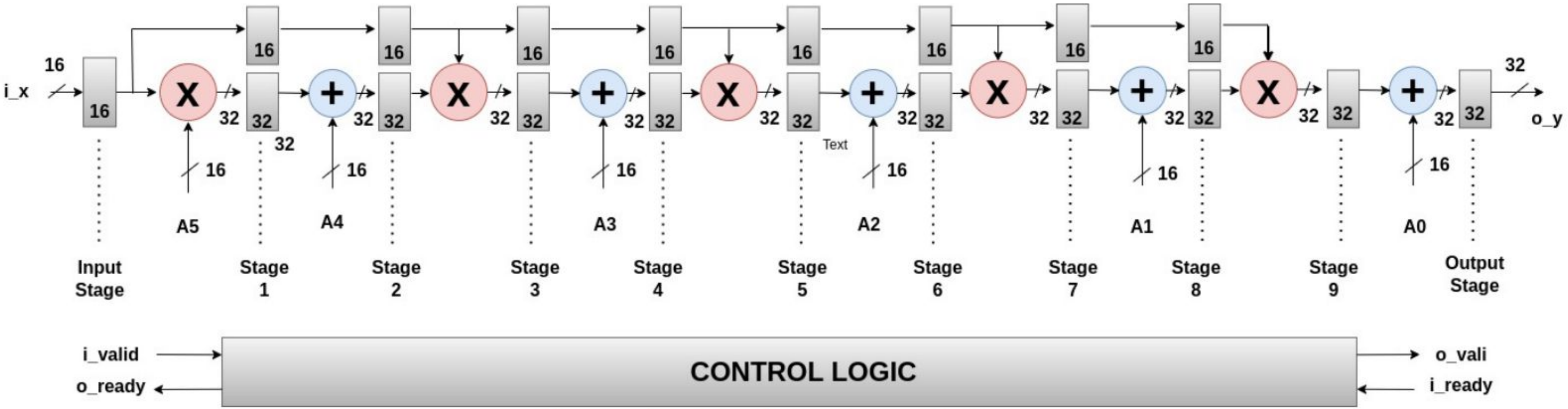
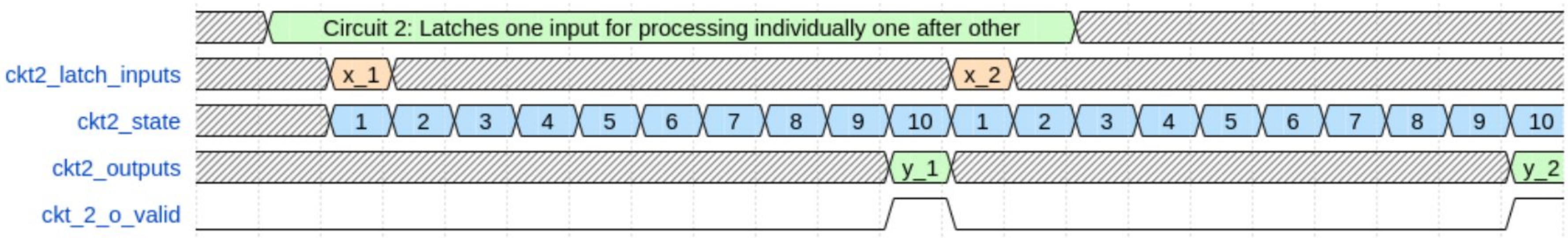
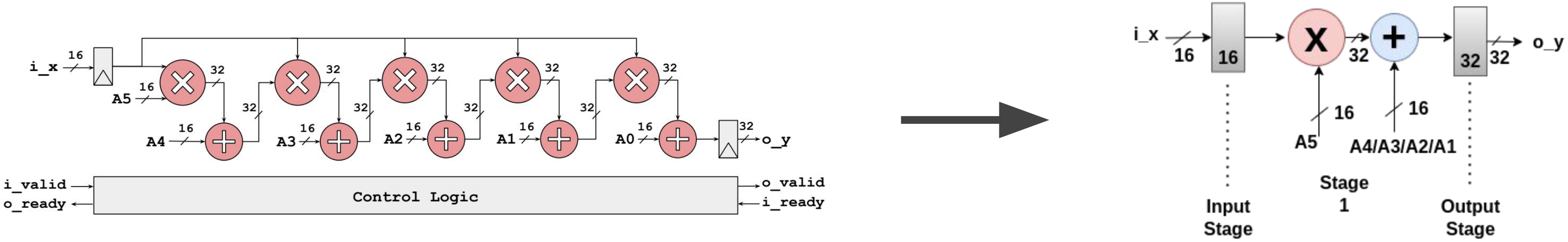


Figure 2: Final Implementation

But what if we have only one multiplier and adder in the system?





Shared HW Architecture

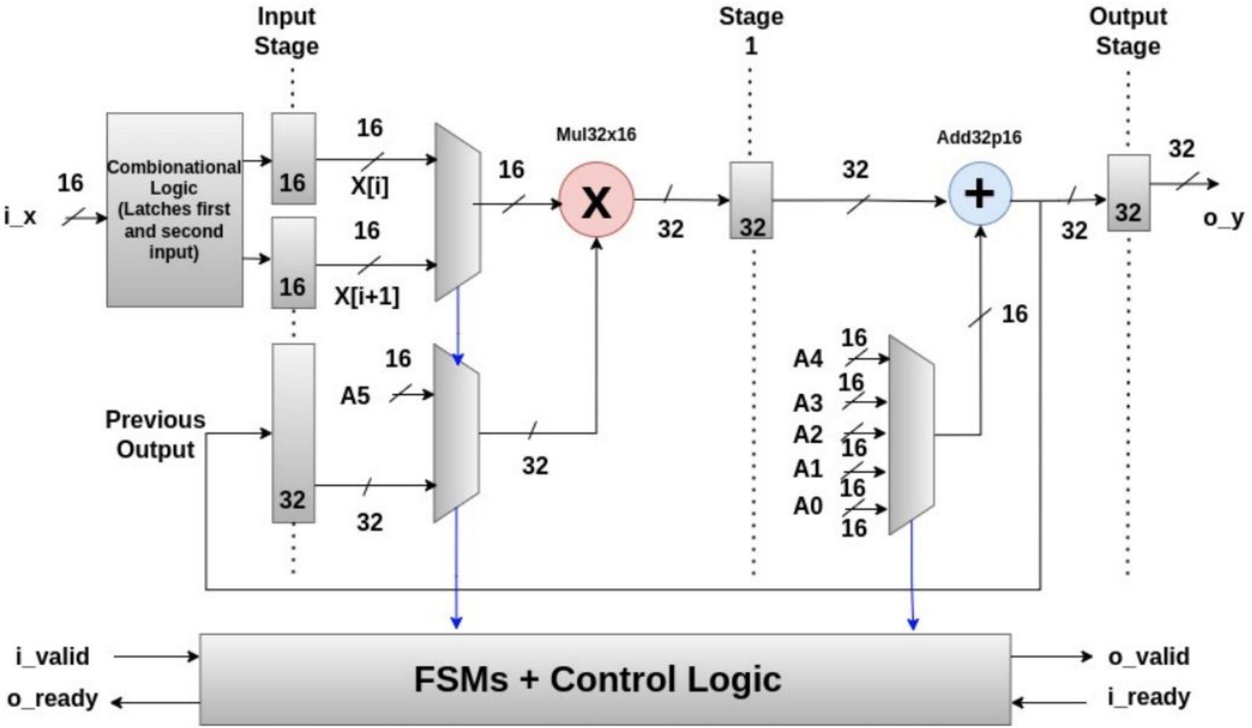


Figure 7: Shared circuit Implementation

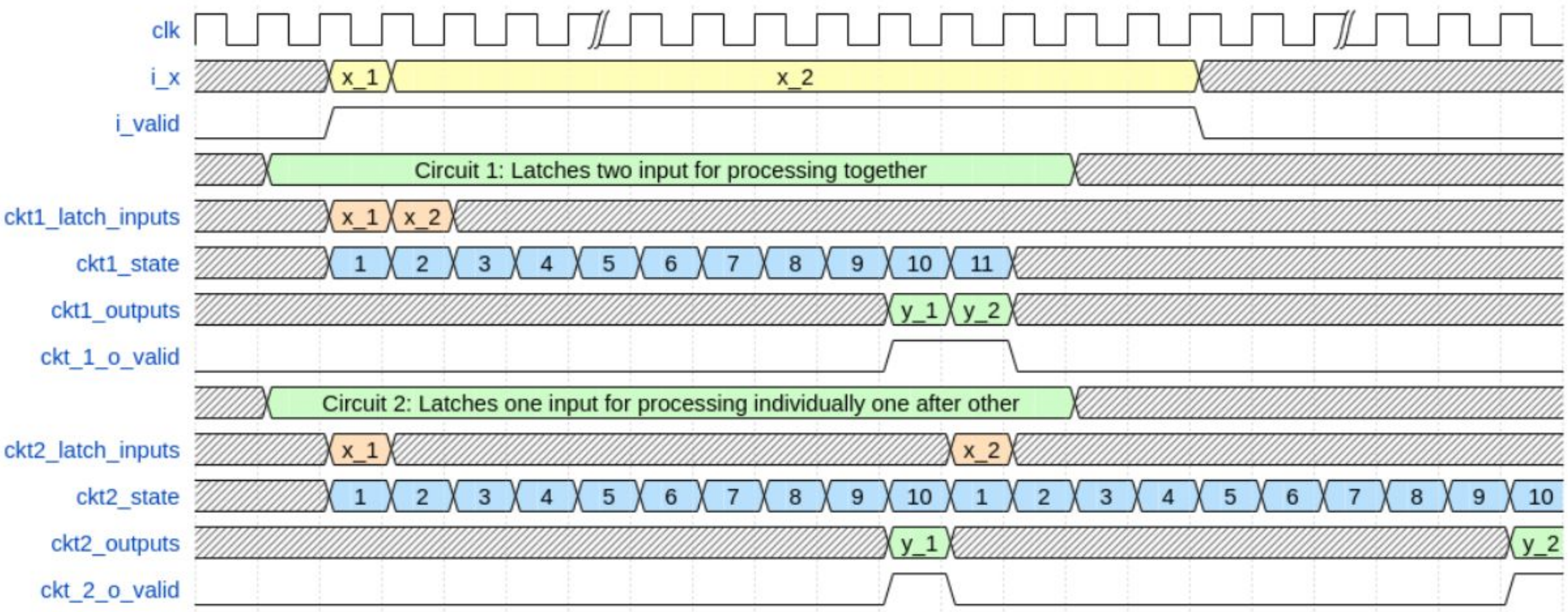
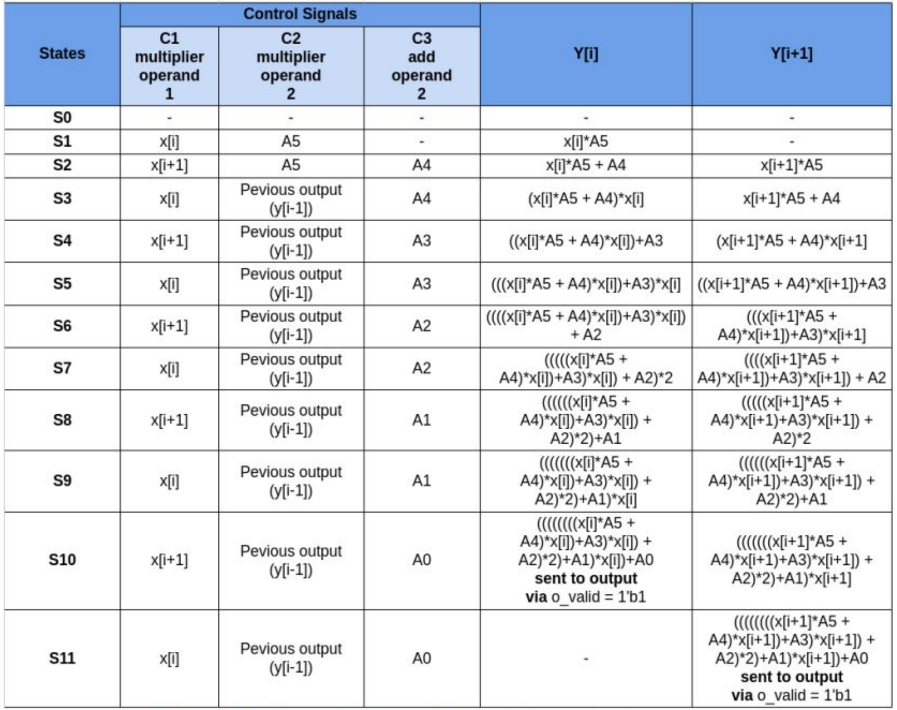


