



HAL
open science

Multiple Constant Multiplication: From Target Constants to Optimized Pipelined Adder Graphs

Rémi Garcia, Anastasia Volkova

► **To cite this version:**

Rémi Garcia, Anastasia Volkova. Multiple Constant Multiplication: From Target Constants to Optimized Pipelined Adder Graphs. 2023 33rd International Conference on Field-Programmable Logic and Applications (FPL), Sep 2023, Gothenburg, France. pp.137-143, 10.1109/FPL60245.2023.00027 . hal-03943273v2

HAL Id: hal-03943273

<https://hal.science/hal-03943273v2>

Submitted on 4 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multiple Constant Multiplication: From Target Constants to Optimized Pipelined Adder Graphs

Rémi Garcia[✉] and Anastasia Volkova[✉]
Nantes Université, École Centrale Nantes,
CNRS, LS2N, UMR 6004
F-44000 Nantes, France
surname.lastname@univ-nantes.fr

Abstract—Multiple Constant Multiplication (MCM) is a ubiquitous problem for numerous computation-intensive applications. One efficient approach is to replace generic multipliers by multiplierless architectures based on bit-shifts and additions. The adder graphs describing the multiplierless circuits can be optimized according to various metrics. In this paper, we optimize for throughput and improve the state-of-the-art for the design of pipelined adder graphs. In contrast to existing approaches, which pipeline *a posteriori* or use heuristics, our solution is to optimally solve the pipelined adder graph design problem using Mixed-Integer Linear Programming (MILP). We first model the pipelining and its cost, and then incorporate it into the state-of-the-art ILP model for the MCM design. Fusing the MCM design with pipelining into a single global optimization problem exactly solved by powerful MILP solvers demonstrated clear benefits on numerous benchmarks. Moreover, mathematical modeling allows for an easily extendable tool which can adapt to evolving hardware models/metrics.

Index Terms—Adder graph, optimal design, multiplierless hardware, integer linear programming

I. INTRODUCTION

Many numerical algorithms and applications involve multiplications by integer constants. Fixed-Point (Fxp) numbers, which can be assimilated to integers, are the preferred choice for algorithms targeting embedded systems such as Field-Programmable Gate Arrays (FPGAs). In particular, the evaluation of FIR filters [1, Chapter 6.4], Fast Fourier Transform [2] or parallel neural network controllers [3] require to multiply integer variables with tens to thousands of constants.

While generic multipliers perfectly handle these multiplications by constants, their cost largely exceeds multiplierless architectures that benefit from the knowledge of constants' values [4]. The shift-and-add approach is hence the privileged method to reduce hardware cost. It consists in replacing multiplications by bit-shifts, additions and subtractions where bit-shifts are multiplications by a power of two, that can be hardwired for a negligible cost. For example, multiplying an integer variable x by the constant 7 can be rewritten as $7x = 2^3x - x$, reducing the cost to a single bit-shift by three positions to the left and a subtraction, instead of a multiplication.

The Multiple Constant Multiplication (MCM) problem consists in finding the implementation with the lowest cost that achieves the multiplication by a given set of target constants to multiply with. The ultimate goal would be to search for the lowest cost by considering the final hardware metrics. Yet, all the information required to compute the final hardware cost is

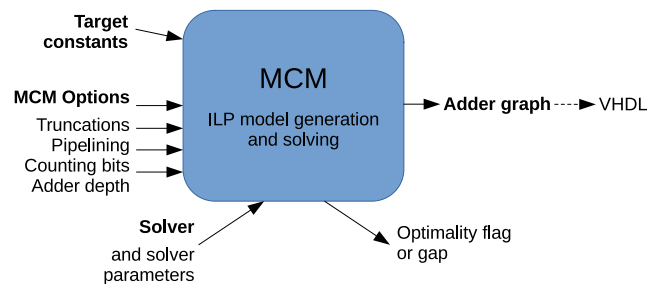


Fig. 1: Our tools takes target constants and multiple options to optimize for different MCM problem flavors

not disclosed and the typical way is to use proxy metrics, such as the number of adders, to estimate the synthesized hardware cost [5]–[10].

The hardware cost is actually manifold and while most higher-level metrics try to target those as a whole, in this work we put a specific focus on delay and throughput by considering adder pipelining. A direct solution is to simply pipeline the optimal solutions to the MCM problem [11], but this is not as efficient as adding registers into the design approach and searching directly for the best multiplierless topologies minimizing the number of registers [8]. With this work, we propose a new Integer Linear Programming (ILP) based model that solves the Pipelined MCM problem (PMCM). This new model is incorporated into the \mathcal{J}^{MCM} tool providing solutions to different flavors of the MCM problem. In particular, we explore, for the first time, pipelining in the MCM targeting the low-level hardware metric that counts the number of one-bit adders.

