Mutual Exclusion Sizing for Hoi Polloi

Swetha Mettala Gilla, Marly Roncken, Ivan Sutherland, and Xiaoyu Song

***Abstract*—Granting access to a shared self-timed resource requires a mutual exclusion circuit to resolve contention. All such circuits contain cross-coupled logic gates to decide contested cases on a first-come-first-served basis. An end-of-metastability detector grants a decision only after metastability, if any, ends. This brief contrasts two previously published mutual exclusion designs and offers previously unavailable guides to achieving least *uncontested grant* delay for each design. The faster design integrates its cross-coupled gates and end-of-metastability detector into a single stage. The slower design pays a time penalty by isolating these two functions in separate sequential logic stages.**

***Index Terms*—Mutual exclusion, arbiter, self-timed circuits, asynchronous logic, transistor sizing.**

I. Introduction

A CLOCKED system can use priority to choose between contending requests for a shared resource. In contrast, a self-timed system must arbitrate on a first-come-first-served basis. This brief considers how to optimize self-timed first-come-first-served circuits, also known as *Mutual Exclusion* circuits or simply as *Arbiters*.

Although arbiters superficially resemble the synchronizers seen in clocked systems, their design criteria differ [4], [5]. A clocked system suffers errors if metastability lasts longer than a fixed time, and so synchronizer designs must focus on rapid exit from metastability. Self-timed systems avoid errors by waiting for metastability to end. Self-timing gains freedom from error by accommodating uncertain delay.

A mutual exclusion or arbiter circuit chooses which of two contending requests for a shared resource shall first be served [10]. Arbitration is an essential element for designing asynchronous, or self-timed, circuits and systems [7], [11]. Besides appearing as a design element in its own right, arbitration also appears in Q-flops [9] as a crucial part of just-in-time adjustable delay [1], in MrGO for test and debug [8], and in synchronous-asynchronous interfaces [3]. Recent work by Zhang *et al.* shows how to test arbiter circuits [13].

Arbiters also play a role in the search for hardware security primitives to build a secret key for identification and authentication that is unique for each integrated circuit [2], [6].

Manuscript received June 11, 2018; revised August 28, 2018; accepted September 13, 2018. Date of publication October 4, 2018; date of current version May 28, 2019. This work was supported by DARPA *Flexible Specification, Analysis, and Implementation of Self-Timed Circuits* under Grant UTA17-000001. This brief was recommended by Associate Editor Y. Liu. *(Corresponding author: Marly Roncken.)*

The authors are with the Department of Electrical and Computer Engineering, Portland State University, Portland, OR 97207 USA, and also with the Asynchronous Research Center, Portland State University, Portland, OR 97207 USA (e-mail: swetha.mettalagilla@gmail.com; mroncken@pdx.edu; ies@pdx.edu; songx@pdx.edu).

Digital Object Identifier 10.1109/TCSII.2018.2874009

Our focus here is on two CMOS two-way arbiter circuits based on Seitz [10] and published in [8] and [11]. Both are used and referenced frequently but without the practical guide to design included here.

II. A Tale of Two Arbiters

Seitz' early design for an nMOS arbiter [10] detects the end of metastability when the difference in voltage of the two outputs of its cross-coupled logic gates exceeds a transistor threshold. The two CMOS arbiters in this brief follow Seitz' scheme. The ARC arbiter in Fig. 1(a) combines cross-coupled gates and threshold circuit into a single inverting logic stage. The SF arbiter in Fig. 1(b) separates them into two inverting logic stages and thus avoids net signal inversion.

The two arbiter schematics in Fig. 1 are often drawn using logic gate notations with cross-coupled NAND gates. For the ARC arbiter, the two series nMOS transistors A and B and the two parallel pMOS transistors P_1 and P_2 on each side form a two-input NAND gate. Each NAND output, x or y, serves as input for the other NAND. Likewise, for the SF arbiter, transistors A, B, P_1, and Q on the left form a two-input NAND gate that is cross-coupled to the two-input NAND gate formed by transistors A, B, P_1, and Q on the right. Fig. 1 uses the expanded transistor notation because we will size NAND gate transistors A, B, and Q separately and individually.