Figure 8: Optimization for multiple inputs acceptance

STATE MACHINE



States	Control Signals			Y[i]	Y[i+1]
	C1 multiplier operand 1	C2 multiplier operand 2	C3 add operand 2		
S0	-	-	-	-	-
S1	x[i]	A5	-	x[i]*A5	-
S2	x[i+1]	A5	A4	x[i]*A5 + A4	x[i+1]*A5
S3	x[i]	Pevious output (y[i-1])	A4	(x[i]*A5 + A4)*x[i]	x[i+1]*A5 + A4
S4	x[i+1]	Pevious output (y[i-1])	A3	((x[i]*A5 + A4)*x[i])+A3	(x[i+1]*A5 + A4)*x[i+1]
S5	x[i]	Pevious output (y[i-1])	A3	(((x[i]*A5 + A4)*x[i])+A3)*x[i]	((x[i+1]*A5 + A4)*x[i+1])+A3
S6	x[i+1]	Pevious output (y[i-1])	A2	((((x[i]*A5 + A4)*x[i])+A3)*x[i]) + A2	(((x[i+1]*A5 + A4)*x[i+1])+A3)*x[i+1]
S7	x[i]	Pevious output (y[i-1])	A2	(((((x[i]*A5 + A4)*x[i])+A3)*x[i]) + A2)*2	((((x[i+1]*A5 + A4)*x[i+1])+A3)*x[i+1]) + A2
S8	x[i+1]	Pevious output (y[i-1])	A1	((((((x[i]*A5 + A4)*x[i])+A3)*x[i]) + A2)*2)+A1	(((((x[i+1]*A5 + A4)*x[i+1)+A3)*x[i+1]) + A2)*2
S9	x[i]	Pevious output (y[i-1])	A1	(((((((x[i]*A5 + A4)*x[i])+A3)*x[i]) + A2)*2)+A1)*x[i]	((((((x[i+1]*A5 + A4)*x[i+1])+A3)*x[i+1]) + A2)*2)+A1
S10	x[i+1]	Pevious output (y[i-1])	A0	((((((((x[i]*A5 + A4)*x[i])+A3)*x[i]) + A2)*2)+A1)*x[i])+A0 **sent to output via** o_valid = 1'b1	(((((((x[i+1]*A5 + A4)*x[i+1)+A3)*x[i+1]) + A2)*2)+A1)*x[i+1]
S11	x[i]	Pevious output (y[i-1])	A0	-	((((((((x[i+1]*A5 + A4)*x[i+1])+A3)*x[i+1]) + A2)*2)+A1)*x[i+1])+A0 **sent to output via** o_valid = 1'b1

Figure 14: State Machine of Shared Circuit Implementation

How it affects Power?

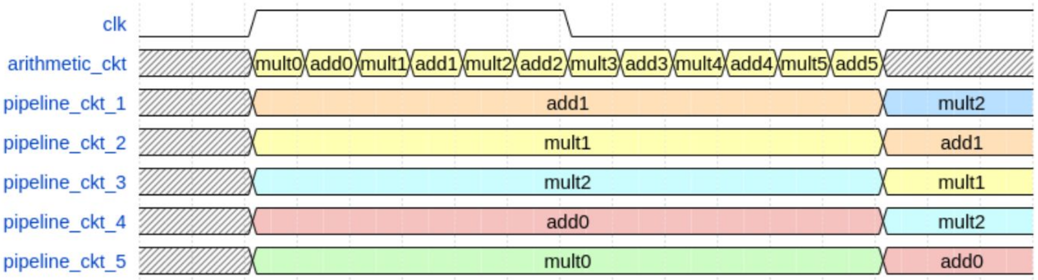
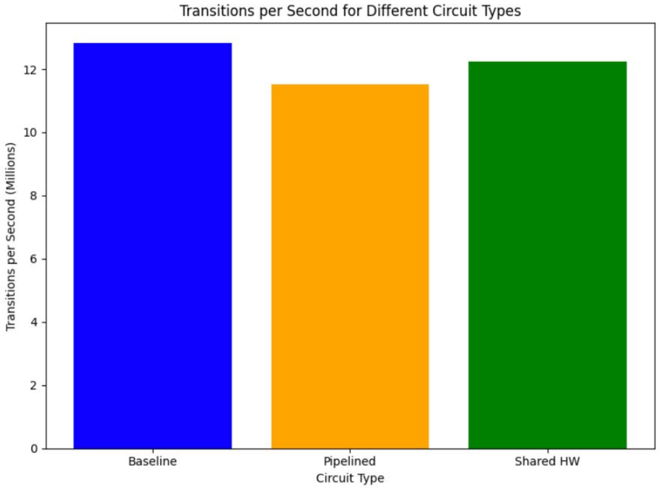
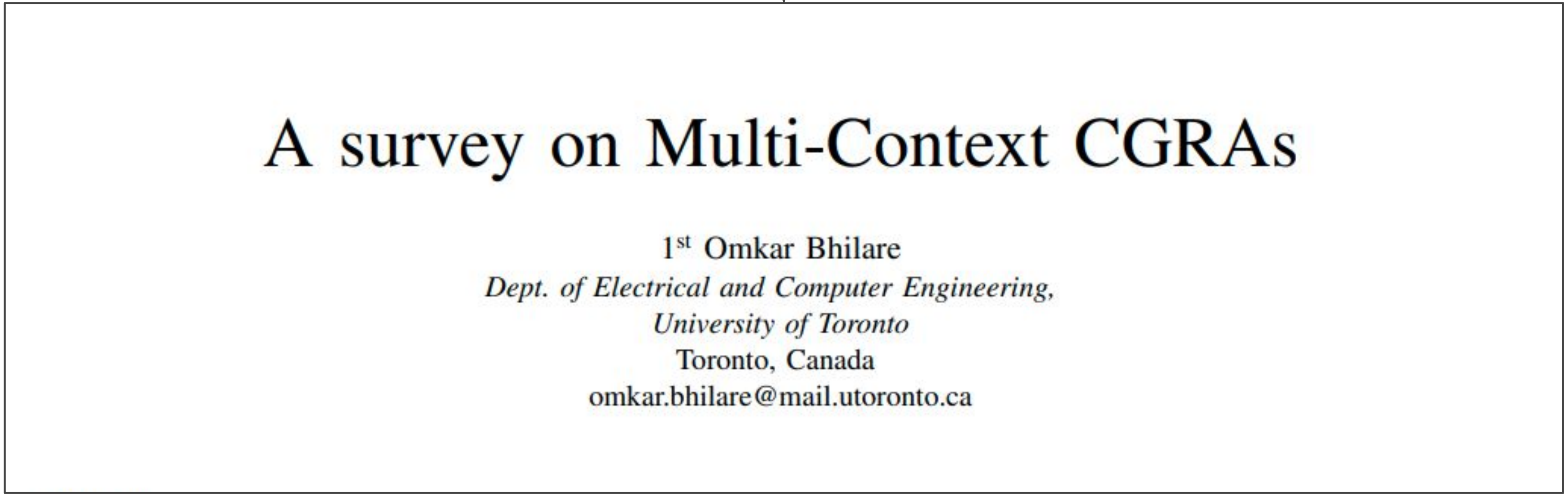