The easy-to-use and open-source tool in Fig. 1 generates MCM solutions as adder graphs, and can be used to automatically generate the VHDL code. We demonstrate, on a large set of benchmarks, that our optimal approach generates solutions that demand 17.57% fewer adders and registers, on average, than the state-of-the-art heuristic RPAG [8].

II. STATE OF THE ART

The MCM problem has been studied for decades and first solutions [12], [13] were based on the Canonical Signed Digit (CSD) representation. This consists in rewriting of the

number using a signed digit representation using $\{-1, 0, 1\}$ in a Non-Adjacent Form (NAF), *i.e.*, nonzero bits are not adjacent. For example, $23x = 0010\bar{1}00\bar{1}_{\text{CSD}} \times x$ replaces the multiplication by 23 by two additions, as illustrated by an adder graph in Fig. 2a. In an adder graph, numbers inside nodes represented with rectangles are called *fundamentals* and denote the constants by which the input is multiplied with, and labels on edges denote either the negation, or an integer power of two the signal is multiplied with. However, CSD does not always provide optimal results and does not give any guarantee on optimality. Moreover, since CSD is applied on each constant separately, resource sharing should be explored by additional algorithms.

Many heuristics enhance the results obtained with the CSD method for the MCM problem [5], [14]–[17], and more recently optimal approaches have been developed [6], [7], [9], [10], [18].

The first optimal methods were based on dedicated algorithms relying on hypergraphs [6] or branch-and-bound [7]. Both methods were not easily extensible and an ILP based model [9] which could solve either the MCM or PMCM problems has been proposed to benefit from the versatility of mathematical modeling. A simpler model [10], which did not rely on heavy precomputations, was used more recently and has been adapted [19] to find solutions for the single constant multiplication using SAT/SMT solvers.

All these methods have in common a focus on the number of adders, solving what we call the MCM-Adders problem. Finer-grain metrics, such as the number of one-bit adders, the adder depth or the glitch path count, have first been tackled heuristically [5], [8]. The recent ILP-based optimal approach [18] proposes a new MCM model that efficiently encodes the adder graph topology and various metrics as ILP constraints, such that no precomputations are required and the model is easily extensible for different optimization criteria. In particular, [18] proposes a new model that minimizes the number of one-bit adders, *i.e.* solving the, as we shall call it, MCM-Bits problem.

In order to obtain a higher throughput, designers often consider pipelined adder graphs, *i.e.*, adder graphs with registers before each stage [8], [9], [11], [20]. In particular, we assume that all outputs should be available at the same clock cycle. Pipelining of a fixed adder graph, for example coming from the MCM-Adders or MCM-Bits, can be obtained by performing cut-set retiming (CSR) [21]. For example, to pipeline the stages of the optimal adder graph for $23x$, shown in Fig. 2a, one must put a register at the input, one register after each adder and one register for the fundamental 1 on Stage 1, as shown in Fig. 2b. When implementing on FPGAs, we will consider that registers that follow adders basically come for free. It is worth to mention that pipelining the individual adders instead of the *adder stage* is possible and efficient for ASICs design [22], but we leave this option out of the scope of our paper.

As it was shown in [11], adapting the adder graph topology specifically for *pipelined* implementation yields designs with a lower cost. The RPAG tool [8] provides heuristic solutions to the problem and [9] gives first optimal ILP-based approach

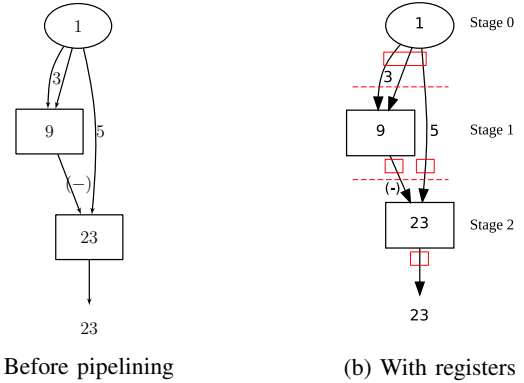


Fig. 2: Pipelining adder stages

for PMCM. The idea of Kumm [9] is to solve a satisfaction problem, given a bound on the number of adders and on the adder depth, in which a full enumeration of the design space is performed to construct the ILP problem. Pipelining is then ensured by a constraint that the adder’s inputs on stage n should come exclusively from the preceding stage $n - 1$, and not from earlier stages. The drawback of this approach is a strong limitation in the number of possible stages, due to heavy precomputations. Moreover, the ILP models by Kumm cannot be easily combined/updated with other metrics and constraints.

