

Analytical Modeling of Asynchronous Event-Driven Readout Architectures Using Queueing Theory

Dominik S. Górni¹ and Grzegorz W. Deptuch

AGH University of Krakow,

Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering,

Department of Metrology and Electronics,

al. A. Mickiewicza 30, 30-059 Krakow, Poland

E-mail: dgorni@agh.edu.pl

ABSTRACT: Event-driven imagers and sensor arrays commonly employ asynchronous arbiter trees with a synchronous acknowledge to serialize requests. We present an analytical framework that models the root as an $M/D/1$ queue with deterministic quantum T and implements losses at the sources through one-slot gating. The admitted rate, loss probability, utilization, and mean sojourn time are coupled by self-consistent relations; a closed form for $\mathbb{E}[S_t]$ separates fixed path delay τ_0 from queueing effects. The framework matches post-layout results of a physical prototype over light to heavy traffic, reproducing saturation at $1/T$ and the observed latency growth, while classical $M/G/1/K$ and Engset-type abstractions diverge at higher occupancy. Because all relations are algebraic, they enable rapid sizing at design time, including the impact of partitioning into independent tiles: reducing fan-in lowers arbitration depth and τ_0 , decreases loss, and improves latency at fixed T , with throughput adding across tiles. The model thereby links architectural parameters to performance metrics and supports selection of acknowledge period, tiling, and link count under practical constraints.

KEYWORDS: Front-end electronics for detector readout, Electronic detector readout concepts (solid-state), VLSI circuits, Analysis and statistical methods

¹Corresponding author.

Contents

1	Introduction	1
2	Working Principle of Event-Driven Readout with Arbiter Tree	3
2.1	Asynchronous request/acknowledge handshake	3
2.2	Cell-level arbitration with Seitz mutexes	4
2.3	Tree composition, path clearing, and fairness	4
2.4	Example waveforms	5
3	Analytical Model	5
3.1	Modeling assumptions and notation	6
3.2	Timing decomposition with tree delays	7
3.3	Per-source one-slot gating and admitted rate	7
3.4	Root queue: M/D/1 core	7
3.5	Closed-form solution	8
3.6	Stability and asymptotics	8
3.7	Summary of computable outputs	9
4	Model validation	9
4.1	Model parameters	9
4.2	Post-layout reference data	10
4.3	Modeling results	12
5	Using the Model During System Design	13
6	Summary and Conclusions	14

1 Introduction

Modern radiation detectors, particle trackers and intelligent sensor arrays increasingly rely on *event-driven* readout to achieve high throughput at low power while preserving precise timing information [1, 2]. In an event-driven system, pixels (or, more generally, any sensing channels) request access only when any activity occurs, and the readout architecture arbitrates among the outstanding requests without imposing a global frame [3]. This contrasts with frame-based schemes, in which the entire matrix is periodically sampled at a fixed rate. Even some pseudo-event-driven schemes that use only combinational logic for arbitration, like Address-Encoder Readout-Decoder (AERD) need to freeze the matrix state over coarse capture windows (typically 2–10 μ s) to avoid dynamic switching [4]. In a truly event-driven architecture, exemplified by EDWARD (Event-Driven With Access and Reset Decoder), a fully asynchronous binary-tree network of arbiters is

used to grant bus access to requesting pixels, while a synchronous acknowledge clock provides the data-transfer quantum that allows full synchronization with the external data-acquisition system [5]. When a validated hit occurs, the corresponding pixel autonomously requests the shared bus – if it wins arbitration, it transmits its address and optional payload, receives an acknowledge, and self-resets, readying the system for the next event.

An analytical description of such event-driven systems is essential at design time. Given project constraints, expected and peak per-channel rates, aggregate load, acceptable loss (pileup), and target timing resolution, an analytical model lets the designer quickly explore architectural trade-offs, including: (i) the number of independent arbitration trees (and their fan-in), (ii) the number of parallel outputs and serialization speed, (iii) the acknowledge period T and its distribution of delays along the tree, and (iv) buffering policies at the pixel and system levels. With a predictive model, one can select parameters that avoid saturation over the operating conditions while meeting timing-resolution requirements, rather than relying on time-consuming end-to-end simulations alone.