Figure 13: Circuit stages for baseline vs pipeline



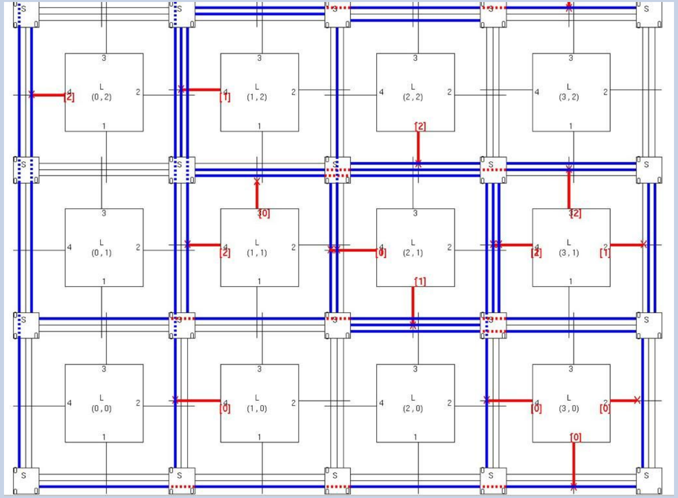
Expanding this idea into Term Paper for the course:

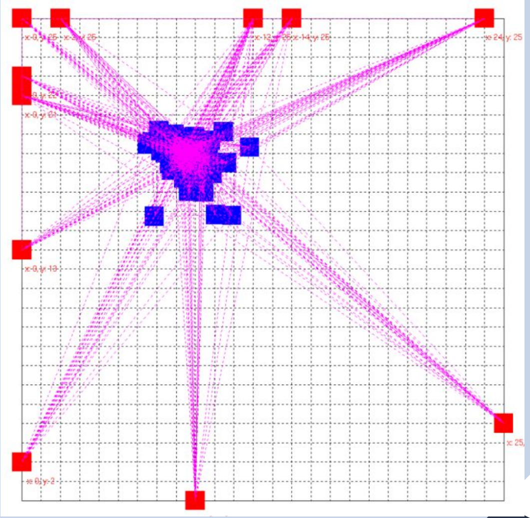


A survey on Multi-Context CGRAs

1st Omkar Bhilare
Dept. of Electrical and Computer Engineering,
University of Toronto
Toronto, Canada
omkar.bhilare@mail.utoronto.ca

If you have any suggestions then please let me know

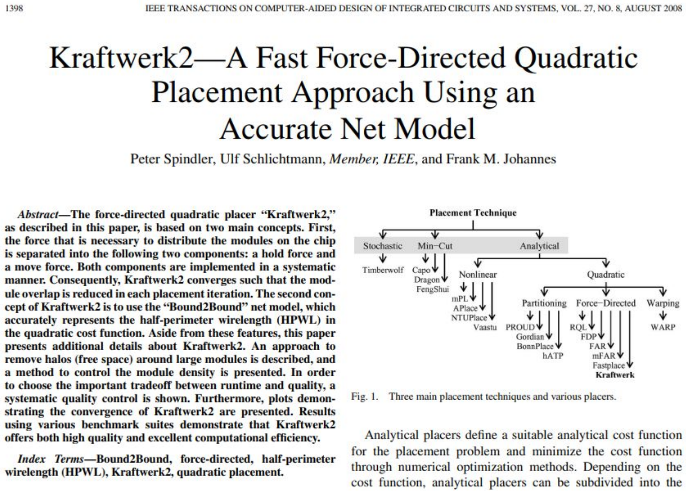




ECE1387: CAD Course

ECE1387: CAD Course

- **Goal:** To implement **analytical placement** using the Clique model and also the spreading of overused bins based on Darav’s algorithm.
- **Idea of AP:** Write an Equation whose minimum is placement. (Solving the problem analytically in **One Shot!**)



Kraftwerk2—A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model

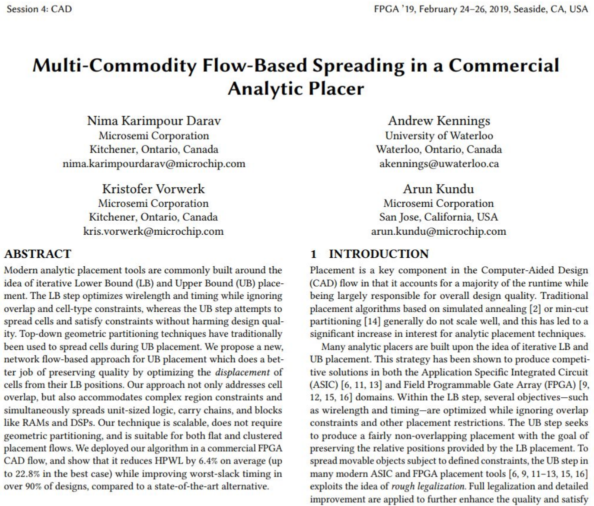
Peter Spindler, Ulf Schlichtmann, *Member, IEEE*, and Frank M. Johannes

***Abstract*—The force-directed quadratic placer “Kraftwerk2,” as described in this paper, is based on two main concepts. First, the force that is necessary to distribute the modules on the chip is separated into the following two components: a hold force and a move force. Both components are implemented in a systematic manner. Consequently, Kraftwerk2 converges such that the module overlap is reduced in each placement iteration. The second concept of Kraftwerk2 is to use the “Bound2Bound” net model, which accurately represents the half-perimeter wirelength (HPWL) in the quadratic cost function. Aside from these features, this paper presents additional details about Kraftwerk2. An approach to remove halos (free space) around large modules is described, and a method to control the module density is presented. In order to choose the important tradeoff between runtime and quality, a systematic quality control is shown. Furthermore, plots demonstrating the convergence of Kraftwerk2 are presented. Results using various benchmark suites demonstrate that Kraftwerk2 offers both high quality and excellent computational efficiency.**

***Index Terms*—Bound2Bound, force-directed, half-perimeter wirelength (HPWL), Kraftwerk2, quadratic placement.**

Fig. 1. Three main placement techniques and various placers.

Analytical placers define a suitable analytical cost function for the placement problem and minimize the cost function through numerical optimization methods. Depending on the cost function, analytical placers can be subdivided into the



Multi-Commodity Flow-Based Spreading in a Commercial Analytic Placer

Nima Karimpour Darav
Microsemi Corporation
Kitchener, Ontario, Canada
nima.karimpourdarav@microchip.com

Andrew Kennings
University of Waterloo
Waterloo, Ontario, Canada
akennings@uwaterloo.ca

Kristofer Vorwerk
Microsemi Corporation
Kitchener, Ontario, Canada
kris.vorwerk@microchip.com

Arun Kundu
Microsemi Corporation
San Jose, California, USA
arun.kundu@microchip.com

ABSTRACT

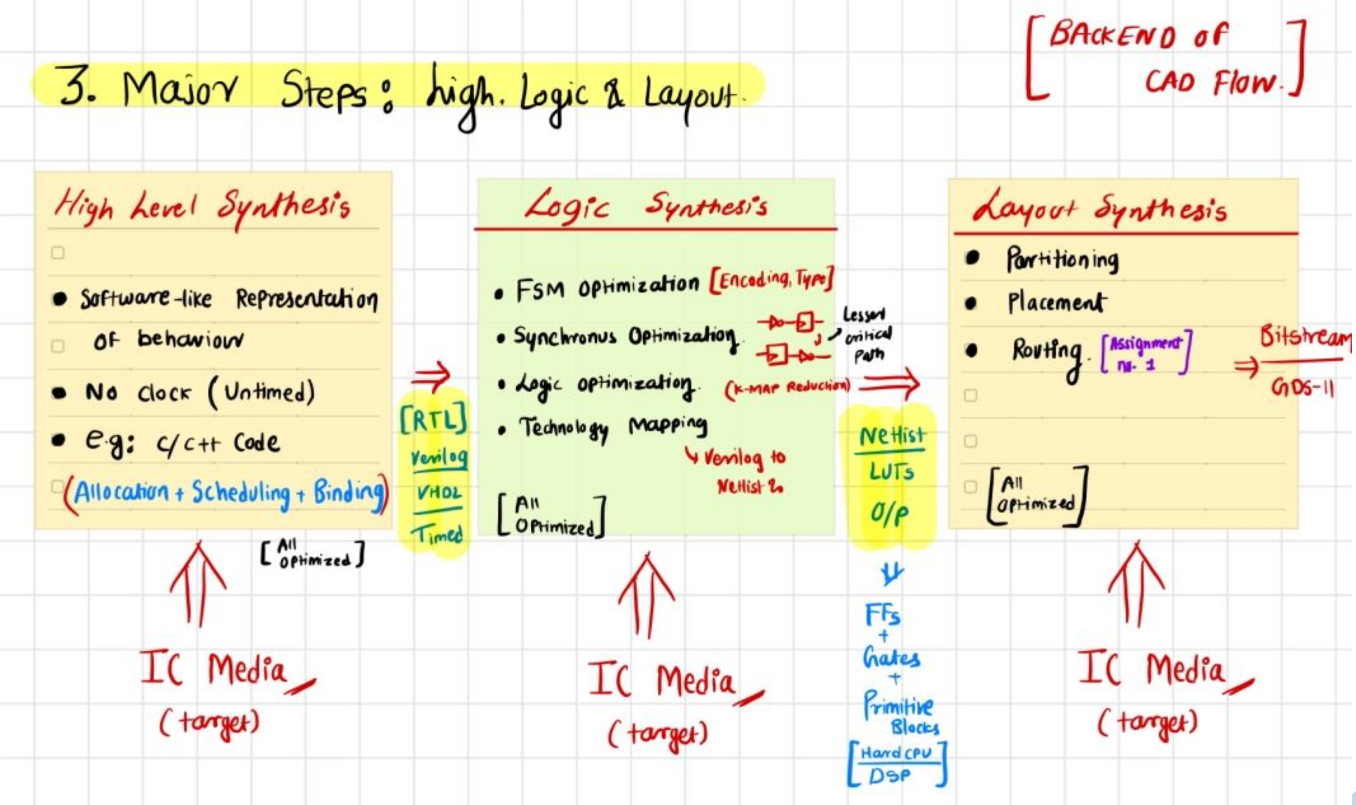
Modern analytic placement tools are commonly built around the idea of iterative Lower Bound (LB) and Upper Bound (UB) placement. The LB step optimizes wirelength and timing while ignoring overlap and cell-type constraints, whereas the UB step attempts to spread cells and satisfy constraints without harming design quality. Top-down geometric partitioning techniques have traditionally been used to spread cells during UB placement. We propose a new, network flow-based approach for UB placement which does a better job of preserving quality by optimizing the *displacement* of cells from their LB positions. Our approach not only addresses cell overlap, but also accommodates complex region constraints and simultaneously spreads unit-sized logic, carry chains, and blocks like RAMs and DSPs. Our technique is scalable, does not require geometric partitioning, and is suitable for both flat and clustered placement flows. We deployed our algorithm in a commercial FPGA CAD flow, and show that it reduces HPWL by 6.4% on average (up to 22.8% in the best case) while improving worst-slack timing in over 90% of designs, compared to a state-of-the-art alternative.

