Search-based Transformation Synthesis for 3-valued Reversible Circuits

D. Michael Miller* and Gerhard W. Dueck**

*University of Victoria
Victoria, BC Canada

**University of New Brunswick
Fredericton, NB, Canada

Reversible Computation 2020

Introduction

- The synthesis of r-valued reversible circuits is considered primarily for $r = 3$.
- The approach is based on transformation-based synthesis introduced for 2-valued reversible functions in 2003 and extended to the MVL case in 2004.
- Here we employ a bounded recursive search to explore possible circuits for a given reversible MVL function.
- Logical optimizations during synthesis and physical optimizations during mapping to quantum circuits are considered.

Background

- An n-input, n-output totally-specified r-valued function is ***reversible*** if it maps each input assignment to a unique output assignment.
- A ***3-valued*** $p \times p$ ***controlled unary reversible gate*** passes $p-1$ *control* lines through unchanged, and applies a specified unary operator to the p^{th} line, the *target* line, if the control lines assume *particular specified values*. Otherwise the target line is passed through unaltered.
- A gate must have a single target and a gate may have no controls in which case it is *uncontrolled*.
- A ***reversible circuit*** realizing an $n \times n$ reversible function is a cascade of reversible gates with no fanout or feedback. The circuit has n inputs and n outputs.

3-valued Unary Operators

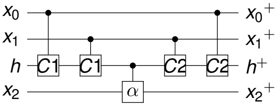
x	$C_1[x]$	$C_2[x]$	$N[x]$	$D[x]$	$E[x]$
0	1	2	2	0	1
1	2	0	1	2	0
2	0	1	0	1	2

- C_1 and C_2 are inverses of each other. D, E and N are each self-inverse.
- The following identities are used in circuit simplification.

$$\begin{aligned}
C_2[C_2[x]] &= C_1[x] \\
C_1[C_1[x]] &= C_2[x] \\
D[x] &= E[C_1[x]] = N[C_2[x]] \\
E[x] &= D[C_2[x]] = N[C_1[x]] \\
N[x] &= D[C_1[x]] = E[C_2[x]]
\end{aligned}$$

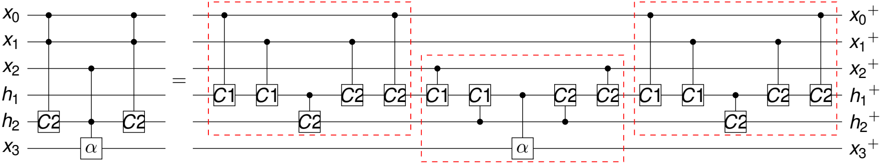
Quantum Gates and Circuits

- A ***Muthukrishnan and Stroud (MS) gate*** is a reversible gate as defined earlier with at most one control.
- It is clear from equations (1) to (5) that a single cycle gate, C_1 or C_2, and at least one of D, E or N, is sufficient as the other gates can be implemented by suitable gate pairings.
- In this work, we distinguish between the gates that are ***logically available*** during circuit synthesis and the gates that are ***physically available*** for the quantum circuit with the assumption that all physically available gates are available for use during the synthesis process.
- We also assume that both C_1 and C_2 are physically, and therefore logically, available as a cycle gate is implemented as a rotation the difference between C_1 and C_2 being the direction of rotation.



Implementation for $\alpha[x_2, x_1{=}2, x_0{=}2]$ with helper line h

$$\alpha = C_1, C_2, D, E, N$$



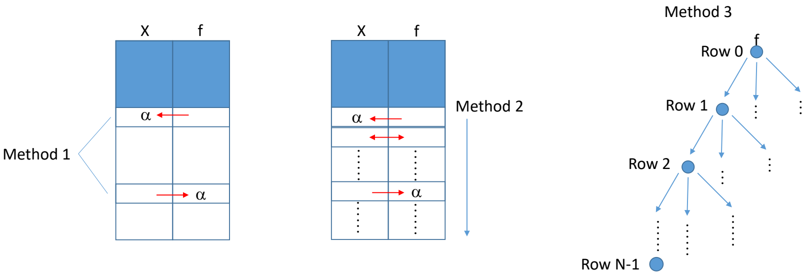
Implementation for $\alpha[x_3, x_2{=}2, x_1{=}2, x_0{=}2]$ with helper lines h_1 and h_2

$$\alpha = C_1, C_2, D, E, N$$

Quantum Cost Model

- A gate which applies $\alpha \in \{C_1, C_2, D, E, N\}$ with 0, 1, 2, 3 controls has a base cost of 1, 1, 5, 15, respectively.
- In general, the base cost for a k-control gate, $k > 2$, is $5 + 2\times$the cost of a gate with $k - 1$ controls.
- If a particular gate type is not physically available as a single MS gate a pair of gates is required and the cost increases by 1.
- The cost increases by 2, for each control that does not have the **global control value**.