Timing resolution is a first-order benefit of true event-driven readout: every accepted hit can be time-stamped at the readout boundary. However, the achievable resolution is not solely set by the local time-stamp circuit – it also depends on the *service (sojourn) time* from event occurrence to completion of readout, which includes (i) request/acknowledge propagation along the arbitration tree and (ii) any queueing delay when multiple sources contend. Properly choosing T (the acknowledge quantum), the number of trees/outputs and the arbitration depth bound this service time and, therefore, the effective time-stamp uncertainty. This is fundamentally different from frame-based or matrix-freeze approaches that limit resolution to the frame interval regardless of instantaneous activity. With appropriately chosen event-driven parameters, sub-microsecond and often tens-of-nanoseconds, effective timing can be achieved.

Prior work on Address-Event Representation (AER) architectures analyzed timing mainly from the *arbitration/bus* perspective rather than through a single centralized queue. In particular, [6] quantified the latency and temporal dispersion on the arbiter-tree/bus links under burst-ensemble traffic and derived bandwidth conditions to preserve spike-timing precision, while [7] treated AER as a traffic/queueing problem at the encoder/arbiter chain to estimate latency, queueing delay, and occupancy. In short, AER timing has been studied, but *system-level queueing theory for event-driven architecture has not been widely or successfully applied*, and existing analyses emphasize *distributed contention* rather than a monolithic buffer.

Building on that background, we previously experimented with canonical queueing-model abstractions known from Queueing Theory but found they *fail at medium-to-high load*. Specifically, $M/G/1/K^1$ (finite buffer with Poisson input) [9] places losses at a *central* queue and folds path delays into the server time, and $M/G/1//N$ (Engset; finite population, also denoted as $M/G/1/K/N$, where $K = \infty$) [10] suppresses arrivals when many sources are busy. Neither captures the *per-source one-slot blocking* that dominates in pixelated structures, such as radiation detectors, where each pixel can hold only one pending request and the server completes at most one job per acknowledge period T whenever work is present. Consequently, these models agree with measurements at low rates but *diverge near saturation*, mispredicting pile-up, utilization, and sojourn time.

Motivated by these shortcomings, this work introduces and validates a *tractable, physics-*

¹Kendall’s notation [8]

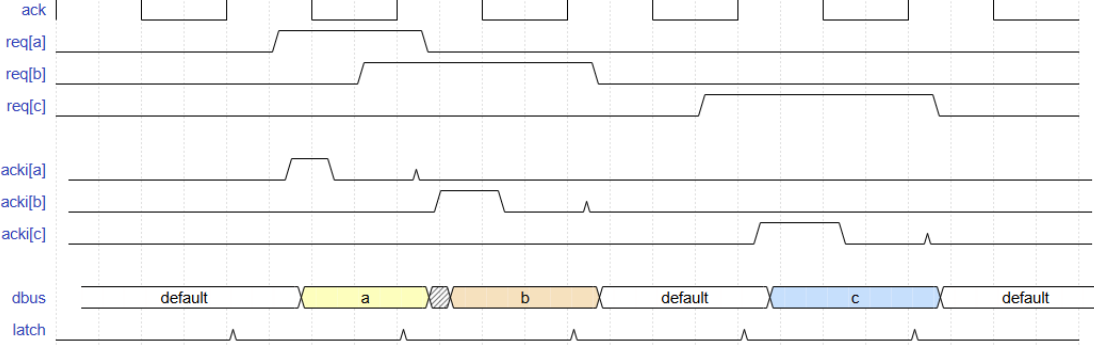


Figure 1: Asynchronous request/acknowledge handshake in the EDWARD architecture. Three pixels ($a - c$) issue requests, and arbitration gates the global acknowledge ack to produce per-winner signals $ack_i[k]$. Only the granted pixel observes the two same-polarity edges required to start and complete a transfer. The data bus ($dbus$) carries each pixel’s payload in non-overlapping windows ($a - c$). Unassigned intervals (*default*) correspond to idle bus states, while hatched regions denote short bus turnaround times. The $latch$ marks sampling instants used by the serializer – one completion is available per server quantum T .

faithful model for an event-driven readout system based on the EDWARD architecture. We model the readout core as an $M/D/1$ server with deterministic quantum T , and describe the input via *per-source one-slot gating* that thins the admitted Poisson arrivals without invoking a central buffer. From this construction, we derive closed-form expressions for (i) mean sojourn time as a function of T , arbitration depth/delays, and aggregate load; (ii) probability of pile-up (loss) per-source; (iii) utilization and throughput; and (iv) design guidance for selecting the number of trees/outputs and serialization speed to avoid saturation. Analytical results and simulations jointly explain why other models fail at higher occupancy and quantify the timing-resolution gains achievable with properly dimensioned event-driven designs like EDWARD.