The arrows in Fig. 1 trace the signal flow from request R_1 to grant G_1. The purple arrow in Fig. 1(a) traces how a rising input on R_1 turns on the series transistors A and B on the left to pull down the NAND gate's output signal labeled x, thus disabling A and enabling P_2 on the right to leave y high. Output G_1 goes low only after x and y differ by more than the threshold of transistor M. The green arrow in Fig. 1(b) traces how a rising input on R_1 drives x low to disable A and enable Q on the right to leave y high. The red arrow traces how the current flows through Q and M in series to drive output G_1 high when x becomes more than a threshold lower than y. Note that the grants in Fig. 1(a) fall whereas those in Fig. 1(b) rise. End-of-metastability detection in each circuit is done by M, but note that M's transistor type and connections are very different in the two circuits.

III. Least Uncontested Grant Delay

This brief explains to anyone[1] who plans to use either arbiter in Fig. 1 how best to size its transistors. In self-timed systems, metastability occurs only upon the very rare near-simultaneous arrival of contending requests. Therefore, we seek to minimize

[1]The double entendre of "Hoi Polloi" in the title refers to the original use in Greek for "the majority" of people or cases.

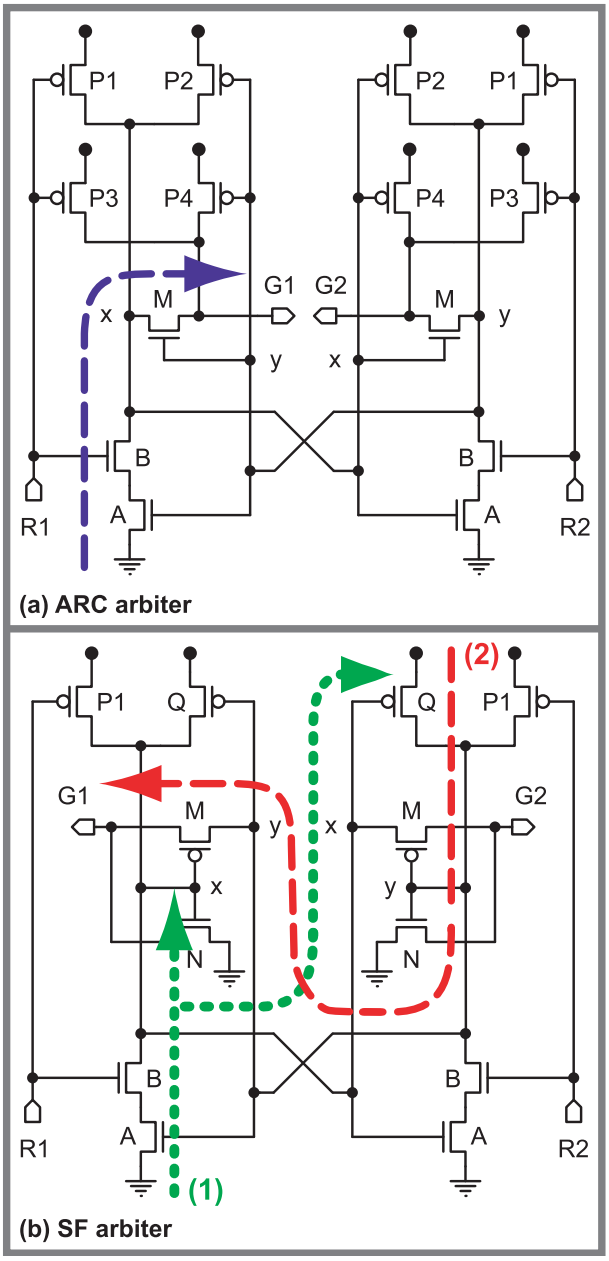

Fig. 1. Schematics of (a) ARC arbiter and (b) Sparsø-Furber (SF) arbiter. The goal of this brief is to optimize transistor drive strengths for least uncontested grant delay as represented by the purple R_1 to G_1 path in (a) and the corresponding combination of green and red paths in (b). Key transistors to optimize are A, M, and Q, which we size relative to B. The other transistors play various keeper roles and influence each arbiter's release delay.

the grant delay in the vast majority[1] of cases where contention is absent — the uncontested grant delay.

Without loss of generality, we will use R_1 to G_1 as the reference uncontested grant path in each arbiter. We assume that the arbiters have reasonable input and output rise times.