1 INTRODUCTION

Placement is a key component in the Computer-Aided Design (CAD) flow in that it accounts for a majority of the runtime while being largely responsible for overall design quality. Traditional placement algorithms based on simulated annealing [2] or min-cut partitioning [14] generally do not scale well, and this has led to a significant increase in interest for analytic placement techniques.

Many analytic placers are built upon the idea of iterative LB and UB placement. This strategy has been shown to produce competitive solutions in both the Application Specific Integrated Circuit (ASIC) [6, 11, 13] and Field Programmable Gate Array (FPGA) [9, 12, 15, 16] domains. Within the LB step, several objectives—such as wirelength and timing—are optimized while ignoring overlap constraints and other placement restrictions. The UB step seeks to produce a fairly non-overlapping placement with the goal of preserving the relative positions provided by the LB placement. To spread movable objects subject to defined constraints, the UB step in many modern ASIC and FPGA placement tools [6, 9, 11–13, 15, 16] exploits the idea of *rough legalization*. Full legalization and detailed improvement are applied to further enhance the quality and satisfy

CAD FLOW



3. Major Steps: high. Logic & Layout.

[BACKEND of CAD Flow.]

High Level Synthesis

- Software-like Representation of behaviour
- No clock (Untimed)
- e.g: C/C++ Code

(Allocation + Scheduling + Binding)

[All optimized]

⇒ [RTL] Verilog / VHDL / Timed

Logic Synthesis

- FSM optimization [Encoding, Type]
- Synchronus Optimization → Lesser critical Path
- Logic optimization. (K-MAP Reduction)
- Technology Mapping
  - ↳ Verilog to Netlist 2o

[All Optimized]

⇒ Netlist / LUTs / O/P

⇓

FFs + Gates + Primitive Blocks [Hard CPU / DSP]

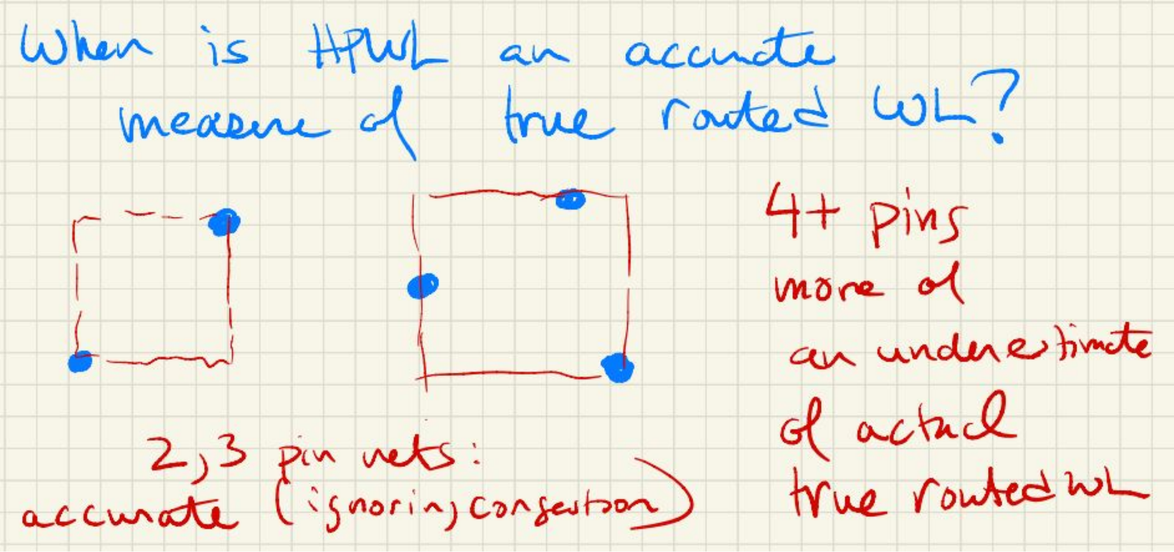
Layout Synthesis

- Partitioning
- Placement
- Routing. [Assignment no. 1]

[All optimized]

⇒ Bitstream / GDS-II

We need estimated wirelength for Placement

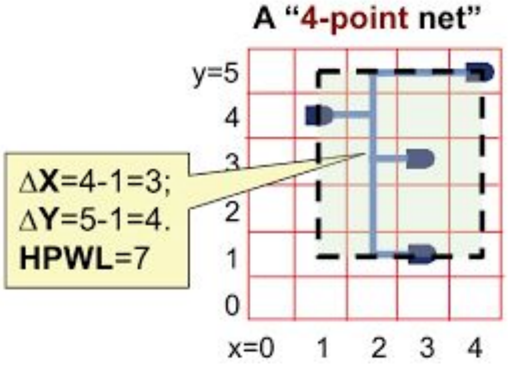


When is HPWL an accurate measure of true routed WL?

2,3 pin nets: accurate (ignoring congestion)

4+ pins more of an underestimate of actual true routed WL

HPWL



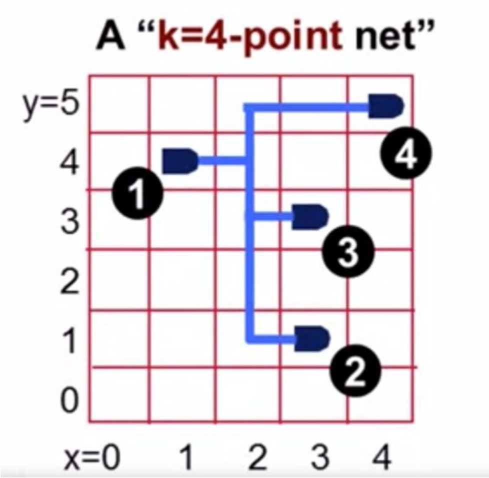
A "4-point net"

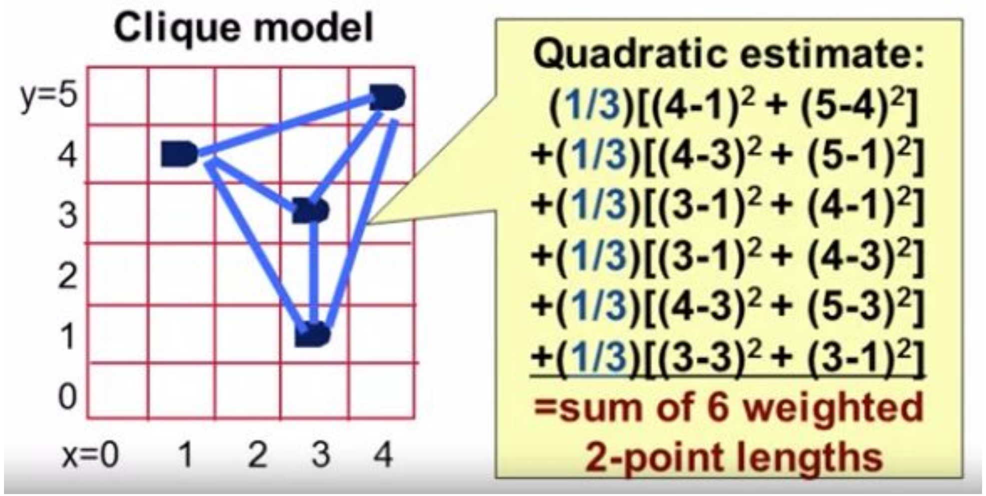
ΔX=4-1=3;
ΔY=5-1=4.
HPWL=7

Understatement for 4+ Nets

CLQUE MODEL

- **Idea of Clique Model:** We can convert K-Net into **K(K-1)/2** Nets. (Each Nets weight gets changed.)





Quadratic estimate:

$(1/3)[(4-1)^2 + (5-4)^2]$
$+(1/3)[(4-3)^2 + (5-1)^2]$
$+(1/3)[(3-1)^2 + (4-1)^2]$
$+(1/3)[(3-1)^2 + (4-3)^2]$
$+(1/3)[(4-3)^2 + (5-3)^2]$
$\underline{+(1/3)[(3-3)^2 + (3-1)^2]}$
=sum of 6 weighted 2-point lengths

Analytical Placement

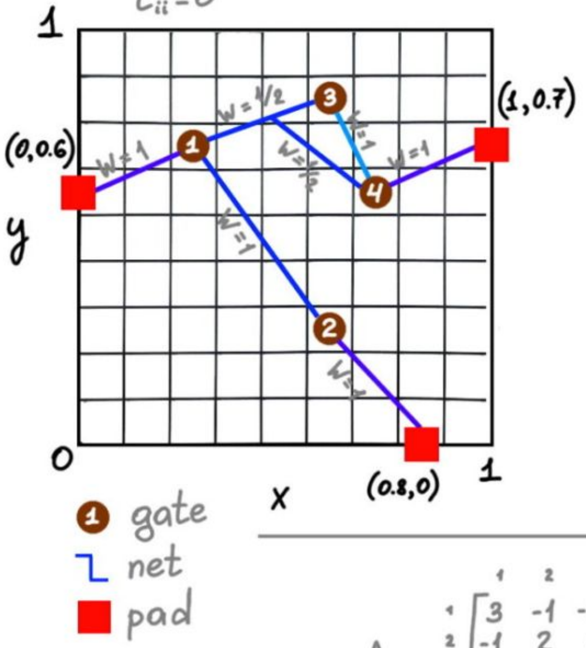
N gates
$N = 4$

Connectivity matrix $N \times N$

$C_{ij\neq i}$ = total weight of the wires connecting gates i and j

$C_{ii} = 0$

$$C = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0.5 & 0.5 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 0.5 & 0 & 0 & 1.5 \\ 4 & 0.5 & 0 & 1.5 & 0 \end{array}$$



$$Ax = b_x \qquad Ay = b_y$$

$$A_{ij\neq i} = -C_{ij}$$

$$A_{ii} = \sum_{j=1}^{N} C_{ij} + \sum W_{pad\_gate\,i}$$

$$b_x = \sum_{pad} W_{pad\_gate\,i} \cdot x_{pad}$$

$$b_y = \sum_{pad} W_{pad\_gate\,i} \cdot y_{pad}$$

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 3 & -1 & -0.5 & -0.5 \\ 2 & -1 & 2 & 0 & 0 \\ 3 & -0.5 & 0 & 2 & -1.5 \\ 4 & -0.5 & 0 & -1.5 & 3 \end{array} \quad b_x = \begin{bmatrix} 0 \\ 0.8 \\ 0 \\ 1 \end{bmatrix} \quad b_y = \begin{bmatrix} 0.6 \\ 0 \\ 0 \\ 0.7 \end{bmatrix}$$

