Unleashing the Power of T1-cells in SFQ Arithmetic Circuits

Rassul Bairamkulov, Mingfei Yu and Giovanni De Micheli
Integrated Systems Laboratory, EPFL
Lausanne, Switzerland
{rassul.bairamkulov,mingfei.yu,giovanni.demicheli}@epfl.ch

***Abstract*—Rapid single-flux quantum (RSFQ), a leading cryogenic superconductive electronics (SCE) technology, offers extremely low power dissipation and high speed. However, implementing RSFQ systems at VLSI complexity faces challenges, such as substantial area overhead from gate-level pipelining and path balancing, exacerbated by RSFQ's limited layout density.**

T1 flip-flop (T1-FF) is an RSFQ logic cell operating as a pulse counter. Using T1-FF the full adder function can be realized with only 40% of the area required by the conventional realization. This cell however imposes complex constraints on input signal timing, complicating its use. Multiphase clocking has been recently proposed to alleviate gate-level pipelining overhead. The fanin signals can be efficiently controlled using multiphase clocking. We present the novel two-stage SFQ technology mapping methodology supporting the T1-FF. Compatible parts of the SFQ network are first replaced by the efficient T1-FFs. Multiphase retiming is next applied to assign clock phases to each logic gate and insert DFFs to satisfy the input timing. Using our flow, the area of the SFQ networks is reduced, on average, by 6% with up to 25% reduction in optimizing the 128-bit adder.

I. Introduction

Rapid Single-Flux Quantum (RSFQ) [1] is a prominent superconductive digital technology. RSFQ circuits operate at tens to hundreds of gigahertz and dissipate two to three orders of magnitude less power as compared to CMOS, even accounting for refrigeration [2]. These advantages position RSFQ as a compelling candidate for large-scale stationary computing [3], space electronics [2] and interface circuitry for quantum computing systems [4]. These applications require efficient arithmetic circuits.

RSFQ systems consist of Josephson Junctions (JJ) and superconductive storage loops communicating using single flux quantum (SFQ) pulses. [1] Due to pulse-based operation, most SFQ gates, such as `NOT` and `XOR`, require a clock signal, necessitating gate-level pipelining. Fanins to each clocked SFQ gate need to have equal logic depth to operate correctly. To satisfy this requirement, dummy DFFs are inserted into the network, occupying a significant portion of the layout.

A. *T1-flip-flop*

T1-flip-flop [5] is an SFQ device with two inputs, `T` (*toggle*) and `R` (*reset*); and three outputs, `S` (*sum*), `C` (*carry*) and `Q`, as illustrated in Fig. 1a. Initially, the bias current I_0 flows along the blue dotted line, corresponding to the internal state `0`. A pulse arriving at input `T` switches $J_{\texttt{Q}}$, producing the pulse at output `Q`* (see Fig. 1b). The bias current is redirected along the red solid arrows, corresponding to the storage of logical `1`. A second pulse at input `T` switches $J_{\texttt{C}}$, producing the pulse at output `C`*, and resetting the bias current towards $J_{\texttt{Q}}$, i.e., logical `0`. If the loop state is `1`, a pulse at the input `R` switches $J_{\texttt{S}}$, producing the pulse at output `S`, while resetting the loop state to `0`. If the loop state is `0`, a pulse at the input `R` is simply rejected by $J_{\texttt{R}}$.

The T1-FF can realize a full adder with only 29 JJs, 60% fewer than a regular implementation [6] (see Fig. 1c). Outputs `R`, `C`, and `Q` execute, respectively, `XOR3`, majority-3 (`MAJ3`), and `OR3` functions. In addition, outputs `C`* and `Q`* can be connected to inverters to produce inverted `MAJ3` and `OR3`. Therefore, the extended T1-FF can efficiently produce up to five synchronous outputs. The main challenge of using T1-FFs is temporal separation of input pulses. Two overlapping input pulses may be treated as a single pulse, producing a data hazard. We propose using multiphase clocking to mitigate this issue.

B. *Multiphase clocking*

An n-phase system utilizes n periodic signals $\{t_0, \cdots, t_{n-1}\}$ operating at the same frequency [7]. Each clocked element g within the network is synchronized by only one clock signal at phase $\varphi(g)$. The epoch $S(g)$ of a gate g is defined as the number of clock cycles separating the gate g from the primary inputs. The clock signals are ordered by phase $\varphi \in \{0, \cdots, n-1\}$, i.e., during any epoch, the clock signal t_i arrives before clock signal t_j if $i < j$. For convenience, we define a *stage* $\sigma(g)$ of a gate g as

$$\sigma(g) = nS(g) + \varphi(g). \tag{1}$$

We observe that multiphase clocking enables precise control of the input arrival time, as illustrated by phases φ_0, φ_1, and φ_2 in Fig. 1c. The input a is released to the T1-FF at φ_0, next b is released at φ_1, and, finally, c is released at φ_2; i.e., assigning three different phases to the inputs of a T1-FF is sufficient to ensure no temporal overlap.