2 Working Principle of Event-Driven Readout with Arbiter Tree

2.1 Asynchronous request/acknowledge handshake

In the EDWARD readout architecture, each pixel asserts a *request* ($req[k]$) when a validated event (e.g., a particle hit) occurs. An arbitration tree grants exclusive access to the shared readout bus to at most one requester at a time (see Figure 1). Bus access is transacted by a two-edge handshake using a global acknowledge (ack) that is distributed back to the pixels through the arbitration tree. The tree behaves like a clock-gating network and delivers a per-winner gated acknowledge, $ack_i[k]$. The first active edge of $ack_i[k]$ begins the transfer by enabling the drivers from the selected pixel that drive data onto the data bus. Then the second active edge, of the same polarity, completes the transfer, causes the pixel to self-reset, and latches the data in the serializer located in the periphery. While requests are asynchronous with respect to the acknowledge timing, the downstream server (bus + serializer) operates in fixed quanta of duration T , completing at most one transfer per period whenever work (a request) is present.

2.2 Cell-level arbitration with Seitz mutexes

Each binary arbiter *cell* is built from Seitz RS-latch based mutual-exclusion elements (mutexes) [11] arranged to avoid glitches and races when *ack* is in flight. Functionally, the cell contains *three* arbiters, as shown in Figure 2:

1. A request arbiter that resolves the two child requests (*req0*, *req1*) into one-hot local requests (*freq0*, *freq1*).
2. Two acknowledge interlock arbiters (one per child) that qualify the local request with the parent acknowledge to ensure that the cell does not change state while *ack* is asserted inside the cell.
3. The combination guarantees a *two-step* update: (i) decide the winner on requests; (ii) release and retime updates only after *ack* has been observed low again at the cell, thereby preventing hazards and race conditions on the upward request and downward acknowledge paths.

This organization enforces local First-Come, First-Served (FCFS) arbitration: once a child's request state is captured, the opposite child is blocked until the handshake completes. When the winning pixel resets on the second *ack_i* edge, its request is cleared, allowing a pending child request to propagate but only after *ack* is withdrawn by the parent.

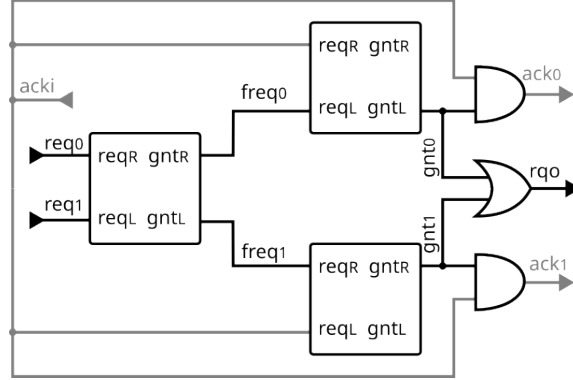


Figure 2: Binary arbiter cell based on Seitz mutexes. Left: child requests (*req0*, *req1*) are resolved into one-hot outputs (*freq0*, *freq1*). Right: per-branch acknowledge interlocks qualify local requests with the *ack* to prevent in-cell races or glitches. OR/AND networks form the upward propagated request and the gated acknowledges toward the children [5].

2.3 Tree composition, path clearing, and fairness

Cells are organized into a binary tree whose leaves connect to pixels and whose root connects to the acknowledge generator. The two-stage arbitration tree is shown in Figure 3. A decision propagates upward, stage by stage, toward the root, after which the *ack* signal is back-propagated downward along the single selected path using the arbitration mechanism described earlier.