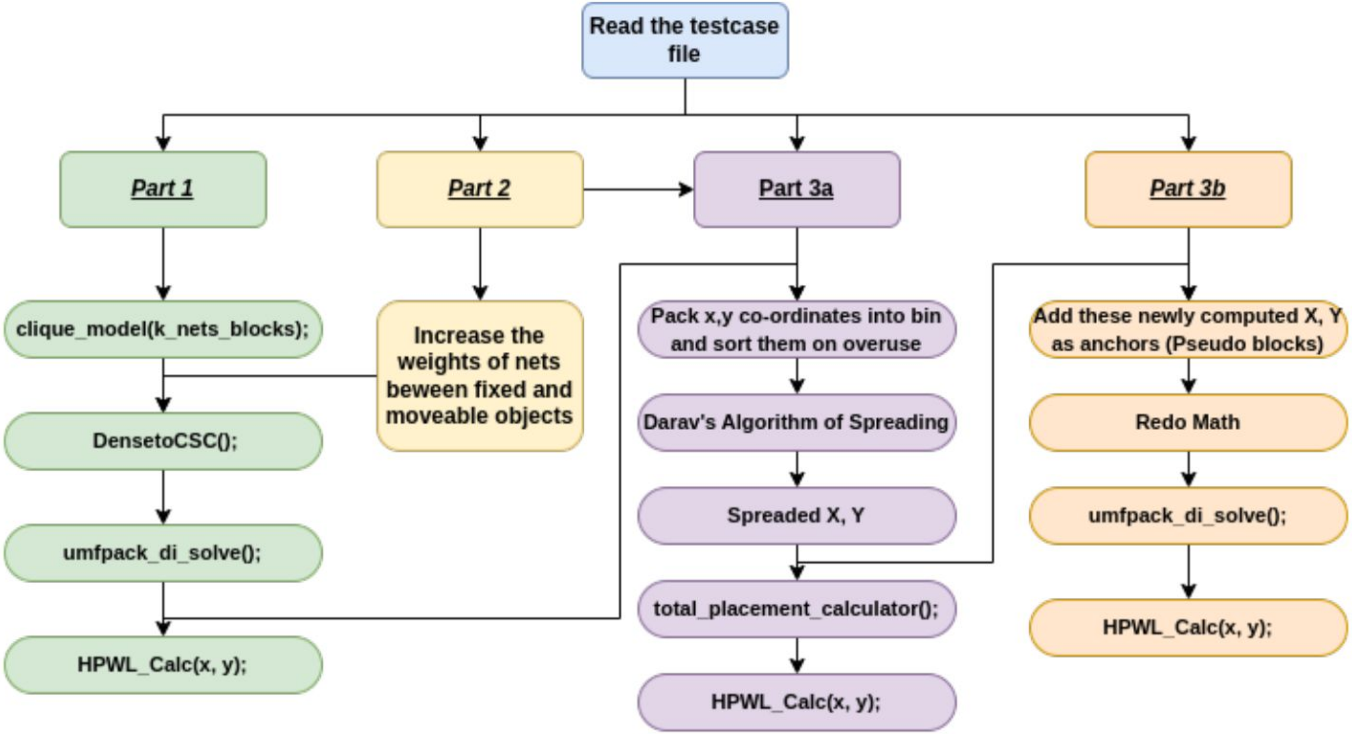
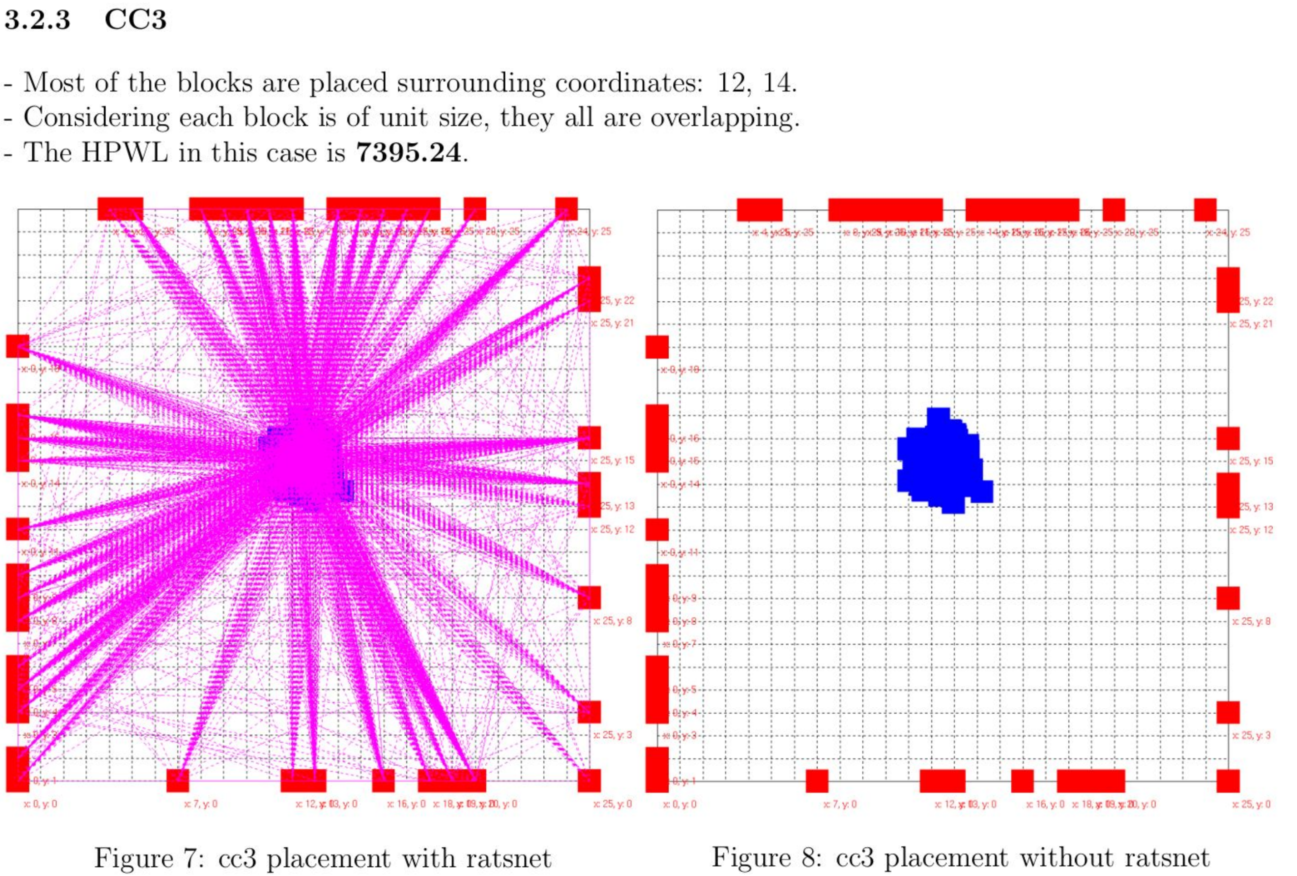


Figure 1: Flow Diagram of Assignment 2

Need to Spread Placement.



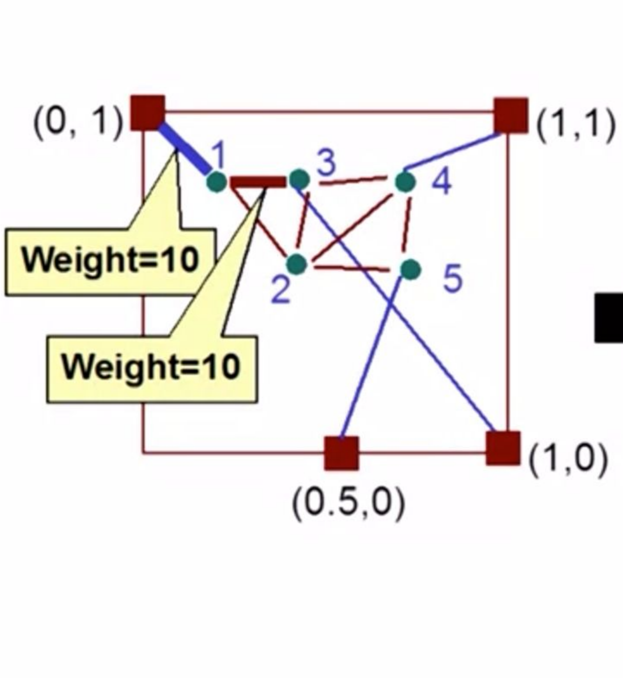
3.2.3 CC3

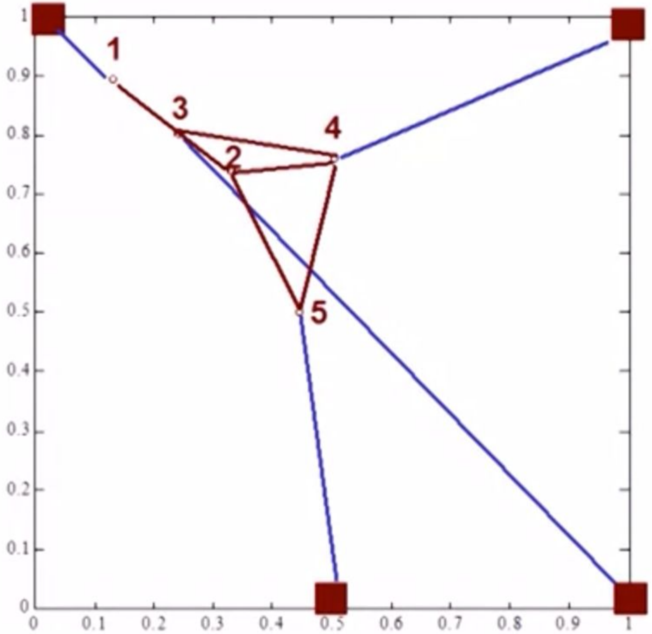
- Most of the blocks are placed surrounding coordinates: 12, 14.
- Considering each block is of unit size, they all are overlapping.
- The HPWL in this case is **7395.24**.