Instead, in the following we will use the versatile framework proposed in [18] and extend it to solve the PMCM problem. In the following we present a few sets of variables and constraints which permit to take the register cost into account.

III. PIPELINED MULTIPLE CONSTANT MULTIPLICATION

In [18], two main models are used: MCM-Adders, optimizing for the adder count, and the MCM-Bits optimizing the number of one-bit adders. In this section, we add pipelining as a constraint to both of them.

Unique adders. Let us first make the following observation. The solution of the MCM-Adders problem is guided by the number of adders, and naturally, each fundamental (the constant associated with the adder) should be unique in the adder graph. However, when incorporating the pipelining, it is the length of the connection between an adder and the maximum stage it interacts with, which is guiding the topology. Hence, it might sometimes be beneficial to compute the same fundamental twice, as it saves registers. For example, consider the adder graph in Fig. 3a, computing $5x$, $69x$, $553x$ and $2483x$. To pipeline this adder graph, we need 8 registers in addition to the 5 adders, as illustrated in Fig. 3b. However, there exists an adder graph, represented in Fig. 4, which only requires 6 registers and 6 adders but computes the same fundamental (5) twice: on Stage 1, and then on Stage 4. This permits to avoid propagating the fundamental 5 through all the stages and to recompute it when needed, which reduced the overall cost.

With this observation we first modified the base model for MCM-Adders and MCM-Bits to allow for multiple occurrences of the same fundamental, if they are computed on different stages.

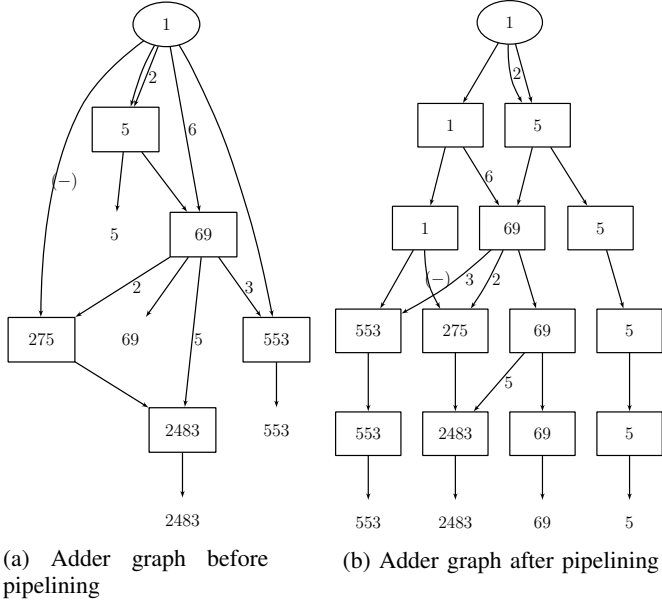


Fig. 3: Adding register to pipeline a fixed adder graph

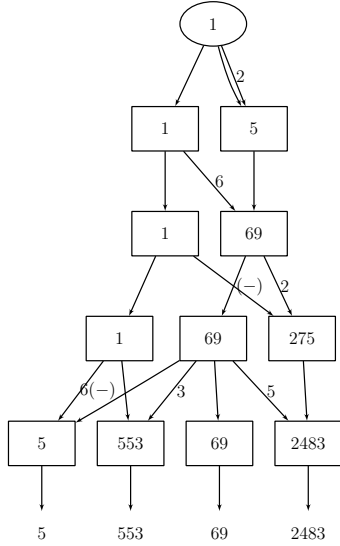


Fig. 4: Increasing the number of adders to reduce the pipelined adder graph overall cost

Register cost. Now, our overall idea is to avoid modeling the registers explicitly, as entities similar to adders, but to compute the register cost for each adder in the following way. For each adder $a \in \{a_0, a_1, \dots, a_{\overline{N}_A}\}$, where a_0 is the input and \overline{N}_A is the upper bound on the number of used adders, we can extract the stage when they are computed, s_a , and the maximum stage they interact with as a direct input, \overline{s}_a . For example, in Fig. 4, for the initial input the stage $s_{a_0} = 0$ and the maximum stage it interacts with is $\overline{s}_{a_0} = 4$ (to compute the fundamental 553).

Then, the cost r_a of each adder a in terms of registers is directly computed as

$$r_a = \overline{s}_a - s_a. \quad (1)$$

Hence, for the input in Fig. 4, the register cost is $r_{a_0} = 4 - 0$.

With these observations, we can expect that the optimal PMCM might have more adders than the solutions by MCM-Adders and MCM-Bits. In Section IV, we illustrate that it indeed occurs.