II. T1-aware technology mapping

We present next the three-stage T1-aware technology mapping flow. First, compatible parts of the logic network are

[1] For background in RSFQ technology, an interested reader is referred to [2]

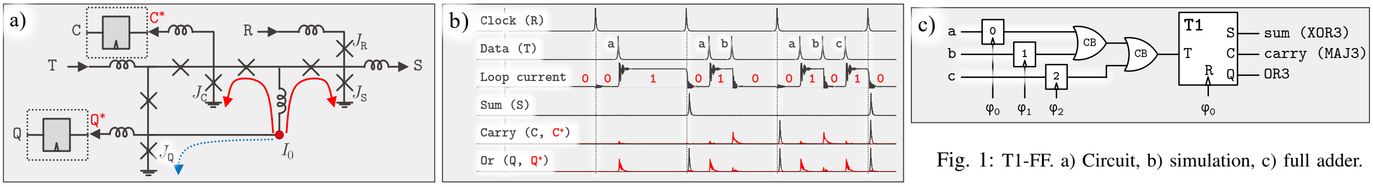

Fig. 1: T1-FF. a) Circuit, b) simulation, c) full adder.

TABLE I: Multiphase clocking with T1 cells applied to a subset of EPFL and ISCAS benchmark circuits

	T1 cells		#DFF			Ratio vs.		Area			Ratio vs.		Depth			Ratio vs.	
benchmark	found	used	1φ	4φ	T1	1φ	4φ	1φ	4φ	T1	1φ	4φ	1φ	4φ	T1	1φ	4φ
`adder`	127	127	32’768	7’963	5’958	0.18	0.75	238’419	64’784	48’844	0.20	0.75	128	32	33	0.26	1.03
`c7552`	17	9	2’489	713	765	0.31	1.07	32’038	19’606	19’907	0.62	1.02	16	4	5	0.31	1.25
`c6288`	142	142	2’625	1’431	1’349	0.51	0.94	47’198	38’840	35’386	0.75	0.91	29	8	10	0.34	1.25
`sin`	81	77	13’416	4’631	4’714	0.35	1.02	164’938	103’443	102’806	0.62	0.99	88	22	25	0.28	1.14
`voter`	252	252	10’651	5’779	5’584	0.52	0.97	222’101	187’997	182’972	0.82	0.97	38	10	11	0.29	1.10
`square`	861	806	44’675	16’645	14’304	0.32	0.86	525’311	329’101	301’287	0.57	0.92	126	32	32	0.25	1.00
`multiplier`	824	769	58’717	14’641	13’745	0.23	0.94	682’792	374’260	356’984	0.52	0.95	136	33	36	0.26	1.09
`log2`	644	593	86’985	33’790	33’946	0.39	1.00	978’178	605’813	598’292	0.61	0.99	160	40	47	0.29	1.18
Average						0.35	0.94				0.59	0.94				0.29	1.13

replaced by the T1-FF. Then, we formulate an integer linear programming problem to assign a phase to each gate, while minimizing the number of DFFs. Finally, the DFFs are inserted to satisfy the timing requirements of each gate, including T1-FFs.

A. T1-FF detection

Our T1-FF detection is based on cut enumeration [8] followed by Boolean matching [9]. If a set of cuts $\mathcal{C} = \{C(u_1), \ldots, C(u_n)\}$, $2 \leq n \leq 5$ sharing the same leaves $\{a, b, c\}$ executes the functions implementable with the T1-FF, the cuts $\{C(u_1), \ldots, C(u_n)\}$ are considered for being replaced by a T1-FF. To ensure the substitution is beneficial, the area reduction ΔA due to replacement is calculated as

$$\Delta A = \sum_{i=1}^{n} A(\mathtt{MFFC}(u_i)) - A_{\mathtt{T1}}(\mathcal{C}), \tag{2}$$

where $A(\mathtt{MFFC}(u_i))$ is the total area of the nodes within the maximum fanout free cone (MFFC) of node u_i, and $A_{\mathtt{T1}}(\mathcal{C})$ is the area of the T1-FF implementing the functions realized by cuts in $\mathcal{C}$ considering possible input and output negations. If $\Delta A > 0$, the MFFCs of nodes $u_1, \ldots, u_n$ are replaced by the T1-FF-based circuit.