Figure 7: cc3 placement with ratsnet

Figure 8: cc3 placement without ratsnet

One Option: Keep blocks connecting fixed pad weights very high





One Option: Keep blocks connecting fixed pad weights very high

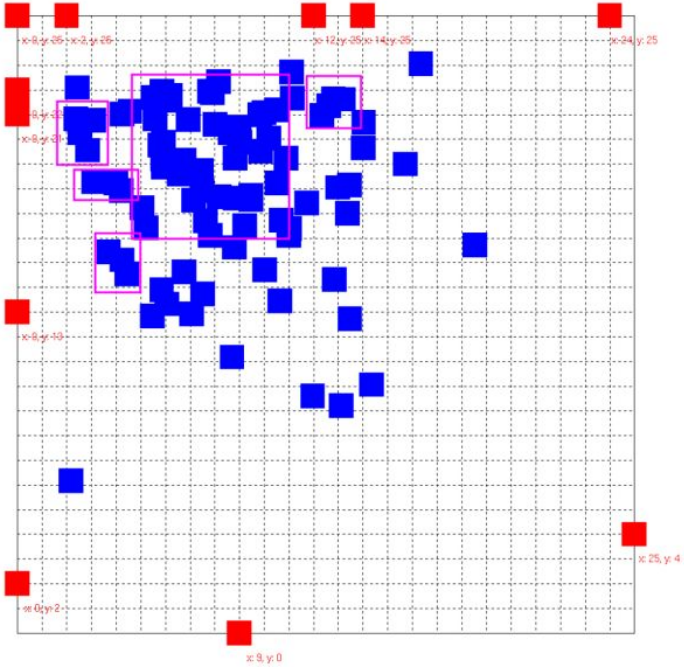


Figure 11: CC2: Weak weight between fixed and variable blocks

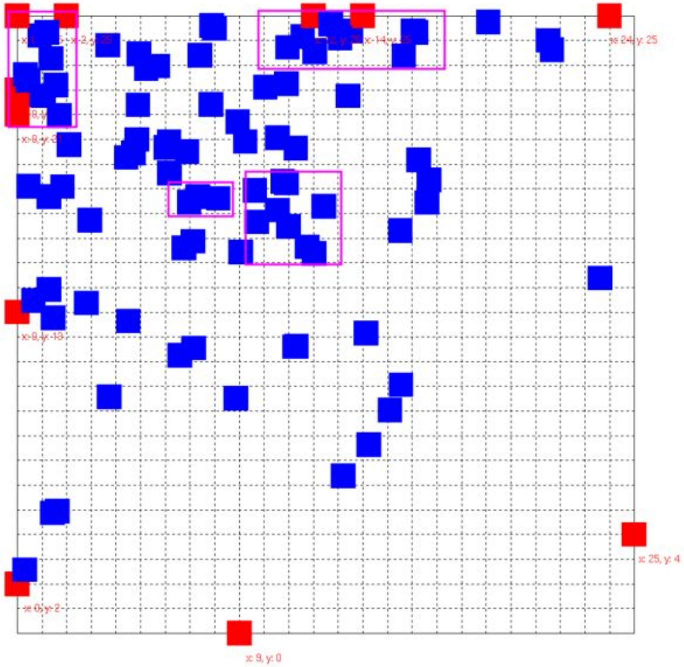


Figure 12: CC2: Strong weight between fixed and variable blocks

One Option: Keep blocks connecting fixed pad weights very high

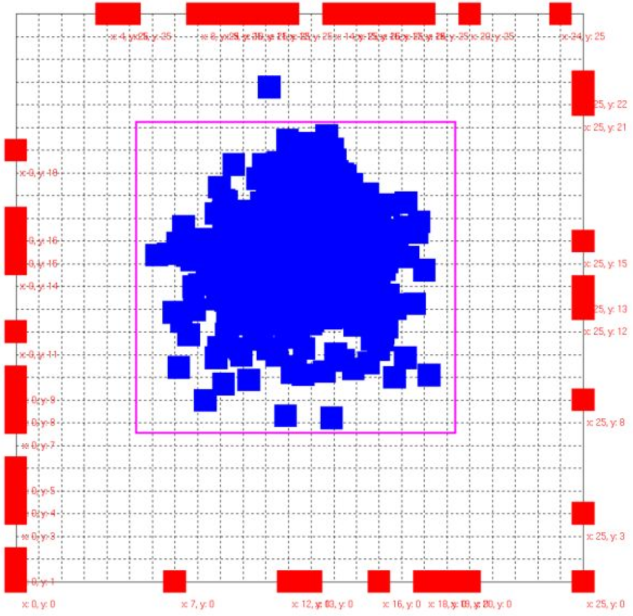


Figure 13: CC3: Weak weight between fixed and variable blocks

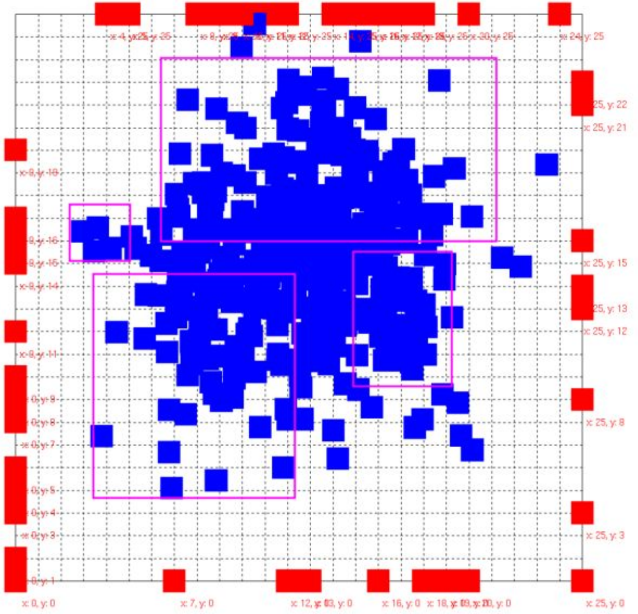
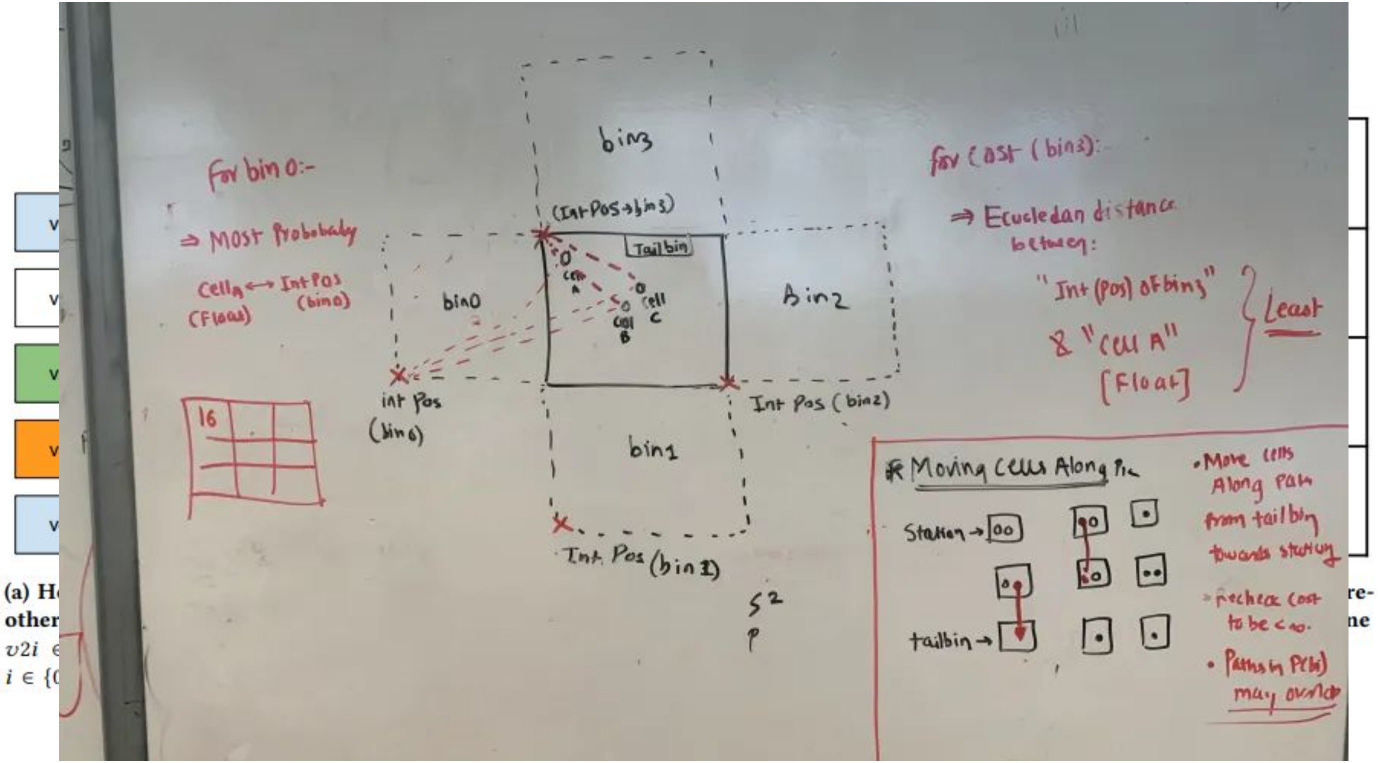