Path clearing is triggered by request resets. Upon completion, the *leaf* withdraws its request (self-resetting on the second *ack_i* edge). Each cell along the winning path then clears its output

request, if only momentarily, in the case where a second leaf request is active – before passing control to the next stage. As a result, if a sibling branch is actively requesting, it is immediately revealed to the higher level and becomes eligible to win arbitration. This request withdrawal propagates upward like a domino toward the root, which in turn removes the gating condition for ack along that path (i.e., ack is *withdrawn* from the path that has just completed).

In aggregate, this yields:

- No global FCFS guarantee (decisions are made on a *per-cell* basis).
- Practical starvation avoidance via request-driven path clearing: a branch that just won tends to defer to its sibling until that sibling is serviced, approximating round-robin among active sub-trees without global state.

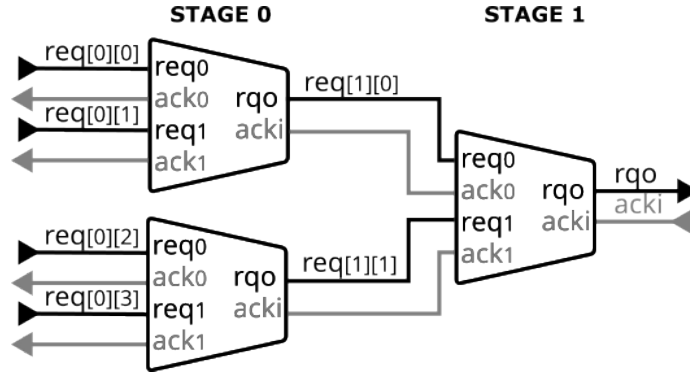


Figure 3: Two-stage arbitration tree. Stage 0 cells arbitrate leaf pairs, and Stage 1 arbitrates between their winners, forming the complete request/acknowledge hierarchy.

2.4 Example waveforms

Figure 4 illustrates the timing of a four-leaf tree under overlapping requests. The green numbers 1–4 mark successive *request arrivals*; due to local FCFS and the path clearing mechanism, the *actual service order* is 1 → 3 → 2 → 4. Orange asterisks mark the instants when pixel actions are triggered (latch on first acki edge and self-reset on the second). The global ack is periodic with quantum T , so at most one completion occurs per T . The acknowledge path is omitted on the waveforms for clarity but in reality its effect is visible through the gated acki [k] pulses that only appear on the winner’s branch.

It is important to note that there is *no dead time between consecutive readouts* – within the same ack period, the system completes the readout from one pixel and immediately begins servicing another, provided that a new request is pending.

3 Analytical Model

This section develops a tractable and experimentally validated analytical model for event-driven readout with arbiter trees. The fundamental observation is that the *server side* of the system (the

