Optimal
SWAP Gate Insertion
for Nearest Neighbor
Quantum Circuits

Robert Wille$^{1,2}$, Aaron Lye$^1$, Rolf Drechsler$^{1,2}$

$^1$ Institute of Computer Science
University of Bremen, Germany
$^2$ Cyber Physical Systems
DFKI GmbH Bremen, Germany

rwille@informatik.uni-bremen.de
Outline

- (Motivation and Background)
- Nearest Neighbor Constraints
- Ensuring Nearest Neighbor Constraints through SWAP Gates
- Minimizing the Number of SWAP Gates
- Experimental Evaluation
- Conclusions
Quantum Circuits

• Computation not only with 0 and 1 but also superposition of both
• Enables significant speed-ups for certain problems (e.g. factorization, database search)
Quantum Circuits

- Cascade of quantum gates
Quantum Gates

- Cascade of quantum gates
- Realize unitary operations $U$
- If $c=0$: all states remain unchanged
- If $c=1$: $U$ is applied to $t$

Control

$q_0$
$q_1$
$q_2$
$q_3$

$\text{target line } t$
Nearest Neighbor Constraints

- Motivated by physical realizations
- Control and Target Lines need to be adjacent
Ensuring Nearest Neighbor

- Through SWAP gates
Ensuring Nearest Neighbor

- Through SWAP gates
Ensuring Nearest Neighbor

How to minimize the number of SWAP gate insertions?
State-of-the-art

- Heuristic approaches
  - Re-ordering of circuit lines
  - Window-based schemes
  - Mapping the graph arrangement problem
  - ...
  (Van Meter & Oskin, 2006; Mottonen & Vartiainen, 2006; Chakrabarti & Sur-Kolay, 2007; Khan, 2008; Saeedi, Wille & Drechsler, 2010; Shafaei, Saeedi & Pedram, 2013)

- Exact approaches
  - Enumerative
  - Through gate order changes
General Idea
Consideration of
• all possible permutations before each gate
• the cost of implementing them (i.e. the number of SWAP gates)
  – Can be calculated using inversion vectors
  – Example: \((0; 1; 2; 3) \rightarrow (2; 3; 1; 0)\)
    • \(v = (3; 2; 0; 0)\)
    • \(3 + 2 + 0 + 0 = 5\) SWAP gates
Naïve Approach

1. Enumerately consider all possible permutations for all gates of the given circuit
2. For each permutation satisfying the nearest neighbor condition, calculate the number of SWAP gates required to realize the permutation
3. Afterwards, take the one with the smallest costs

\[ n!^d \text{ possible combinations} \]

- \( n \)\ldots number of lines, \( d \)\ldots number of gates
PBO Solver

- PBO solvers: An algorithm for solving the Pseudo Boolean Optimization problem
- Gets a Boolean function $f$ and an optimization function $F$ as input and determines
  - an assignment $a$ such that $f(a)=1$ and $F$ is minimized
  or
  - proofs that no such assignment exists
- Challenge:
  How to encode the PBO instance?
PBO Encoding

Consistency-constraints:
\[
\begin{align*}
x^{0}_{00} + x^{0}_{01} + x^{0}_{02} + x^{0}_{03} &= 1 \\
x^{0}_{10} + x^{0}_{11} + x^{0}_{12} + x^{0}_{13} &= 1 \\
x^{0}_{20} + x^{0}_{21} + x^{0}_{22} + x^{0}_{23} &= 1 \\
x^{0}_{30} + x^{0}_{31} + x^{0}_{32} + x^{0}_{33} &= 1 \\
x^{0}_{00} + x^{0}_{10} + x^{0}_{20} + x^{0}_{30} &= 1 \\
x^{0}_{01} + x^{0}_{11} + x^{0}_{21} + x^{0}_{31} &= 1 \\
&\ldots
\end{align*}
\]

Adjacency-constraints (for \( g_1 \) with \( q_0 \) and \( q_2 \)):
\[
\begin{align*}
(x^{1}_{00} \land x^{1}_{12}) \\
\lor (x^{1}_{10} \land x^{1}_{22}) \\
\lor (x^{1}_{20} \land x^{1}_{32}) \\
\lor (x^{1}_{02} \land x^{1}_{10}) \\
\lor (x^{1}_{12} \land x^{1}_{20}) \\
\lor (x^{1}_{22} \land x^{1}_{30})
\end{align*}
\]
PBO Encoding