We use as basis the ILP models from [18] and below report only the modifications to them.

A. PMCM-Adders

The ILP from [18], given the target coefficients, first computes the upper bound on the minimum number of adders \overline{N}_A . Then, the solver searches the fundamentals associated with each adder $a \in \{a_0, a_1, \dots, a_{\overline{N}_A}\}$, such that the topology abides to the rules of adder graphs and computes the target coefficients, such that the cost function is minimized.

Denote $s_a \in \mathbb{N}$ to be the stage, on which the adder a is computed. Then, for all adders a and stages s we add the binary variable $\overline{s}_{a,s}$, such that $\overline{s}_{a,s} = 1$ if adder a is used as an input of stage s . The maximum number of stages can be computed *a priori* using the bound \overline{N}_A on the adder count. Then, in order to compute the maximum stage with which the adder a interacts with, $\overline{s}_a \in \mathbb{N}$, we add the constraints

$$\overline{s}_a \geq s \times \overline{s}_{a,s}, \quad \forall a, s, \quad (2)$$

Then we simply introduce the equation (1) to count the register cost of each adder.

To minimize the number of registers, the objective function should be adjusted accordingly. In the original models, the objective function minimizes the sum of actually used adders, which are indicated by binary variables u_a , that are true if the adder a is used. We use this objective function as a basis and add the register cost into the weighted sum representing the total PMCM cost:

$$\min \text{cost}_a \sum_{a=1}^{\overline{N}_A} u_a + \text{cost}_r \sum_{a=1}^{\overline{N}_A} r_a, \quad (3)$$

where cost_a and cost_r are the weights of the adder and register costs, respectively. This objective function with the original MCM-Adders model and the new variables and constraints permits to solve the PMCM-Adders problem.

B. PMCM-Bits

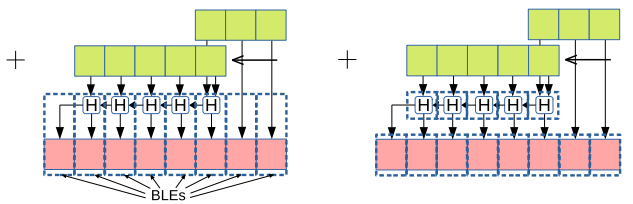
When the input data word length is *a priori* known, solving the MCM-Bits problem leads to hardware with lower cost [18] than MCM-Adders, which should hold for the PMCM-Bits model we present below. Indeed, counting the one-bit adders presents a finer-grained metric.

In the original MCM-Bits model, the most significant bit (MSB) propagation of the data is already encoded by integer variables for MSB propagation while the least significant bit (LSB) is fixed to zero. For simplicity, we consider the integer variable wl_a corresponding to the data word length after adder a . Then, the number of one-bit registers, r_a^b , required for each adder is

$$r_a^b = wl_a \times r_a. \quad (4)$$

TABLE I: Comparison of our results to RPAG [8]. Here #A, #A_b, #R and #R_b denote the number of adders, one-bit adders, registers, and one-bit registers, respectively. $C = \#A + \#R$ and $C_b = \#A_b + \#R_b$. Numbers in bold indicate best results.

Bench	RPAG						PMCM-Adders						PMCM-Bits					
	#A	#R	C	#A _b	#R _b	C _b	#A	#R	C	#A _b	#R _b	C _b	#A	#R	C	#A _b	#R _b	C _b
G.3	4	2	6	43	25	68	4	1	5	43	15	58	4	1	5	43	15	58
G.5	6	2	8	69	34	103	6	2	8	69	34	103	6	2	8	69	34	103
HP5	4	4	8	42	45	87	4	2	6	42	24	66	4	2	6	42	24	66
HP9	5	4	9	50	49	99	5	2	7	50	27	77	5	2	7	50	27	77
HP15	12	7	19	122	104	226	12	3	15	122	56	178	12	3	15	122	56	178
L.3	4	2	6	45	24	69	4	1	5	45	16	61	5	0	5	58	3	61
LP5	7	2	9	75	40	115	7	1	8	68	30	98	8	1	9	85	25	110
LP9	13	6	19	152	102	254	15	3	18	167	68	235	13	4	17	145	81	226
LP15	27	26	53	306	434	740	27	9	36	304	191	495	27	10	37	306	201	507
U.3-1	4	2	6	32	34	66	4	1	5	38	18	56	4	1	5	35	21	56
U.3-2	6	3	9	62	50	112	5	2	7	60	33	93	5	2	7	59	32	91
Total:	92	60	152	998	941	1939	93	27	121	1025	508	1533	93	29	123	1029	517	1546



(a) One-bit adder outputs, carry outputs and go-through outputs are all BLEs. (b) One-bit adders and flip-flops are independent.