IV. Mathematical Analysis

We will consider the $R1$ to $G1$ uncontested grant delay in each arbiter as partitioned serially over multiple paths. Each path consists of a combination of transistors driving an individual load. We can express the total delay as the sum of the path delays as if the delays for driving the individual loads were separate in time. In reality, the drives overlap in time. Nevertheless, the sum is valid because the time it takes to fill a combination of loads with charge is the sum of the times it would take to fill each load. Each path delay, D, can be computed as the ratio of the path's output load to its drive strength [12]:

$$D = Load/Strength \tag{1}$$

where

- nMOS and pMOS transistors present a load of γ_N respectively γ_P times their drive strength, with $\gamma_N + \gamma_P = 1$ representing the load of an inverter with an nMOS and a pMOS transistor of strength 1.
- Drive strengths of transistors in parallel add up.
- The combined strength of k transistors in series, each with strength S_i, for $i = 1, \ldots, k$, is expressed as $strength(S_1, \ldots, S_k) = 1/((1/S_1) + \cdots + (1/S_k))$.

We will model *switching* delays associated with changing the charge in a transistor's channel and we will model *parasitic* delays associated with changing the charge in a transistor's source or drain. For simplicity, we assume that a transistor's channel, source, and drain each present the same amount of load. Because the wire connections in each arbiter are short, we ignore their resistance and capacitance.

A. Mathematical Analysis for the ARC Arbiter

We consider the total delay for $R1$ to $G1$, marked by the purple arrow in Fig. 1(a), as partitioned serially over two paths: A, B in series driving M, and A, B, M in series driving G_1.

We first consider the delay for series transistors A and B driving M by driving their series output, x, low. The switching delay associated with driving x low, $D^{switching}_{ARC_1}$, concerns changing the charge in the channels of transistors A, P_2, P_4 on the right and M on the left, as explained below.

- *nMOS transistor A on the right:* Initially, A is turned on and its channel filled with charge. Then x goes low, turns off A, and drains this charge.
- *pMOS transistors P_2 and P_4 on the right:* Initially, gate x, the power source, and drains y and G_2 of P_2 and P_4 are all high, turning off P_2 and P_4, leaving their channels without charge. Then x goes low, turns on P_2 and P_4, and fills both channels with charge.
- *nMOS transistor M on the left:* Initially, M's gate y, source x, and drain G_1 are all high, turning off M, leaving its channel without charge. Then x goes low while the originally high G_1 drain is disconnected from the supply power because P_3 turns off. As a result, M turns on and its channel fills with charge.
- *Any other transistor can be ignored:* The switching delay for driving x low *excludes* changing the charge in the channel or channels of:
  - *nMOS transistor B on the left*, which is included in the external logic that drives R_1 high, and tied into the total delay as a stepup load on G_1,
  - *pMOS transistors P_1 and P_2 on the left*, which are without charge as P_1 is turned off by the

TABLE I
KEY TRANSISTOR DRIVE STRENGTH FORMULAS FOR *R1* TO *G1* LEAST UNCONTESTED GRANT DELAY FOR A STEPUP OF *s* TIMES THE INPUT LOAD AT *B*

Parameter		Strength for Least Delay w/o Parasitics	Strength for Least Delay w/ Parasitics
ARC	a	$\sqrt{b*(m+(s*b)+(\frac{\gamma_P}{\gamma_N}*(p_2+p_4)))}$	$\sqrt{b*((3*m)+((s+1)*b)+(\frac{\gamma_P}{\gamma_N}*(p_1+p_3+(2*(p_2+p_4)))))}$
	m	$\sqrt{strength(a,b)*s*b}$	$\sqrt{\frac{strength(a,b)}{3}*((s*b)+(\frac{\gamma_P}{\gamma_N}*(p_3+p_4)))}$
SF	a	$\sqrt{b*(n+(\frac{\gamma_P}{\gamma_N}*(m+q)))}$	$\sqrt{b*(b+n+(\frac{\gamma_P}{\gamma_N}*(p_1+(2*(m+q)))))}$
	m	$\sqrt{strength(a,b)*\frac{\gamma_N}{\gamma_P}*s*b}$	$\sqrt{\frac{strength(a,b)*(q+p_1)}{strength(a,b)+(2*(q+p_1))}*\frac{\gamma_N}{\gamma_P}*(n+(s*b))}$
	q	$-p_1+\sqrt{strength(a,b)*\frac{\gamma_N}{\gamma_P}*s*b}$	$-p_1+\sqrt{\frac{strength(a,b)}{2}*(m+(\frac{\gamma_N}{\gamma_P}*(n+(s*b))))}$

external logic that drives R_1 high and P_2 by its high gate, y,

- *nMOS transistor M on the right*, which has nothing in its channel to discharge when x goes low because its gate x, source y, and drain G_2 are initially high.