Init. mapping to qubits

\[ l_0 \leftrightarrow q_0 \]
\[ x_0^0 = (x_{00}^0 x_{01}^0 x_{02}^0 x_{03}^0) \]

\[ l_1 \leftrightarrow q_1 \]
\[ x_1^0 = (x_{10}^0 x_{11}^0 x_{12}^0 x_{13}^0) \]

\[ l_2 \leftrightarrow q_2 \]
\[ x_2^0 = (x_{20}^0 x_{21}^0 x_{22}^0 x_{23}^0) \]

\[ l_3 \leftrightarrow q_3 \]
\[ x_3^0 = (x_{30}^0 x_{31}^0 x_{32}^0 x_{33}^0) \]

\[ \pi \]

\[ x_0^1 = (x_{00}^1 x_{01}^1 x_{02}^1 x_{03}^1) \]
\[ x_1^1 = (x_{10}^1 x_{11}^1 x_{12}^1 x_{13}^1) \]
\[ x_2^1 = (x_{20}^1 x_{21}^1 x_{22}^1 x_{23}^1) \]
\[ x_3^1 = (x_{30}^1 x_{31}^1 x_{32}^1 x_{33}^1) \]

Permutation-constraint (for \( \pi = (2310) \) and \( k = 1 \))

\[ (x_0^0 = x_2^1 \land x_1^0 = x_3^1 \land x_2^0 = x_1^1 \land x_3^0 = x_0^1) \Rightarrow s_{2310}^1 \]

Objective function:

\[
\min \left( (0 \cdot s_{0123}^2 + 1 \cdot s_{0132}^2 + 1 \cdot s_{0213}^2 + 2 \cdot s_{0231}^2 + 2 \cdot s_{0312}^2 + 3 \cdot s_{0321}^2 + 1 \cdot s_{1023}^2 + 2 \cdot s_{1032}^2 + 2 \cdot s_{1203}^2 + 3 \cdot s_{1230}^2 + 3 \cdot s_{1302}^2 + 4 \cdot s_{1320}^2 + 2 \cdot s_{2013}^2 + 3 \cdot s_{2031}^2 + 3 \cdot s_{2103}^2 + 4 \cdot s_{2130}^2 + 4 \cdot s_{2301}^2 + 5 \cdot s_{2310}^2 + 3 \cdot s_{3012}^2 + 4 \cdot s_{3021}^2 + 4 \cdot s_{3102}^2 + 5 \cdot s_{3120}^2 + 5 \cdot s_{3201}^2 + 6 \cdot s_{3210}^2 + \ldots ) \right)
\]
**Experimental Evaluation**

- Implemented on top of RevKit (www.revkit.org)
- *clasp* as PBO solver (www.cs.uni-potsdam.de/clasp/)
- Benchmarks from RevLib (www.revlib.org)

| Benchmark | $n$ | $|G|$ | $n!|G|$ | Swaps | Time |
|-----------|-----|------|---------|-------|------|

Experimental Evaluation

- Implemented on top of RevKit (www.revkit.org)
- *clasp* as PBO solver (www)
- Benchmarks from RevLib (www.revlib.org)
Conclusion

- Minimizing the number of SWAP gate insertions
- Exploiting the deductive power of PBO solvers
- Enabled to compare results obtained by heuristic methods to the actual optimum
- Future Work: Consideration of alternative architectures (e.g. nearest for 2D quantum architectures)
Optimal SWAP Gate Insertion for Nearest Neighbor Quantum Circuits

Robert Wille\textsuperscript{1,2}, Aaron Lye\textsuperscript{1}, Rolf Drechsler\textsuperscript{1,2}

\textsuperscript{1} Institute of Computer Science
University of Bremen, Germany
\textsuperscript{2} Cyber Physical Systems
DFKI GmbH Bremen, Germany

rwille@informatik.uni-bremen.de