Technical Paper Proposal:
Placement Algorithms for Heterogenous FPGAs

Brian B Cheng
Department of Electrical and Computer Engineering

1 Proposal

In the software world, an engineer writes a program in a high-level langauge like C, Java, or Python. Then the engineer compiles the program using some compilation software like gcc or clang. First, the compiler takes the engineer's high-level program, and through multiple stages of lexing, parsing, and optimizations, generates a low-level program in an assembly language like x86 or MIPS. Then, the compiler assembles the assembly code into machine code. The produced machine code is often referred to as a binary and is ready to execute on the machine. Different compilers may have different strategies for this process. For example, LLVM and clang skip the assembly stage and go directly from high-level code to machine code, while gcc sticks to the traditional two-step process. The Java and Python are more complex in that they compile high-level code into an intermediate language as bytecode. In Java, the compiler assembles the bytecode into machine code at runtime in a process called Just-in-Time compilation which allows it to be platform independent. Python offers the choice to assemble the machine code before or at runtime. Simplify this downto 2 sentences.

In the FPGA world, the design entry to product pipeline is despairingly complex by comparison. First, the design entry. The engineer describes a digital system using a high-level HDL like Verilog or VHDL. Then the engineer submits the design entry to an EDA like Vivado or Quartus to perform three stages of automated design. These are: synthesis, placement, and routing. and then yap about NP-hardness and massive search space and hellish runtime.

2 Abbreviations

- **FPGA**: Field Programmable Gate Array
- **VLSI**: Very Large Scale Integration
- **EDA**: Electronic Design Automation
- **VHSIC**: Very High Speed Integrated Circuits

- **HDL**: Hardware Description Language
- **VHDL**: VHSIC HDL
- **HLS**: High Level Synthesis: Generating synthesizable HDL from high-level software languages. A company might want to have a software engineer write C or C++ code and have a program translate it into synthesizable Verilog. This can can boost productivity and save the company the need to hire a hardware engineer.
- **IP**: Intellectual Property: In FPGA context, this means pre-built modules or subsystems like a hardened microprocessor or Ethernet controller. These are usually proprietary.
- **SoC**: System on Chip: An FPGA device (chip) that features hardened IP in addition to the programmable logic fabric.
- **PL-PS**: Programmable Logic - Processing System: A design that utilizes the on-chip hard microprocessor in conjunction with the programmable logic fabric.
- **EDIF**: Electronic Design Interchange Format
- **HPWL**: Half Perimeter Wire Length

3 Keywords

- FPGA, EDA, Synthesis, Placement, Routing, Parallel, Optimization

4 Ideas

- **FPGA**: Field Programmable Gate Array
  - FPGA Vendors:
    - AMD-Xilinx (∼50% FPGA vendor market share)
    - Intel-Altera (∼35% share)
    - Lattice
    - Microsemi
- **EDA**: Electronic Design Automation
  - Proprietary software for FPGA and VLSI development:
    - Xilinx - Vivado (Design + Simulation) + Vitis (HLS + PL-PS codesign)
    - Altera - Quartus (Design) + ModelSim (Simulation)
    - Synopsis (VLSI)

    - Cadence (VLSI)
  - Open source software for FPGA development:
    - **VTR**: Simulated Annealing placer for FPGAs. Popular among researchers who study placement techniques. Commonly referred to as an "academic placer".
    - **OSS-CAD**: a full-flow software suite that includes ABC synthesis, Yosys synthesis, Yosys nextpnr.
    - **AMF-Placer**: Analytical Placer for FPGAs
    - **RapidWright**: Semi-open source API that provides backend access to Xilinx Vivado EDA using design checkpoints.
    - **RapidLayout**: Hard Block Placer for Systolic Arrays. Built with RapidWright.
    - **RapidStream**: HLS Placer. Built with RapidWright.
    - **DREAMPlace**: GPU-powered deep learning placement for VLSI.
    - **DREAMPlaceFPGA**: DREAMPlace, adapted to FPGAs via the RapidWright API.

- **Synthesis**
  - Takes a design written in a high-level HDL like VHDL or Verilog and "synthesizes" a **logical netlist** out of it.
  - The logical netlist is usually generated as an EDIF, JSON, or a low-level Verilog file.
  - The netlist describes the necessary basic elements of logic (BELs) and the wired connections between them that are necessary to implement the design.

- **Placement**
  - Takes the **logical netlist** and produces a **physical netlist**.
  - For each BEL in the netlist, assign the BEL to a Cell, Site, and Tile on the physical FPGA device.

- **Routing**
  - Takes the **physical netlist** and maps the connections between BELs onto wires, interconnects, and switchboxes on the FPGA.