Figure 14: CC3: Strong weight between fixed and variable blocks

Second Option: Darav’s Spreading Algorithm



(a) H

other

$v2i$ ∈

$i \in \{($

re-

ne

Second Option: Darav's Spreading Algorithm

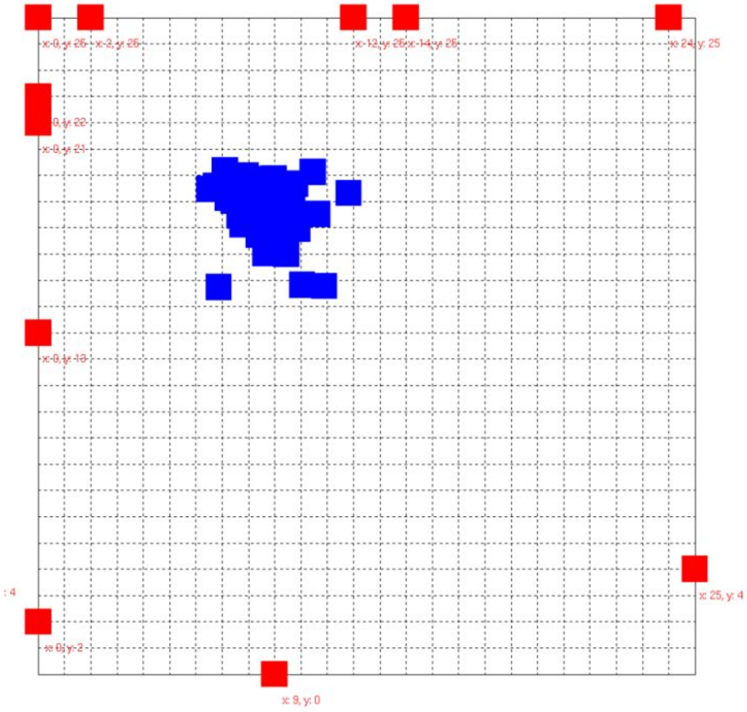


Figure 6: cc2 placement without ratsnet

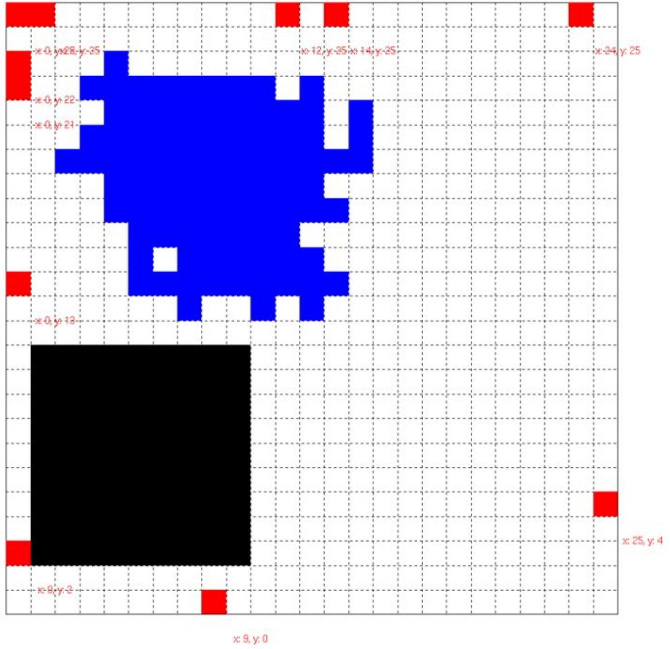


Figure 15: CC2: Spreading with Approach 1: Least constraint method

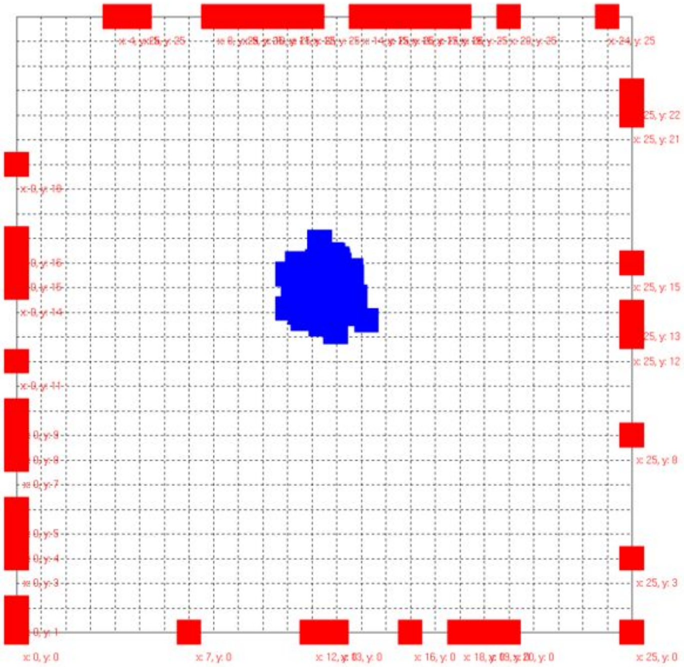


Figure 8: cc3 placement without ratsnet

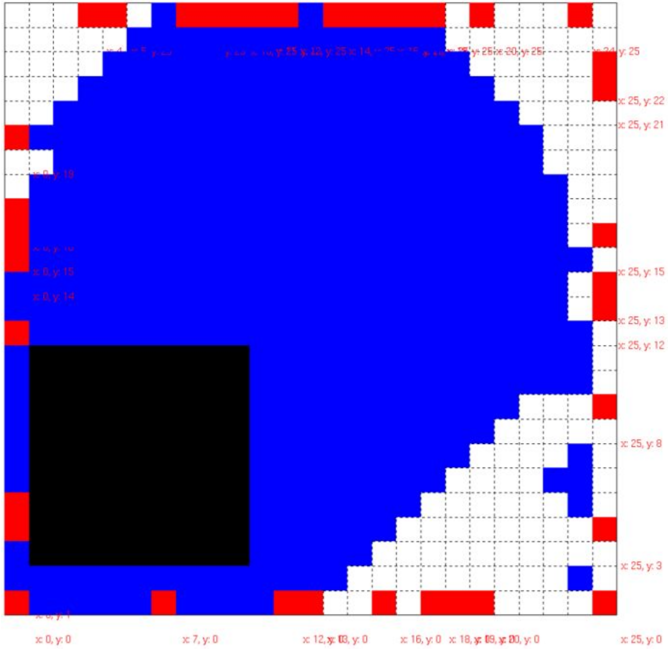
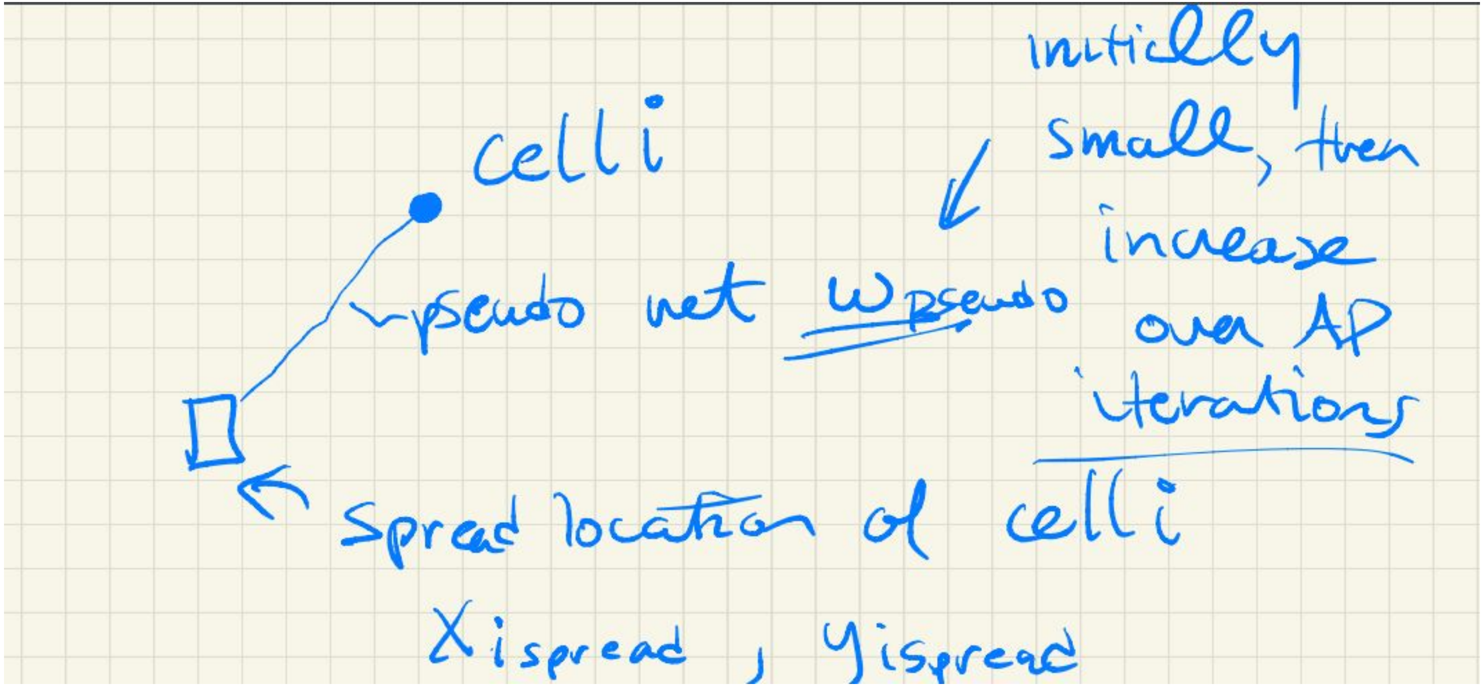


Figure 16: CC3: Spreading with Approach 1: Least constraint method

Still needs to actually place the cell at spreaded location!



