Parallel Gröbner Basis Rewriting and Memory Optimization for Efficient Multiplier Verification

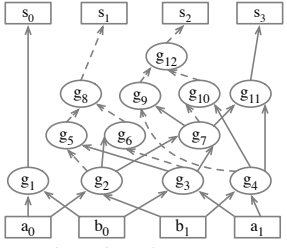
Hongduo Liu[1], Peiyu Liao[1], Junhua Huang[2], Hui-Ling Zhen[2], Mingxuan Yuan[2], Tsung-Yi Ho[1], Bei Yu[1]

[1]The Chinese University of Hong Kong
[2]Huawei Noah's Ark Lab

March 27, 2024

Background

- **Integer multipliers** have wide applications in signal processing, cryptography, scientific computing, etc.
- **Formal Verification** is essential to ensure reliability.
- **Symbolic Computer Algebra** (SCA) based methods have achieved SOTA performance compared with BDD and SAT.

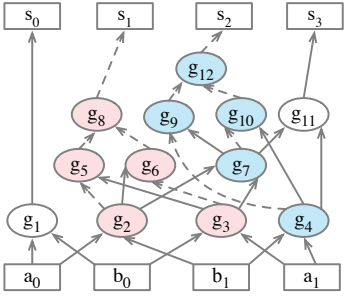


$sp := 8s_3 + 4s_2 + 2s_1 + s_0 -$
$(2a_1 + a_0)(2b_1 + b_0)$
$f_{s_3} := -s_3 + g_{11}$
$f_{g_{11}} := -g_{11} + g_4 g_7$
$f_{s_2} := -s_2 + 1 - g_{12}$
$f_{g_{12}} := -g_{12} + 1 - g_9 - g_{10} + g_9 g_{10}$
...

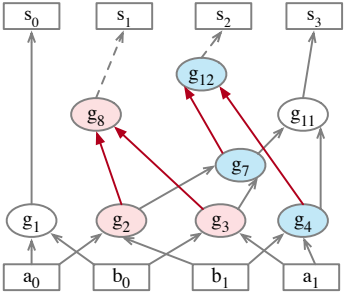
- Model the circuit as Gröbner basis polynomials $G = \{g_1, ..., g_s\}$ and the specification as a polynomial sp.
- Rewrite the Gröbner basis G to a new Gröbner basis G_n that has fewer variables. [Contribution 1: parallel rewriting]
- Reduce (divide) sp wrt. polynomials in G_n. [Contribution 2: double buffering and operator scheduling]
- Check the remainder.

Parallel Rewriting

- We observe that the elimination of certain variables can operate independently of others.
- The elimination of (g_5, g_6) and (g_9, g_{10}) in the example AIG are independent of each other and thus can be done in parallel.



$$f_{g_8} := -g_8 + 1 - g_5 - g_6 + g_5 g_6$$
$$f_{g_5} := -g_5 + g_3 - g_2 g_3$$
$$f_{g_6} := -g_6 + g_2 - g_2 g_3$$
$$\dots$$



$$f_{g_8} := -g_8 - g_2 - g_3 + 2g_2 g_3$$
$$f_{g_{12}} := -g_{12} - g_4 - g_7 + 2g_4 g_7$$
$$\dots$$

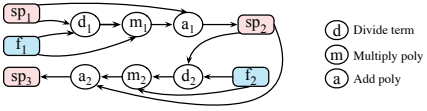
Procedure of Specification Polynomial Reduction

Suppose we desire to reduce $sp = c_1x_1x_2x_3 + \cdots + c_2x_2x_4$ through a Gröbner basis polynomial $f_{x_2} = -x_2 + c_3x_5x_6$. The reduction process can be accomplished by the following steps:

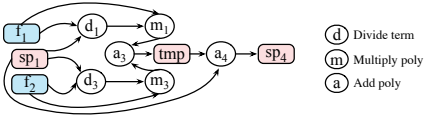
- Divide term: identify all terms containing the variable x_2 in sp, divide x_2 from those terms and add them together to get $quo := c_1x_1x_3 + c_2x_4$.
- Multiply poly: multiply the obtained quo with f_{x_2}, resulting in a polynomial $mul := -c_1x_1x_2x_3 + c_1c_3x_1x_3x_5x_6 - c_2x_2x_4 + c_2c_3x_4x_5x_6$.
- Add poly: add mul to sp to cancel all terms containing the variable x_2, which are $c_1x_1x_2x_3$ and $c_2x_2x_4$.

Double Buffering and Operator Scheduling

- Double buffering: Store sp_1 in the first buffer and sp_2 in the second buffer. The first buffer can be rewritten by sp_3.
- Operator scheduling: Suppose we have $f_1 := -\alpha + h(T_\alpha)$ and $f_2 := -\beta + h(T_\beta)$. $h(T_\alpha)$ and $h(T_\beta)$ are both polynomials. If $\alpha \notin h(T_\beta)$, $\beta \notin h(T_\alpha)$ and $\forall u \in sp_1, u \xrightarrow{\alpha\beta} r \neq 0$, where u is a monomial in sp_1, then the reduction of f_1 and f_2 can be performed concurrently.



(a)



(b)

(a) Original computation graph. (b) Computation graph after rescheduling.

Experimental Results

Table: Verification runtime comparison on multipliers generated by GenMul[1].

benchmark	size	#gates	Amulet 2.2[2]			Ours (16 threads)		
			rewriting	reduction	overall	rewriting	reduction	overall
SP-AR-LF	128×128	194314	0.98	0.16	1.30	0.43	0.57	1.15
SP-DT-LF		193806	2.26	0.38	2.82	0.45	0.82	1.39
SP-WT-BK		197774	2.31	2.65	5.24	0.48	1.26	1.92
SP-AR-LF	256×512	781834	6.20	0.87	7.72	2.37	1.52	4.55
SP-DT-LF		780814	17.85	2.09	20.80	1.79	3.57	6.06
SP-WT-BK		790610	17.84	25.85	44.66	1.86	5.63	8.16
SP-AR-LF	512×512	3136522	55.42	5.70	63.81	10.94	6.97	20.13
SP-DT-LF		3134478	185.53	12.02	201.13	8.28	15.22	26.33
SP-WT-BK		3157890	186.46	322.87	512.96	8.84	33.05	45.02
SP-AR-LF	1024×1024	12564490	506.55	39.39	573.05	54.26	37.41	102.11
SP-DT-LF		12560398	1817.96	92.74	1940.23	38.32	73.96	123.68
SP-WT-BK		12606714	1807.25	3519.13	5356.14	37.65	311.19	360.71
Average Ratio			15.64	2.80	6.24	1.00	1.00	1.00

[1] Alireza Mahzoon, Daniel Große, and Rolf Drechsler (2021). "GenMul: Generating architecturally complex multipliers to challenge formal verification tools". In: *Recent Findings in Boolean Techniques*. Springer, pp. 177–191.

[2] Daniela Kaufmann and Armin Biere (2022). "Fuzzing and Delta Debugging And-Inverter Graph Verification Tools". In: *International Conference on Tests and Proofs*. Springer, pp. 69–88.