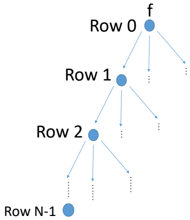
Transformation-based Synthesis



- Procedure **transform** determines a set of gates to map $a \to b$ where $a > b$ without affecting any $c < b$.
- The procedure is to choose a gate to map each $a_i \to b_i$ for $a_i \neq b_i$ starting from the least significant bit.
- For each gate, steps are taken to minimize the number of controls but making sure the gate does not apply for any $c < b$.
- For certain transitions a choice of gate type may be possible.

transition	options	transition	options
$0 \to 1$	C_1 E	$0 \to 2$	C_2 N
$1 \to 0$	C_2 E	$1 \to 2$	D
$2 \to 0$	C_1 N	$2 \to 1$	D

Method3: Bounded Search



- The search starts with f, an empty circuit and a infinite best circuit cost.
- Each node of the search tree is associated with a row of the specification and from that node there is a branch for each possible transition and choice of gates to make that row match.
- A path records a partial circuit until we reach a leaf.
- When a leaf is reached, the circuit on the path is compared to the best circuit to date.
- The search is pruned by aborting a path if the partial circuit is more expensive than the best circuit found so far.

Post-synthesis Simplification

- Two gates are **inverses** of each other if they have the same target, the same control variables and control values and either they are both D, E or N gates, or one is a C_1 gate and the other is a C_2 gate.
- Two C_k gates are **mergeable** if they have the same target, the same control variables and control values. The merge into a single gate has the given target, control variables and control values and is of type C_{3-k}.
- Two gates G_i and G_j are **control reducible** if they are of the same type, have the same target and controls and matching control values except for one control x_k. The gates can be modified by removing x_k from G_i and setting the control value for x_k for G_j to $3 - s$ where s is the sum of the original x_k control values for the two gates. If the gates are C gates, G_j is replaced by its inverse.

Moving Rule, Reduce and Insert_C

- Two adjacent gates G_i and G_{i+1} can be **interchanged** unless:
  1. The two gates have the same target but the gates are not both of the same type (C, D, E or N);
  2. If G_i has type $t \in \{C, D, E, N\}$, the target of G_i is a control for G_{i+1} with control value v and $t{=}C$, or $t{=}D, E, N$ and $v \neq 0, 2, 1$ respectively; or
  3. If G_{i+1} has type $t \in \{C, D, E, N\}$, the target of G_{i+1} is a control for G_i with control value v and $t{=}C$ or $t{=}D, E, N$ and $v \neq 0, 2, 1$ respectively.
- Procedure **reduce** simplifies a circuit by looking for inverse, mergeable and control reducible gate pairs using the moving rule to determine when such gates can be made adjacent in the circuit.
- Procedure **insert_C** scans the circuit from inputs to outputs moving or adding uncontrolled C gates to map all controls to the desired value.

Circuit simplification Strategy

1. apply **reduce** to the circuit produced by the chosen synthesis method;
2. if the target is a quantum circuit, apply **insert_C** to add the required uncontrolled *C* gates to map all gate control values to the desired value;
3. perform any logical gate substitutions for *D*, *E* or *N* gates depending on which types of gate substitution have been specified for the current synthesis.
4. if step 2 and/or step3 has changed the circuit, apply **reduce** a second time to identify any further reductions .

Experimental Results

2-variable 3-valued functions reversible circuit average gate counts: with N and D but no E gates

	No Gate Reduction					
		f			f and f^{-1}	
Method	Avg. Gates	CPU Sec.	Avg. Circ. per Func.	Avg. Gates	CPU Sec.	Avg. Circ. per Func.
1	7.160	2.51		6.957	5.31	
2	7.078	12.67		6.860	25.28	
3	6.125	86.42	21.727	6.083	171.63	43.450
impr. 3 vs 1	14.46%			12.56%		