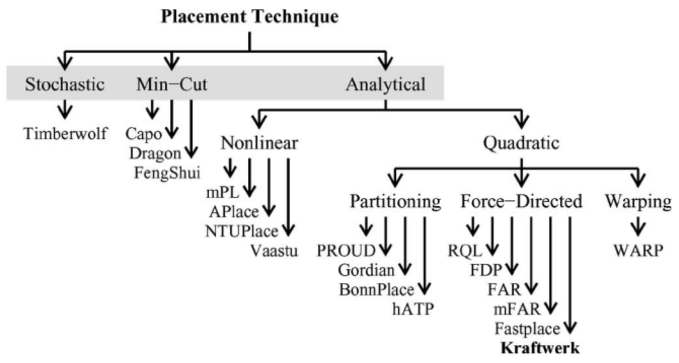

Figure 1: Landscape of VLSI placement techniques (Spindler) [1]

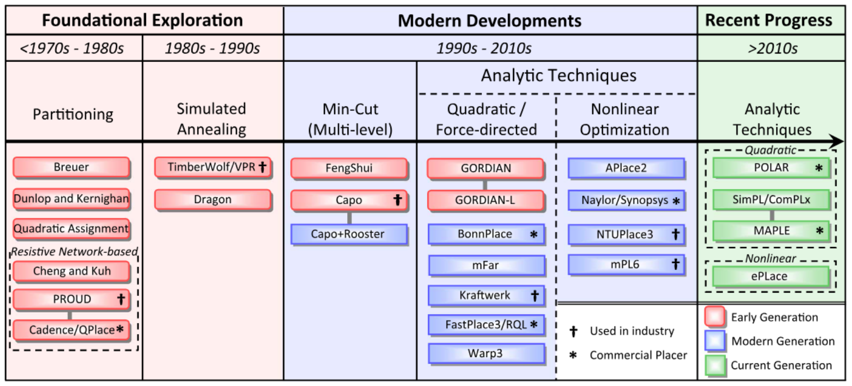

Figure 2: Historical timeline of VLSI placement techniques (Markov) [2]

This is a citation for AMFPlacer. [3]

References

[1] P. Spindler, U. Schlichtmann, and F. M. Johannes, “Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model,” *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 27, p. 1398–1411, aug 2008.

[2] I. L. Markov, J. Hu, and M.-C. Kim, “Progress and challenges in vlsi placement research,” *Proceedings of the IEEE*, vol. 103, no. 11, pp. 1985–2003, 2015.

[3] T. Liang, G. Chen, J. Zhao, S. Sinha, and W. Zhang, “Amf-placer: High-performance analytical mixed-size placer for fpga,” in *2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–6, 2021.

[4] A. Mishra, N. Rao, G. Gore, and X. Tang, “Architectural exploration of heterogeneous fpgas for performance enhancement of ml benchmarks,” in *2023 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 232–235, 2023.

[5] U. Farooq, H. Parvez, H. Mehrez, and Z. Marrakchi, “Exploration of heterogeneous fpga architectures,” *Int. J. Reconfig. Comput.*, vol. 2011, jan 2011.

[6] C. Lavin, M. Padilla, S. Ghosh, B. Nelson, B. Hutchings, and M. Wirthlin, “Using hard macros to reduce fpga compilation time,” in *2010 International Conference on Field Programmable Logic and Applications*, pp. 438–441, 2010.

[7] C. Lavin and A. Kaviani, “Rapidwright: Enabling custom crafted implementations for fpgas,” in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 133–140, 2018.

[8] Y. Zhou, P. Maidee, C. Lavin, A. Kaviani, and D. Stroobandt, “Rwroute: An open-source timing-driven router for commercial fpgas,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, nov 2021.

[9] C. Lavin and E. Hung, “Invited paper: Rapidwright: Unleashing the full power of fpga technology with domain-specific tooling,” in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–7, 2023.

[10] A. A. Kennings and I. L. Markov, “Analytical minimization of half-perimeter wirelength,” in *Proceedings of the 2000 Asia and South Pacific Design Automation Conference*, ASP-DAC ’00, (New York, NY, USA), p. 179–184, Association for Computing Machinery, 2000.

[11] S. Areibi, G. Grewal, D. Banerji, and P. Du, “Hierarchical fpga placement,” *Canadian Journal of Electrical and Computer Engineering*, vol. 32, no. 1, pp. 53–64, 2007.

[12] M. Gort and J. H. Anderson, “Analytical placement for heterogeneous fpgas,” in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pp. 143–150, 2012.

[13] J. Chen, W. Zhu, J. Yu, L. He, and Y.-W. Chang, “Analytical placement with 3d poisson’s equation and admm based optimization for large-scale 2.5d heterogeneous fpgas,” in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.

[14] S. Dhar, L. Singhal, M. Iyer, and D. Pan, “Fpga accelerated fpga placement,” in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 404–410, 2019.

[15] R. S. Rajarathnam, M. B. Alawieh, Z. Jiang, M. Iyer, and D. Z. Pan, “Dreamplacefpga: An open-source analytical placer for large scale heterogeneous fpgas using deep-learning toolkit,” in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 300–306, 2022.

[16] H. Long, Y. Bai, Y. Li, J. Wang, and J. Lai, “Optimizing wirelength and delay of fpga tile through floorplanning based on simulated annealing algorithm,” in *2023 IEEE 15th International Conference on ASIC (ASICON)*, pp. 1–4, 2023.

[17] T. Liang, G. Chen, J. Zhao, S. Sinha, and W. Zhang, “Amf-placer 2.0: Open source timing-driven analytical mixed-size placer for large-scale heterogeneous fpga,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2024.

[18] B. Ray, A. R. Tripathy, P. Samal, M. Das, and P. Mallik, “Half-perimeter wirelength model for vlsi analytical placement,” in *2014 International Conference on Information Technology*, pp. 287–292, 2014.

[19] B. N. B. Ray, S. K. Mohanty, D. Sethy, and R. B. Ray, “Hpwl formulation for analytical placement using gaussian error function,” in *2017 International Conference on Information Technology (ICIT)*, pp. 56–61, 2017.

[20] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, “Dreampiace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement,” in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.