Fig. 5: Counting resources on (a) FPGA and (b) ASICs

This leads to a finer-grain objective function:

$$\min \text{cost}_b \sum_{a=1}^{\overline{N}_A} b_a + \text{cost}_{r_b} \sum_{a=1}^{\overline{N}_A} r_a^b, \quad (5)$$

where b_a is the number of one-bit adders of adder a , and cost_b and cost_{r_b} are the cost of one-bit adders and one-bit registers, respectively.

In details, each basic logic element (BLE) used in modern FPGAs consists of at least one one-bit adder and two optional flip-flops (FF). Thus, not the whole adder output comes with associated registers but only the output bits computed by a one-bit adder as illustrated in Fig. 5a. The carry outputs or the go-through output bits will have to be computed using the BLEs as well. As a consequence, we assume that the one-bit adders come for free in (5), *i.e.* $\text{cost}_b = 0$, and focus on counting register bits. If the FF and one-bit adders were on separated BLEs, as illustrated in Fig. 5b for ASICs, it would be straightforward to adjust our model by modifying the costs of in the objective function.

Previous works [10], [18] have shown that the adder depth and/or the glitch path count can be considered with bounds and/or as a second objective. Yet, pipelining makes both metrics of less interest and the critical path would be a better high-level metric to consider. The critical path of the pipelined adder graph is determined by the adder with the largest number of one-bit

TABLE II: A posteriori pipelining of MCM-Bits solution vs. one-step optimization with PMCM-Bits. $C_b = \#A_b + \#R_b$.

Bench	MCM-Bits			PMCM-Bits		
	#A _b	#R _b	C _b	#A _b	#R _b	C _b
G.3	40	28	68	43	15	58
G.5	57	55	112	84	32	116
HP5	39	77	116	42	24	66
HP9	47	41	88	50	27	77
HP15	105	230	335	122	56	178
L.3	31	59	90	58	3	61
LP5	60	89	149	85	25	110
LP9	137	218	355	145	81	226
LP15	250	688	938	306	201	507
U.3-1	32	34	66	35	21	56
U.3-2	49	85	134	59	32	91
Total:	847	1604	2451	1029	517	1546

adders. Using ILP modeling, it is straightforward to store this value into B_{\max} , an integer variable,

$$B_{\max} \geq b_a, \quad \forall a \in [1; \overline{N}_A]. \quad (6)$$

Then, the objective function can be modified as follows to minimize the critical path as a second objective,

$$\min M \times \left(\text{cost}_b \sum_{a=1}^{\overline{N}_A} b_a + \text{cost}_{r_b} \sum_{a=1}^{\overline{N}_A} r_a^b \right) + B_{\max}, \quad (7)$$

where the integer parameter M is fixed to a known upper bound of B_{\max} such as the maximum data word length in the adder graph, in order to ensure that the first term of the cost function is optimized first, and B_{\max} second.

IV. TOOL AND EXPERIMENT RESULTS

A. Optimization results

The flow of the jMCM tool is summarized in Fig. 1 and can be easily extended to include more high-level options similarly to our approach with the critical path. This tool is implemented in julia using the modeling language JuMP [23], so any generic

MILP solver supported by JuMP can be used for the solving process as backend. The tool is open-source and freely available on [git](https://github.com/remi-garcia/jMCM)¹.

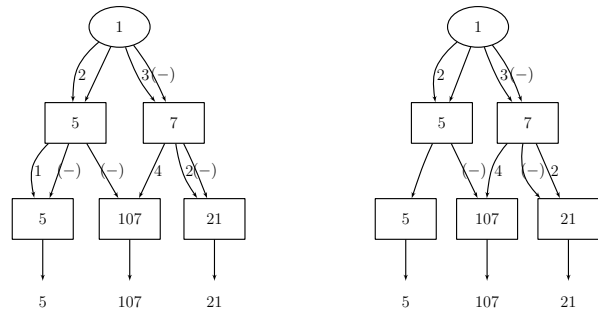
With `jMCM`, embedded system designers can provide target coefficients together with a just few high-level options, such as the input/output word length. Then, an ILP model, adjusted accordingly to the options, will be automatically generated and solved. This generates adder graphs that can be used to generate the VHDL code, for example using the FloPoCo [24] code generator or with our own VHDL generation tool for adder graphs.

We applied our tool to benchmarks from image processing (11 instances) [25] and from the whole FIRsuite project (75 instances) [26], which is a collection of finite impulse response digital filters. Generated models were solved with Gurobi [27], with a time limit of 30 minutes and 4 threads from 2.1 GHz CPUs. When solving the PMCM-Bits problem, input word length is fixed at 8 bits as it is a logical choice for the image-processing benchmarks. Detailed results are given only for the image processing benchmark while statistics integrate all 86 benchmarks.