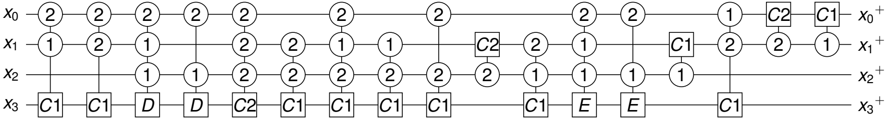
	Gate Reduction					
		f			f and f^{-1}	
Method	Avg. Gates	CPU Sec.	Avg. Circ. per Func.	Avg. Gates	CPU Sec.	Avg. Circ. per Func.
1	7.077	2.67		6.855	5.51	
2	6.989	12.73		6.753	25.65	
3	5.983	103.45	30.030	5.919	209.25	60.070
impr. 3 vs 1	15.46%			13.65%		

2 variable 3-valued functions: average quantum cost

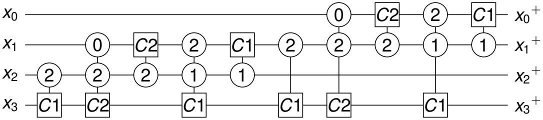
Synth.	Sub. *D*	Sub. *E*	Sub. *N*	METHOD1 CV=2	METHOD1 CV=1	METHOD2 CV=2	METHOD2 CV=1	METHOD3 CV=2	METHOD3 CV=1
D N				9.950	11.667	9.896	11.279	**7.837**	8.488
D E N				**9.701**	**10.884**	**9.623**	**10.553**	7.963	**8.048**
D				10.193	11.995	10.143	11.617	8.057	8.684
D E				10.001	11.301	9.917	10.935	8.163	*8.154*
D N			1	10.416	12.224	10.355	11.823	8.275	8.934
D E N		1		10.386	11.713	10.307	11.404	8.327	8.730
D E N			1	10.244	11.409	10.157	11.063	8.477	*8.345*
D N			2	10.614	12.328	10.546	11.923	8.486	9.082
D E N		2		10.543	11.772	10.464	11.459	8.502	8.800
D E N			2	10.338	11.513	10.245	11.164	8.545	8.507
D E		1		10.653	11.988	10.591	11.674	8.566	8.786
D E		2		10.762	12.109	10.697	11.783	8.722	8.978
D E N	1			11.489	12.215	11.345	11.876	8.799	*8.670*
D E N		1	1	10.898	12.212	10.813	11.892	8.859	9.085
D E N	2			11.627	12.769	11.479	12.346	8.879	*8.809*
D N	1			12.208	13.355	12.086	12.975	8.887	9.237
D E N		2	1	11.003	12.326	10.916	11.996	8.977	9.251
D E N		1	2	11.075	12.289	10.982	11.968	9.045	9.185
D N	2			12.484	14.180	12.37	13.723	9.130	9.643
D E N		2	2	11.181	12.403	11.086	12.072	9.175	9.357
D E	1			12.077	13.132	11.892	12.701	9.197	*8.979*
D E N	1	1		12.211	13.072	12.071	12.755	9.285	9.475
D E N	2	1		12.350	13.626	12.204	13.227	9.386	9.640
D E	2			12.316	13.575	12.13	13.083	9.406	*9.189*
D E N	1		1	12.040	12.992	11.886	12.598	9.443	*9.135*
D E N	1	2		12.365	13.129	12.225	12.807	9.478	9.548
D E N	1		2	12.133	13.093	11.973	12.697	9.518	*9.305*
D E N	2		1	12.179	13.304	12.023	12.867	9.519	*9.219*
D E N	2	2		12.507	13.685	12.362	13.282	9.586	9.718
D E N	2		2	12.274	13.408	12.112	12.968	9.597	*9.393*

Full Adder

$$x_0^+ = sum[x_0, x_1, x_2) \;\; x_1^+ = x_1 \oplus x_2 \;\; x_2^+ = x_2 \;\; x_3^+ = carry[x_0, x_1, x_2) \oplus x_3$$

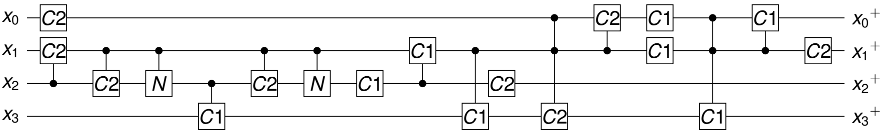


METHOD1 / METHOD2 forward synthesis with simplification but no gate substitution; D and E gates allowed: 17 gates