The parasitic delay associated with driving x low, $D^{parasitic}_{ARC_1}$, concerns changing the charge in the drains of B, P_1, and P_2 on the left and in the source of M on the left. Using equation (1) we can express $D^{switching}_{ARC_1}$ and $D^{parasitic}_{ARC_1}$ as indicated in the following equation (2). For notation, we lowercase a transistor's name to indicate its strength—e.g., a and p_2 represent the strengths of transistors A and P_2 respectively.

$$D^{switching}_{ARC_1} = \frac{(\gamma_N*(a+m))+(\gamma_P*(p_2+p_4))}{strength(a,b)}$$
$$D^{parasitic}_{ARC_1} = \frac{(\gamma_N*(b+m))+(\gamma_P*(p_1+p_2))}{strength(a,b)} \quad (2)$$

Let us now consider the second part of the purple arrow in Fig. 1(a), with series transistors A, B, M driving G_1 low. Because the internally G_1-connected transistors, P_3 and P_4, are turned off, this path's switching delay, $D^{switching}_{ARC_2}$, is determined entirely by the external load on G_1. For simplicity, we set the external load to stepup, s, times the load that transistor B presents to R_1, ignoring the typically much smaller transistors P_1 and P_3. The parasitic delay associated with driving G_1 low, $D^{parasitic}_{ARC_2}$, concerns changing the charge in the drains of the leftmost pMOS transistors P_3 and P_4 and nMOS transistor M. Using equation (1) we find:

$$D^{switching}_{ARC_2} = \frac{s*\gamma_N*b}{strength(a,b,m)}$$
$$D^{parasitic}_{ARC_2} = \frac{(\gamma_N*m)+(\gamma_P*(p_3+p_4))}{strength(a,b,m)} \quad (3)$$

Using (2) and (3) we formulate the $R1$ to $G1$ uncontested grant delays *with* (w/P) and *without* (w/oP) parasitics as:

$$D^{w/oP}_{ARC} = D^{switching}_{ARC_1} + D^{switching}_{ARC_2}$$
$$D^{w/P}_{ARC} = D^{w/oP}_{ARC} + D^{parasitic}_{ARC_1} + D^{parasitic}_{ARC_2} \quad (4)$$

Differentiating the delay expressions in (4) with respect to a and m, the strengths of major driving transistors A and M, and setting their partial derivatives to zero yields formulas for a and m at least uncontested grant delay *with* and *without* parasitics. These formulas, shown in Table I, express a and m relative to b, the drive strength of transistor B that dominates the input load.

B. Mathematical Analysis for the SF Arbiter

We partition the delay from $R1$ to $G1$ in the SF arbiter over the two paths marked by the green arrow and the red arrow in Fig. 1(b). The green arrow uses NAND transistors A and B to drive both M and Q by driving NAND output, x, low. The red arrow uses Q with P_1 in parallel and in series with M to drive G_1 high. The delay analysis is similar to the analysis in Section IV-A for the two partitions of the purple arrow in Fig. 1(a). The resulting a, m, and q formulas for least uncontested grant delay in Table I are based on switching and parasitic delay expressions $D^{switching}_{SF_1}$ and $D^{parasitic}_{SF_1}$ for the green arrow, and $D^{switching}_{SF_2}$ and $D^{parasitic}_{SF_2}$ for the red arrow:

$$D^{switching}_{SF_1} = \frac{(\gamma_N*(a+n))+(\gamma_P*(m+q))}{strength(a,b)}$$
$$D^{parasitic}_{SF_1} = \frac{(\gamma_N*b)+(\gamma_P*(m+q+p_1))}{strength(a,b)}$$
$$D^{switching}_{SF_2} = \frac{s*\gamma_N*b}{strength(q+p_1,m)}$$
$$D^{parasitic}_{SF_2} = \frac{(\gamma_N*n)+(\gamma_P*m)}{strength(q+p_1,m)}. \quad (5)$$

V. TRANSISTOR-LEVEL SIMULATION

To complement our mathematical analysis of the best drive strengths for least uncontested grant delay, we simulated the arbiters in a 90nm CMOS process. We used a SPICE analog circuit simulator to sweep drive strengths a, m, and q of key transistors A, M, and Q relative to drive strength b for B, set arbitrarily at 12. Identical inverters drove each arbiter for a