Place -> Spread -> Place

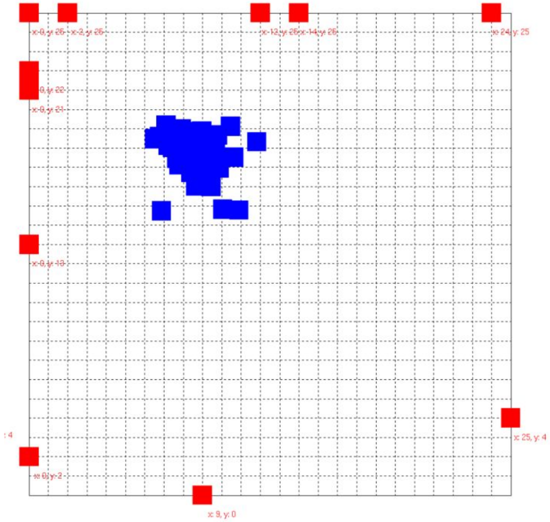


Figure 6: cc2 placement without ratsnet

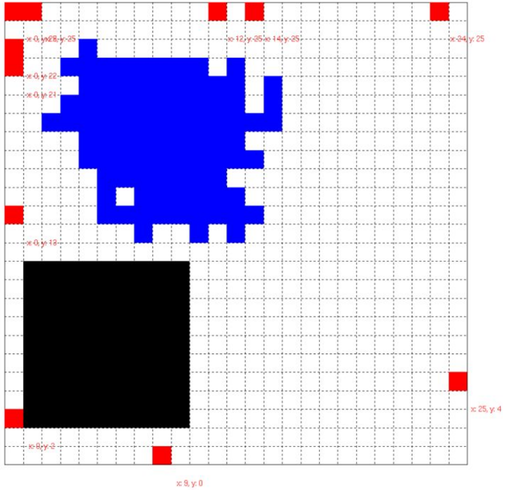


Figure 15: CC2: Spreading with Approach 1: Least constraint method

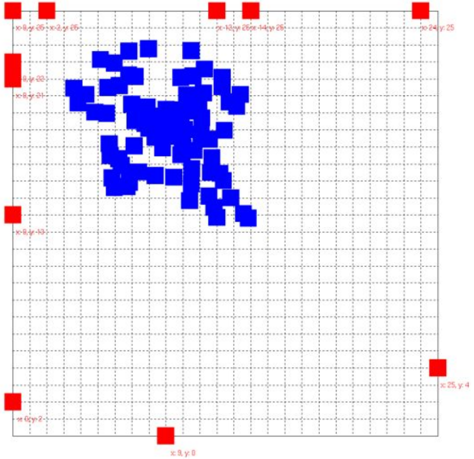


Figure 19: **CC2 Part 3B Strong**

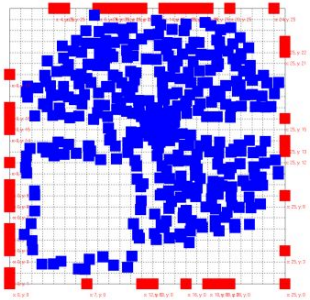


Figure 23: **CC3 Part 3B Strong**

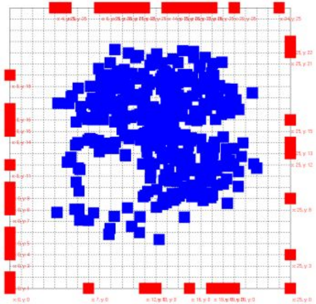


Figure 24: **CC3 Part 3B Weak**

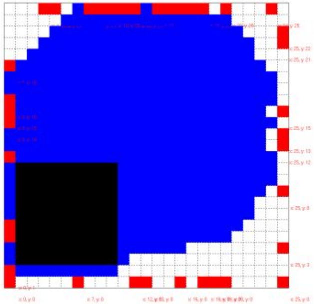


Figure 25: **CC3 Expected spread**

Figure 26: Comparison of CC2 Part 3B results

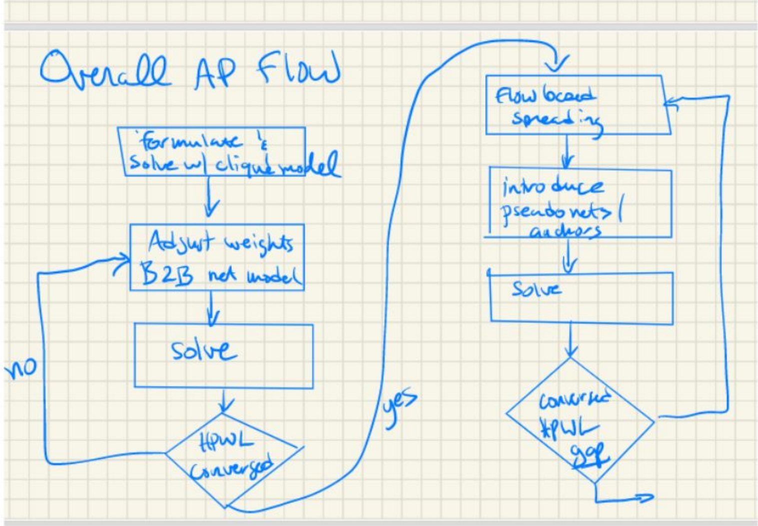
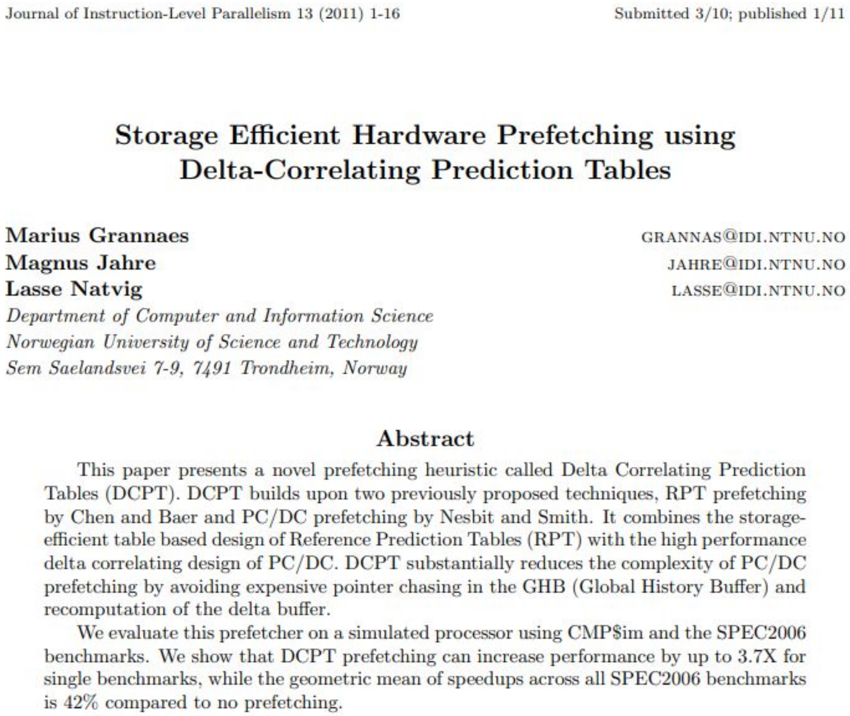


Figure 27: Analytical Placement Flow

ECE552: Computer Architecture



Journal of Instruction-Level Parallelism 13 (2011) 1-16 Submitted 3/10; published 1/11

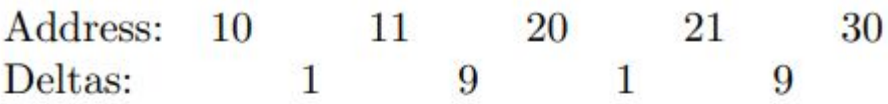
Storage Efficient Hardware Prefetching using Delta-Correlating Prediction Tables

Marius Grannaes GRANNAS@IDI.NTNU.NO
Magnus Jahre JAHRE@IDI.NTNU.NO
Lasse Natvig LASSE@IDI.NTNU.NO
Department of Computer and Information Science
Norwegian University of Science and Technology
Sem Saelandsvei 7-9, 7491 Trondheim, Norway

Abstract

This paper presents a novel prefetching heuristic called Delta Correlating Prediction Tables (DCPT). DCPT builds upon two previously proposed techniques, RPT prefetching by Chen and Baer and PC/DC prefetching by Nesbit and Smith. It combines the storage-efficient table based design of Reference Prediction Tables (RPT) with the high performance delta correlating design of PC/DC. DCPT substantially reduces the complexity of PC/DC prefetching by avoiding expensive pointer chasing in the GHB (Global History Buffer) and recomputation of the delta buffer.

We evaluate this prefetcher on a simulated processor using CMP$im and the SPEC2006 benchmarks. We show that DCPT prefetching can increase performance by up to 3.7X for single benchmarks, while the geometric mean of speedups across all SPEC2006 benchmarks is 42% compared to no prefetching.



Address: 10 11 20 21 30
Deltas: 1 9 1 9

Figure 3: Example delta stream.

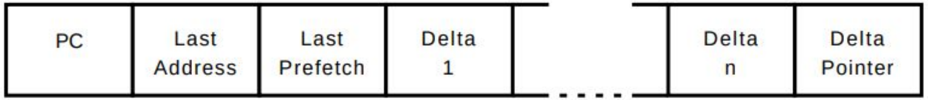


Figure 4: Format of a Delta Correlating Prediction Table Entry.

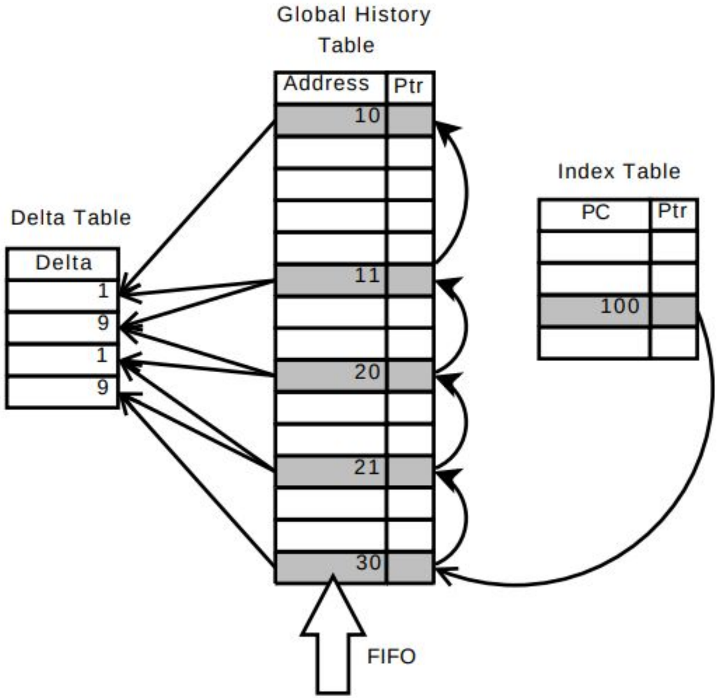


Figure 2: Example of a Global History Buffer.

THANK YOU