PMCM. In [8] it was shown that solutions obtained with the RPAG heuristic require, on average, a lower number of adders plus registers than pipelining fixed adder graphs. In TABLE I, the state-of-the-art RPAG results are compared with our optimal approach for the PMCM-Adders. With our tool, we can observe a 17.57% reduction of the number of adders plus registers, on average and across all 86 benchmarks, compared to RPAG. In details, the number of adders is on average increased by 1.86% allowing to reduce the number of registers that are not associated to an adder by 49.26%. With this trade-off between the number of adders and the number of registers, for PMCM we discourage using tight bounds on the initial bound on the maximum number of adders, which is otherwise advised for MCM-Adders. For instance, in our experiments we arbitrarily increased by two every (initially tight) bound on the number of adders for each benchmark. For 11 instances out of 86, the larger upper bound permitted to obtain lower cost adder graphs.

PMCM-Bits. The state-of-the-art RPAG heuristic demonstrated [8] that solving the PMCM as one step is better, w. r. t. the number of adders and registers, than *a posteriori* pipelining an optimal adder graph. The results presented in TABLE II show that this principle holds for the fine-grained metric of counting one-bit adders and one-bit registers. By solving the PMCM-Bits problem, instead of pipelining *a posteriori* the MCM-Bits solution, we achieve a 33.15% cost reduction in terms of number of one-bit registers, on average.

However, the PMCM-Bits solutions, on average, demonstrate no improvement over the PMCM-Adders solutions. Moreover, we observed that solving PMCM-Bits is more challenging for the solver, which translates into obtaining, in rare cases, worse solutions than PMCM-Adders. Our hypothesis is that, for hard instances, solving the PMCM-Adders problem leads to an adder and register reduction that is not reached by solving the PMCM-Bits within the available time. On the other hand, this might be



(a) Adder is used instead of a register (b) Register can be used

Fig. 6: Postprocessing replaces unnecessary adders with same-cost registers

explained in the following way: the interest of MCM-Bits is to count the number of one-bit adders and look for corner cases that permit to avoid using them, but it is somewhat leveled by the register bit count, which always depends on the word length of the adder output. Though we did observe one case, benchmark `U.3-2`, where PMCM-Bits found a better adder graph than PMCM-Adders.

Post-processing. We observed that sometimes the solver provides solutions in which adders could have been replaced by registers such as in Fig. 6a. When targeting FPGA, this has the same cost as the register used in Fig. 6b since the adder comes for free in the model. Hence, the solver will not differentiate between these solutions. Hardware costs, such as power consumption, would however differ. These unnecessary adders can be replaced by registers with a trivial post-processing that `jMCM` supports.

B. Hardware results

Our tool has a module for the VHDL code generation, which we then synthesized for FPGA using Vivado v2018.2 for the `xc7v585tffg1761-3` Kintex 7 device. Similarly to optimization results, although we only provide details for the image processing benchmarks, we give the statistics on all 86 benchmarks.

In TABLE IV, we compare the critical path delay of five different approaches: RPAG, MCM-Adders (no pipelining), PMCM-Adders, PMCM-Bits and PMCM-Bits minimizing the critical path as a second objective. As expected, pipelining leads to smaller critical path which permits to increase the frequency and the throughput of the final hardware. Interestingly, minimizing the critical path as a second objective was not as efficient as we expected: on average, the critical path delay was reduced by just 0.37% and simply solving PMCM-Bits was even better on multiple instances, in the allocated solving time. The main objective of PMCM-Bits_{CP} was to minimize resources, and the critical path was secondary, hence we think that modifying the cost function and allowing for a small increase in resource objective could improve the delay.

The TABLE I and II demonstrated the advantages of the proposed approach over the state-of-the-art heuristic RPAG on the *a priori* metrics, *i. e.* number of adders and registers.

¹<https://github.com/remi-garcia/jMCM>

TABLE III: Comparison of adder graphs obtained with different high- and low-level models, including pipelining or applied on a fixed adder graph. Registers are also added on the I/Os. We report the number of LUTs and FF and the power consumption (mW) after place and route. Numbers in bold indicate best results. GM stands for geometric mean.