B. Phase assignment

The phase assignment closely follows the integer linear programming (ILP) procedure described in [10]. The T1-FF however requires the constraint and objective functions to be modified. The clock stage $\sigma_{\mathtt{T1}}$ of T1-FF is constrained as

$$\sigma(j) \geq \max\left(\sigma(i_1) + 3, \sigma(i_2) + 2, \sigma(i_3) + 1\right), \tag{3}$$

where i_1, i_2, i_3 are the fanins of T1-FF and $\sigma(i_1) \leq \sigma(i_2) \leq \sigma(i_3)$. Condition (3) is incorporated into the set of ILP constraints, ensuring that the necessary DFFs can be inserted between the inputs and the T1-FF. The number of DFFs required by a T1-FF is determined as

$$\begin{aligned} c_{\mathtt{T1}} = (\phi(i_1) = \phi(i_2)) \wedge (\sigma_{\mathtt{T1}} - \sigma(i_1) \leq n) + \\ + (\phi(i_2) = \phi(i_3)) \wedge (\sigma_{\mathtt{T1}} - \sigma(i_2) \leq n)\,. \end{aligned} \tag{4}$$

$c_{\mathtt{T1}}$ is added to the ILP objective function.

C. DFF insertion

After assigning the stage to each gate, the DFFs can be inserted to each datapath. We extend the DFF insertion methodology based on CP-SAT, described in [10], to support the T1-FF. The inputs to the T1-FF should arrive at different stages. To satisfy this condition, the DFFs d_1, d_2, d_3 preceding the T1-FF at stage $\sigma_{\mathtt{T1}}$ should be placed at a different stage,

$$\sigma(a) \neq \sigma(b)\ \forall\ a \neq b \quad a, b \in \{d_1, d_2, d_3\} \tag{5}$$

III. Experimental results

We integrate the proposed methodology into the technology mapping flow implemented in `mockturtle` logic synthesis library [11]. Phase assignment and DFF insertion procedures are implemented using Google OR-Tools [12]. We apply our flow to synthesize a subset of EPFL [11] and ISCAS [13] benchmark circuits implementing arithmetic functions. We ran our experiments on a laptop with an Apple M1 10-core CPU with 64 GB of RAM. The number of path-balancing DFFs, circuit area (expressed in the number of JJs), and logic depth (in cycles) are shown in Table I. We compare our synthesis results (column T1) with the single-phase (1φ) and four-phase (4φ) clocking without T1-FF. Compared to 4φ, our methodology achieves, on average, a 6% improvement in both area and the number of path-balancing DFFs at the cost of a 13% increase in the logic depth. The increase in depth can be attributed to the additional stages required by the T1-FFs. The largest reduction is observed in the `adder` circuit where almost the entire circuit is replaced with the T1-FFs, yielding a 25% improvement in area. Significant reduction is also observed in `voter`, `square` and `multiplier`, while `c7552` and `sin` yielded inferior area, likely due to the increase in the circuit depth, requiring additional path balancing DFFs.

References

[1] K. Likharev *et al.*, “Resistive Single Flux Quantum Logic for the Josephson-Junction Digital Technology,” *Proc. SQUID*, Vol. 85, 1985.

[2] G. Krylov and E. G. Friedman, *Single Flux Quantum Integrated Circuit Design*, Springer, 2022.

[3] D. S. Holmes *et al.*, “Energy-Efficient Superconducting computing – Power Budgets and Requirements,” *IEEE TASC*, Vol. 23, No. 3, 2013.

[4] M. R. Jokar *et al.*, “DigiQ: A Scalable Digital Controller for Quantum Computers Using SFQ Logic,” *IEEE HPCA*, 2022.

[5] S. Polonsky *et al.*, “Single Flux, Quantum B Flip-Flop and Its Possible Applications,” *IEEE TASC*, Vol. 4, No. 1, 1994.

[6] S. Yorozu *et al.*, “A Single Flux Quantum Standard Logic Cell Library,” *Physica C: Superconductivity*, Vol. 378-381, 2002.

[7] X. Li *et al.*, “Multi-Phase Clocking for Multi-Threaded Gate-Level-Pipelined Superconductive Logic,” *Proc. ISVLSI*, 2022.

[8] J. Cong *et al.*, “Cut Ranking and Pruning: Enabling a General and Efficient FPGA Mapping Solution,” *Proc. FPGA*, 1999.

[9] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill Higher Education, 1994.

[10] R. Bairamkulov and G. De Micheli, “Towards Multiphase Clocking in Single-Flux Quantum Systems,” *Proc. ASP-DAC*, 2024.

[11] M. Soeken *et al.*, “The EPFL Logic Synthesis Libraries,” *arXiv Preprint arXiv:1805.05121v3*, 2018.

[12] L. Perron, “Operations Research and Constraint Programming at Google,” *Proc. CP*, 2011.

[13] M. C. Hansen *et al.*, “Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering,” *IEEE Des. Test. Comput.*, Vol. 16, No. 3, 1999.