METHOD3 forward synthesis (same conditions): 10 gates

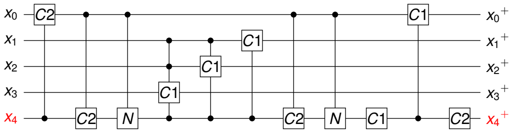
Method	No E Gates			E Gates		
	Gates	Cost	CPU	Gates	Cost	CPU
Forward Synthesis						
1	30	106	0.047	37	157	0.047
2	30	106	0.062	37	157	0.078
3	18	26	0.438	18	26	0.703
impr. 3 vs. 1		75.5%			83.4%	
Reverse Synthesis						
1	44	180	0.063	59	289	0.078
2	44	180	0.094	59	289	0.125
3	18	34	379.8	18	34	593.7
impr. 3 vs. 1		81.1%			88.2%	



METHOD3 forward synthesis with full simplification including INSERT_C: 18 gate full adder quantum circuit, cost 26, $cv = 2$

Controlled Counter

	No E Gates			E Gates		
Method	Gates	Cost	CPU	Gates	Cost	CPU
Forward Synthesis						
1	15	137	0.03	15	137	0.05
2	15	137	0.11	15	137	0.13
3	11	29	182.17	11	29	290.64
impr. 3 vs. 1		78.8%			78.8%	
Reverse Synthesis						
1	17	137	0.03	17	137	0.03
2	17	137	0.11	17	137	0.13
3	11	29	210.96	11	29	272.52
impr. 3 vs. 1		78.8%			78.8%	



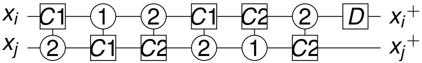
METHOD3 Forward synthesis: 11 gate quantum counter circuit using full simplification, cost 29, $cv = 2$

Controlled Rotation

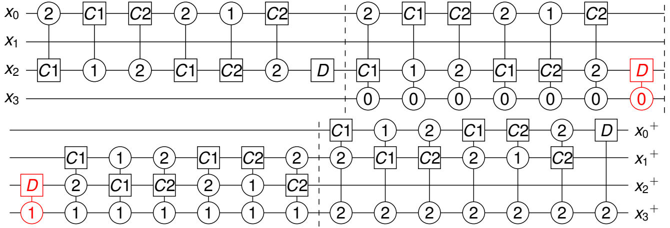
- A reversible function with four inputs and output
  - x_3, x_2, x_1, x_0 if $x_3{=}0$,
  - x_3, x_1, x_0, x_2 if $x_3{=}1$,
  - x_3, x_0, x_2, x_1 if $x_3{=}2$.
- METHOD1 or METHOD2 using N, D and E gates with D and E gate logical substitution and simplification yields a reversible circuit with 76 gates and a quantum cost of 358 (75 and 358 for inverse function).
- METHOD3:
  1. A check is inserted to test if $F_k{=}k$ and if it does to accept that 0 gate case without exploring all other alternatives.
  2. The search is aborted if a preset circuit cost is reached.
- METHOD3 with a cost limit of 170 to the inverse of the rotation function a circuit was found with 50 gates and quantum cost 170 – a bit less than half the cost found using METHOD1. The search took 3.8 CPU hours and considered 1,551 circuits.

$\color{blue}{swap[x_0, x_2, x_3{=}1]}$ $swap[x_1, x_2, x_3{=}1]$ $\color{blue}{swap[x_0, x_2, x_3{=}2]}$ $swap[x_0, x_1, x_3{=}2]$

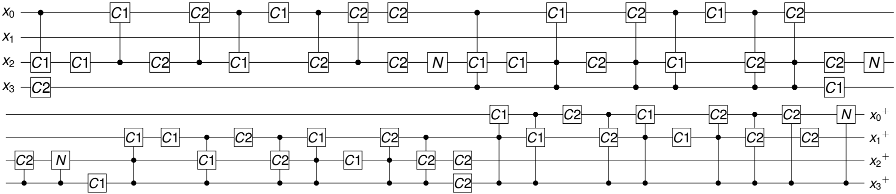
$\color{blue}{swap[x_0, x_2]}$ $\color{blue}{swap[x_0, x_2, x_3{=}0]}$ $swap[x_1, x_2, x_3{=}1]$ $swap[x_0, x_1, x_3{=}2]$



METHOD3 no simplification: uncontrolled swap of x_i and x_j



reversible rotation circuit by swap circuit substitution



50 gate quantum rotation circuit found by simplification: quantum cost 122, $cv = 2$

Conclusion and Future Work

- A new bounded search transformation-based synthesis approach was presented showing its potential and limitations.
- Not yet clear why the search takes so much longer for the rotation circuit compared to arithmetic circuits like to adder and counter.
- Coupling the search method with decomposition techniques should be explored.
- Optimization of the circuits at the final quantum level should be examined.
- It would be interesting to consider how the search method applies in the Boolean case.

Thank you!

Questions / Comments?