Bench	Two-step vs one-step counting adders									Two-step vs one-step counting one-bit adders					
	RPAG			MCM-Adders + P.			PMCM-Adders			MCM-Bits + P.			PMCM-Bits		
	#LUTs	#FF	power	#LUTs	#FF	power	#LUTs	#FF	power	#LUTs	#FF	power	#LUTs	#FF	power
G.3	43	75	116	42	76	114	42	65	114	39	74	112	42	65	114
G.5	69	109	170	58	148	174	68	108	170	57	138	180	68	108	170
HP5	42	93	127	41	92	126	41	73	125	37	118	134	41	73	125
HP9	50	105	163	48	104	156	48	84	156	44	94	161	48	84	156
HP15	121	226	408	119	225	400	118	184	398	100	315	413	118	184	398
L.3	45	75	121	34	95	122	44	68	126	31	93	118	44	68	126
LP5	76	115	213	66	157	218	77	111	221	59	142	212	70	104	207
LP9	149	250	483	143	354	510	179	229	510	139	396	545	153	229	507
LP15	303	691	1181	302	690	1175	311	487	1181	330	791	1219	296	498	1204
U.3-1	32	73	109	30	72	103	35	63	106	30	72	103	35	63	106
U.3-2	62	117	179	59	97	174	116	126	197	51	139	178	61	116	178
GM:	71	133	217	66	146	215	77	117	218	62	158	219	70	116	215

TABLE IV: Critical path delay (ns) for each method.

Bench.	MCMA	RPAG	PMCM-A	PMCM-B	PMCM-BCP
G.3	2.643	1.452	1.494	1.494	1.494
G.5	4.608	1.792	1.772	1.772	1.772
HP5	2.629	1.887	1.566	1.566	1.566
HP9	2.731	1.698	1.712	1.712	1.712
HP15	5.142	2.791	3.423	3.423	3.423
L.3	3.926	1.497	1.502	1.502	1.502
LP5	3.687	1.931	1.705	1.662	1.824
LP9	7.458	3.079	2.875	3.304	3.099
LP15	13.147	5.897	7.586	5.178	5.415
U.3-1	2.535	1.641	1.630	1.630	1.547
U.3-2	4.020	1.547	1.623	1.409	1.409

In TABLE III, we recap the synthesis results for the RPAG, the two-step process consisting in solving MCM-Adders (respectively, MCM-Bits) first and pipelining second, and our optimal approach. Interestingly, the RPAG heuristic is not better than the two-step approaches, but this can be explained by significant improvements to the underlying MCM model that are incorporated into the MCM-Adders and MCM-Bits from [18] that we used as back-end. We see the clear advantage of our optimal solutions with respect to the number of FFs, which are almost everywhere significantly reduced by both PMCM-Adders and PMCM-Bits. However, synthesis results confirm our observation that PMCM-Bits in general has shown no advantage to PMCM-Adders in the allocated time. Finally, the gains in results of number of LUTs and power are more erratic and the one-step process does not lead to a clear cost reduction compared to a *posteriori* pipelining. This inconclusive is nevertheless not discouraging since all these values are close.

V. CONCLUSION AND PERSPECTIVES

With this work, we proposed an optimal approach to incorporating pipelining into the ILP-based design of multiplierless MCM. Our results, evaluated on a large set of benchmarks, demonstrate the superiority of *optimal* approach compared to

the state-of-the-art heuristic RPAG. In contrast to previous approaches, we also incorporate the pipelining into a model that targets a *low-level metric*, based on counting one-bit adders. Our experiments confirm that optimizing in one step, *i.e.* PMCM-Bits, is better than a *posteriori* pipelining of MCM-Bits solutions, following the same trend as with counting adders. However, in the allocated time, PMCM-Bits design exploration is rarely revealing better results than the high-level metric-based PMCM-Adders.

With this work we demonstrate the versatility and power of the ILP-based approach: one can rely on the *jMCM* tool and, by simple modification of constraints/cost function, automatically design multiplierless constant multiplication circuits that target different metrics. This new design automation tool permits to alleviate time and effort embedded system designers put into the data path optimization and fine-tuning, and lets them focus on more high-level tasks related to their implemented algorithm.

In future, we plan to search for high-level metrics that could be correlated with the delay and power consumption and to incorporate these into the objective function. We also plan on to tackle truncated pipelined adder graphs, modifying the constraints from [18]. Finally, we plan to automate the (pipelined) MCM design in presence of a certain budget of DSP blocks.

ACKNOWLEDGMENT

The authors would like to thank Martin Kumm for valuable discussions and remarks. This paper is at this current state thanks to the effort of multiple anonymous reviewers, whom we thank for their helpful comments and remarks.

REFERENCES

- [1] A. Oppenheim and R. W. Schäfer, *Discrete-time signal processing*. Englewood Cliffs, N.J: Prentice Hall, 1989.
- [2] M. Garrido, K. Moller, and M. Kumm, "World's Fastest FFT Architectures: Breaking the Barrier of 100 GS/s," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 4, pp. 1507–1516, Apr. 2019.