TABLE II
CALCULATED AND SIMULATED BEST STRENGTHS AND DELAYS WITH $b, \gamma_N, \gamma_P, n, p_1^{ARC}, p_1^{SF}, p_2, p_3, p_4 = 12, \frac{1}{3}, \frac{2}{3}, 4, 1, 2, 4, 1, 2$ AND $\tau = 7$ PS

Arbiter	Parameter	Analysis w/o Parasitics			Analysis w/ Parasitics			Simulation		
		s=1	s=2	s=3	s=1	s=2	s=3	s=1	s=2	s=3
ARC	a	20.0	24.5	28.0	29.7	33.2	36.3	22	22	24
	m	9.5	13.9	17.4	7.2	9.4	11.2	8	8	12
	Delay in τ (ps)	2.8	3.7	4.4	5.2	6.1	7.0	3.6 (25.3)	4.1 (28.7)	4.4 (31.0)
SF	a	17.7	21.5	24.0	26.7	30.0	32.5	24	22	24
	m	6.6	9.6	12.0	4.6	6.4	7.9	6	8	10
	q	4.6	7.6	10.0	5.2	7.4	9.1	6	8	10
	Delay in τ (ps)	3.3	4.3	5.0	6.5	7.7	8.6	4.8 (33.5)	5.5 (38.5)	6.0 (42.2)

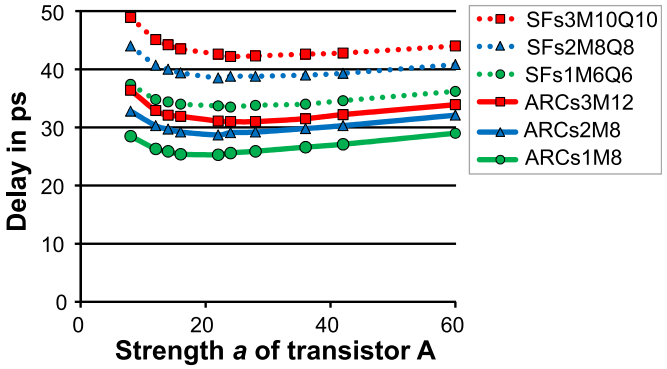

Fig. 2. Simulation sweeps for the ARC and SF arbiters, showing the uncontested grant delay as a function of drive strength a and stepup s under otherwise fixed drive strengths $b, n, p_1^{ARC}, p_1^{SF}, p_2, p_3, p_4 = 12, 4, 1, 2, 4, 1, 2$. Each of the six delay graphs belongs to a specific arbiter and stepup, and uses fixed best-m and best-q values that gave the least delay encountered in the collection of a-m-q sweeps simulated for this particular arbiter and stepup.

nominal input stepup of 3. We set the drive strengths of keeper transistors to provide small but similar loads on related signals in both arbiters. The exact drive strengths are specified in the caption of Table II.

Fig. 2 shows the uncontested grant delay as a function of drive strength a and stepup s. Specifically, Fig. 2 shows three delay graphs per arbiter, one for each stepup s of 1, 2, and 3. Each graph plots the uncontested grant delay simulated for the specific arbiter and stepup. The name of each graph indicates the specific arbiter design, stepup, and best m and q (if present) with the least delay encountered in the entire a-m-q sweep for this arbiter and stepup. For instance, *ARCs1M8*, the name of the graph with the best—i.e., least — delay for the ARC arbiter simulated for the indicated a-range and stepup s of 1, uses a drive strength m of 8.

The graphs in Fig. 2 are relatively flat near their minimum delay. So are the graphs — omitted here — showing the uncontested grant delay as a function of m or q versus s. Clearly, there is flexibility in sizing transistors A, M, and Q without jeopardizing the uncontested grant delay.

Table II captures the best drive strengths a, m, q and the least uncontested grant delay for each delay graph in Fig. 2. Table II also shows the corresponding results for the mathematical analysis *with* and *without* parasitics, with γ_N and γ_P set to $\frac{1}{3}$ and $\frac{2}{3}$, to match the drive strength to the input capacitance of 90nm nMOS and pMOS transistors. The simulated delay values are measured in *picoseconds (ps)*. The analyzed delay values are given in τ, the delay of an inverter driving an identical inverter. Simulations with 90nm CMOS inverter rings indicate that τ is approximately 7 picoseconds.

Bearing in mind that the exact transistor source, drain, and channel loads depend on the layout, and that the strength versus delay graphs are rather flat around their minimum delay, the key message to take away from Table II is that the calculated and simulated drive strengths point in the same direction, and that fine-tuning requires more in-depth layout and process knowledge. The delay results indicate that the ARC arbiter has a better—i.e., smaller — uncontested grant delay than the SF arbiter. In the mathematical analysis the ARC arbiter is 12–15% faster *without* considering parasitics, and 19–20% for the formulas *with* parasitics. In the simulation the ARC arbiter is approximately 25% faster.

VI. LOGICAL EFFORT AND RELEASE DELAY

Fig. 3(a) presents simulation results for the two arbiters using fixed near-optimal transistor strengths, and compares each arbiter's uncontested grant delay to the delay of NAND and inverter circuits. Note that the grant delays of the two arbiters are comparable to a NAND with similar stepup. Note too that the grant delays are relatively insensitive to the output load, represented by stepup s. The internal parasitic delay, p, dominates the uncontested grant delay.

The slope of each graph is an indication of the logical effort, g, of the circuitry [12]. The uncontested grant circuitry has exceptionally low logical effort, $g = 0.5$ for the ARC arbiter and $g = 0.7$ for the SF arbiter — lower even than the logical effort of the inverter circuit. The delay and logical effort of an uncontested grant are so good, i.e., low, only because both arbiters use very small keeper transistors.

The reverse effect of small keeper transistors is that their drive strengths are weak, making the paths that pass through them slow, thus making the arbiter release delay graphs the upper graphs in Fig. 3(a).

At negligible cost to the grant delay and small cost to the arbiter's input capacitance, we can reduce the release delay by resizing just one keeper transistor: P_3 in the ARC arbiter and P_1 in the SF arbiter. Because the side effects are small, we can do this independently from sizing for least uncontested grant delay. Fig. 3(b) shows the reduced release delays with 1, 2 and 3 times larger p_3 and p_1.

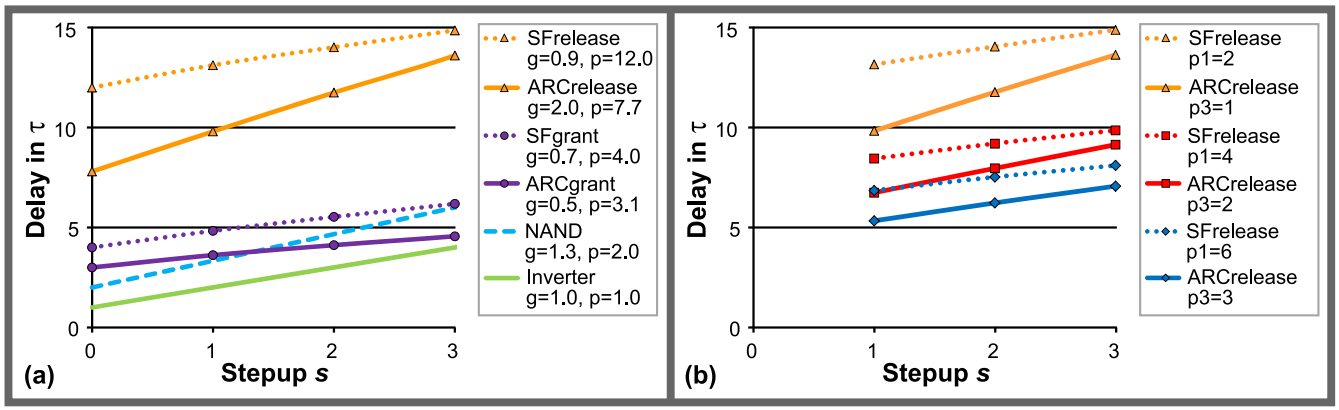

Fig. 3. (a) Uncontested ARC and SF arbiter grant and release delays, simulated for a stepup s of 1, 2, and 3, with fixed transistor parameters as specified in Table II and fixed near-optimal a, m, and q strengths of 24, 8, and 6. For each delay graph we indicate its logical effort, g, and internal parasitic delay, p. Note the relatively low delays and exceptionally good—i.e., low — logical effort of the uncontested grant delay graphs. The contrasting relatively high release delays in (a) can be reduced with a separate optimization step at negligible cost to the grant delays by strengthening keeper transistor P_3 in the ARC arbiter and P_1 in the SF arbiter. The graphs in (b) show the results of resizing these two keeper transistors by a factor of 1, 2, and 3, respectively.