- [3] T. Habermann, J. Kühle, M. Kumm, and A. Volkova, "Hardware-Aware Quantization for Multiplierless Neural Network Controllers," in *IEEE Asia Pacific Conference on Circuits and Systems*, Shenzhen, China, Nov. 2022. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-03827660>
- [4] K. Wiatr and E. Jamro, "Constant coefficient multiplication in FPGA structures," in *Proceedings of the 26th Euromicro Conference. EUROMICRO 2000. Informatics: Inventing the Future*. IEEE Comput. Soc, Sep. 2000.
- [5] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proceedings - Circuits, Devices and Systems*, vol. 141, no. 5, pp. 407–413, Oct. 1994.
- [6] O. Gustafsson, "Towards optimal multiple constant multiplication: A hypergraph approach," in *2008 42nd Asilomar Conference on Signals, Systems and Computers*, Oct. 2008, pp. 1805–1809.
- [7] L. Aksoy, E. O. Güneş, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," *Microprocessors and Microsystems*, vol. 34, no. 5, pp. 151–162, Aug. 2010.
- [8] M. Kumm, P. Zipf, M. Faust, and C.-H. Chang, "Pipelined adder graph optimization for high speed multiple constant multiplication," in *2012 IEEE International Symposium on Circuits and Systems*. IEEE, May 2012.
- [9] M. Kumm, *Multiple Constant Multiplication Optimizations for Field Programmable Gate Arrays*. Wiesbaden: Springer Fachmedien Wiesbaden, 2016.
- [10] —, "Optimal Constant Multiplication Using Integer Linear Programming," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 567–571, May 2018.
- [11] M. Kumm and P. Zipf, "High speed low complexity FPGA-based FIR filters using pipelined adder graphs," in *2011 International Conference on Field-Programmable Technology*. IEEE, Dec. 2011.
- [12] A. D. Booth, "A Signed Binary Multiplication Technique," *The Quarterly Journal of Mechanics and Applied Mathematics*, vol. 4, no. 2, pp. 236–240, 1951.
- [13] R. Bernstein, "Multiplication by integer constants," *Software: Practice and Experience*, vol. 16, no. 7, pp. 641–652, Jul. 1986.
- [14] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 2, pp. 151–165, Feb. 1996.
- [15] V. Lefèvre, "Multiplication by an Integer Constant," INRIA, Research Report RR-4192, May 2001. [Online]. Available: <https://hal.inria.fr/inria-00072430>
- [16] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Transactions on Algorithms*, vol. 3, no. 2, p. 11, May 2007.
- [17] J. Thong and N. Nicolici, "Combined optimal and heuristic approaches for multiple constant multiplication," in *2010 IEEE International Conference on Computer Design*, Oct. 2010, pp. 266–273.
- [18] R. Garcia and A. Volkova, "Toward the Multiple Constant Multiplication at Minimal Hardware Cost," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2023.
- [19] V. Lagoon and A. Metodi, "Deriving Optimal Multiplication-by-Constant Circuits With A SAT-based Constraint Engine," ModRef 2020 – The 19th workshop on Constraint Modelling and Reformulation, Sep. 2020.
- [20] U. Meyer-Baese, J. Chen, C. H. Chang, and A. G. Dempster, "A Comparison of Pipelined RAG-n and DA FPGA-based Multiplierless Filters," in *APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems*. IEEE, dec 2006.
- [21] K. K. Parhi, *VLSI digital signal processing systems*. Wiley, 1999.
- [22] X. Lou, P. K. Meher, and Y. J. Yu, "Fine-grained pipelining for multiple constant multiplications," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2015.
- [23] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A Modeling Language for Mathematical Optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, May 2017.
- [24] F. de Dinechin and B. Pasca, "Designing Custom Arithmetic Data Paths with FloPoCo," *IEEE Design & Test of Computers*, vol. 28, no. 4, pp. 18–27, Jul. 2011.
- [25] M. Kumm, D. Fanghänel, K. Möller, P. Zipf, and U. Meyer-Baese, "FIR filter optimization for video processing on FPGAs," *EURASIP Journal On Advances in Signal Processing*, vol. 2013, no. 1, May 2013.
- [26] FIRsuite, "Suite of Constant Coefficient FIR Filters," 2021, Accessed: Jan., 2022. [Online]. Available: <http://www.firsuite.net>
- [27] Gurobi Optimization, "Gurobi Optimizer Reference Manual," 2020. [Online]. Available: <https://www.gurobi.com/>