VII. Conclusion

The first question in sizing is what delays to consider. We minimize the uncontested grant delay to cover the vast majority of mutual exclusion, or arbitration, cases where contention is absent. The second and harder question is how to partition the delay analysis. For instance, sizing end-of-metastability detection transistor M in Fig. 1(a) for least uncontested grant delay is a compromise between (1) the load M presents to the NAND gates that drive it and (2) M's ability to drive the arbiter output. A stronger M speeds driving the output, but a stronger M is also harder to drive and therefore retards the NAND gates. Partitioning the grant delay path serially over multiple paths addresses such mutual dependencies gracefully, and works because the time it takes to fill a combination of loads with charge is the sum of the times it would take to fill each load separately.

As for most CMOS circuits, relatively large changes in transistor strengths result in relatively small changes in each arbiter's uncontested grant delay, as illustrated in Fig. 2 for transistor A's strength a. We used this insensitivity for the benefit of reducing the release delays, as shown in Fig. 3(b).

The uncontested grant delay does depend, however, on the arbiter's output load, as illustrated by the same Fig. 2 and by Fig. 3(a). Arbiters have less delay when lightly loaded. Combine them with simple inverters if you need more gain.

The ARC arbiter's main advantage is that it avoids an entire stage of logic to make room for inverters that have higher gain than the second inverting stage of the SF arbiter. In addition, the ARC arbiter also uses the better gain of nMOS transistors to detect the end of metastability.

The mathematical analysis and supporting simulation results presented in this brief offer the following insights. Cross-coupled NAND transistor A in Fig. 1 best be stronger than its series transistor B because A follows a stage of gain whereas B loads the input directly. End-of-metastability detection transistor M best be intermediate in strength between the NAND drive of series pair A and B and the output load of the arbiter. Transistor Q, which plays a role in the second stage of the SF arbiter, best be of equal strength or slightly weaker than transistor M because Q has parallel support from P_1 to drive the output load in series with M.

The mutual exclusion circuits presented here involve only a handful of transistors. Because their area and delay are tiny compared to the area and delay taken by latches or flipflops, we feel free to use them profusely in our designs. We include its own mutual exclusion circuit (MrGO) in each and every self-timed action to provide reliable interruption anywhere [8].

References

[1] D. Hand *et al.*, "Blade—A timing violation resilient asynchronous template," in *Proc. IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, 2015, pp. 21–28.

[2] J. Delvaux and I. Verbauwhede, "Fault injection modeling attacks on 65 nm arbiter and RO sum PUFs via environmental changes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1701–1713, Jun. 2014.

[3] S. Jackson and R. Manohar, "Gradual synchronization," in *Proc. IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, 2016, pp. 29–36.

[4] D. Kinniment, *He Who Hesitates Is Lost: Decisions and Free Will in Men and Machines*, School Elect. Electron. Comput. Eng., Newcastle Univ., Tyne, U.K., 2011.

[5] D. Kinniment, *Synchronization and Arbitration in Digital Systems*. Hoboken, NJ, USA: Wiley, 2007.

[6] J. W. Lee *et al.*, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Proc. IEEE Symp. VLSI Circuits (VLSIC)*, 2004, pp. 176–179.

[7] M. Roncken *et al.*, "How to think about self-timed systems," in *Proc. Asilomar Conf. Signals Syst. Comput.*, 2017, pp. 1597–1604.

[8] M. Roncken *et al.*, "Naturalized communication and testing," in *Proc. IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, 2015, pp. 77–84.

[9] F. U. Rosenberger, C. E. Molnar, T. J. Chaney, and T.-P. Fang, "*Q*-modules: Internally clocked delay-insensitive modules," *IEEE Trans. Comput.*, vol. 37, no. 9, pp. 1005–1018, Sep. 1988.

[10] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, C. Mead and L. Conway, Eds. Reading, MA, USA: Addison-Wesley, 1980, pp. 218–262.

[11] J. Sparsø and S. Furber, Eds., *Principles of Asynchronous Circuit Design: A Systems Perspective*. Boston, MA, USA: Kluwer Acad., 2001.

[12] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA, USA: Morgan Kaufmann, 1999.

[13] Y. Zhang *et al.*, "Testable MUTEX design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 8, pp. 1188–1199, Aug. 2016.