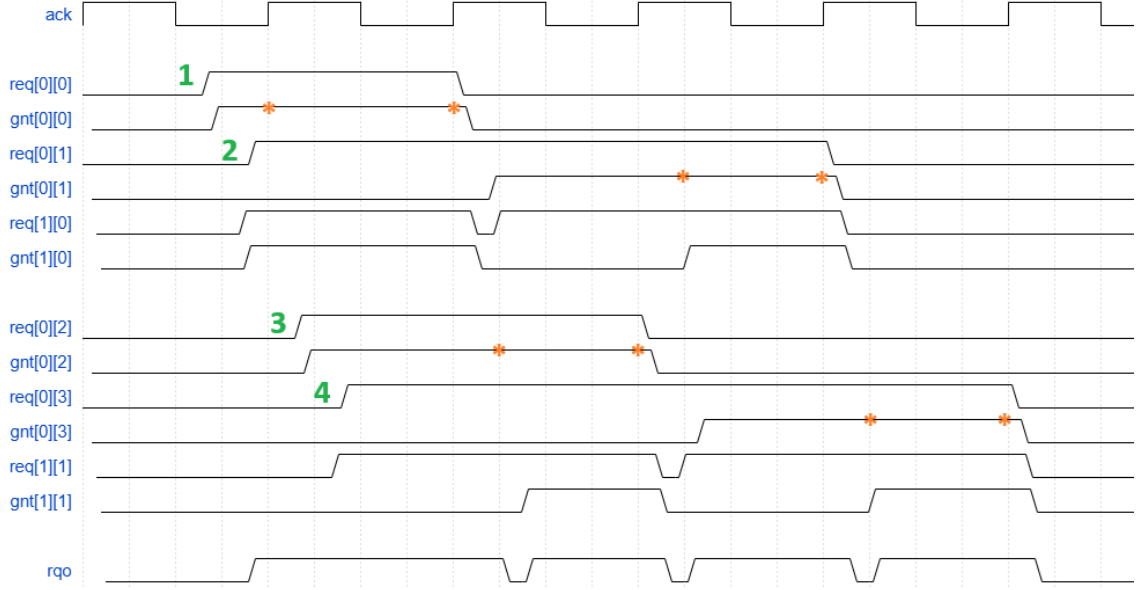


Figure 4: Timing of a four-leaf arbiter tree under overlapping requests. Labels **1–4** denote *request arrivals*. The actual service sequence is **1–3–2–4**. Orange asterisks mark pixel-level actions triggered by the acknowledge edges.

shared bus and serializer) completes exactly one request per acknowledge period T whenever any work is present. The service process is therefore *deterministic* at the granularity of T . Conversely, event losses (pileup) occur *locally at each source*, since every source can buffer at most one pending request. These properties naturally lead to an $M/D/1$ queue fed by a *thinned* Poisson stream that accounts for the per-source one-slot gating.

3.1 Modeling assumptions and notation

We introduce the following symbols:

- N : number of independent sources (pixels or channels);
- λ : per-source Poisson mean arrival rate (events/s);
- T : acknowledge period, corresponding to the deterministic server quantum;
- $L = \lceil \log_2 N \rceil$: number of arbitration stages in the binary tree;
- t_1, t_2 : mean per-stage forward (req-up) and backward (ack-down) delays;
- τ_g : fixed logic overhead;
- $\tau_0 = L(t_1 + t_2) + \tau_g$: path-dependent propagation delay from leaf to root and back;
- $U \sim \text{Unif}[T/2, 3T/2]$: alignment jitter arising from the asynchronous phase between an event and the acknowledge clock edges;
- S : server service time.

The total event-to-completion (sojourn) time observed at a source is therefore

$$S_t = \tau_0 + U + W_q, \quad (3.1)$$

where W_q is the queueing delay at the root (in multiples of T), and the server service time itself is deterministic, $S \equiv T$.

3.2 Timing decomposition with tree delays

Propagation delays in the arbitration tree contribute a fixed overhead,

$$\tau_0 = L(t_1 + t_2) + \tau_g, \quad (3.2)$$

while asynchronous timing relative to the global acknowledge clock introduces a uniformly distributed jitter $U \sim \text{Unif}[T/2, 3T/2]$. For an isolated request (no contention), the event-to-completion time simplifies to

$$S_t = \tau_0 + U, \quad (3.3)$$

yielding the bounds $S_{t,\min} = \tau_0 + T/2$, $S_{t,\max} = \tau_0 + 3T/2$, and the mean value $\mathbb{E}[S_t] = \tau_0 + T$. These single-pixel measurements provide direct experimental calibration of τ_0 .

3.3 Per-source one-slot gating and admitted rate

Each source can store only one pending request – new hits that occur while a request is awaiting acknowledgement are irreversibly lost. According to the *Poisson Arrivals See Time Averages* (PASTA) [12] principle, the fraction of arrivals that find the source busy equals the fraction of time the source is busy. This leads to

$$P_{\text{loss}} = \frac{\lambda \mathbb{E}[S_t]}{1 + \lambda \mathbb{E}[S_t]}, \quad 1 - P_{\text{loss}} = \frac{1}{1 + \lambda \mathbb{E}[S_t]}, \quad (3.4)$$

where P_{loss} is the per-source pileup probability. Summing over all sources gives the total admitted rate at the root:

$$\Lambda = N\lambda(1 - P_{\text{loss}}) = \frac{N\lambda}{1 + \lambda \mathbb{E}[S_t]}, \quad (3.5)$$

which can be treated as effectively having the Poisson nature for large N due to the weak cross-correlation between sources.

3.4 Root queue: M/D/1 core

Given the admitted rate Λ and the deterministic service time T , the root behaves as an $M/D/1$ queue. Let $\rho = \Lambda T$ denote the utilization. For a general $M/G/1$ system under FCFS scheduling, the mean waiting time is

$$\mathbb{E}[W_q] = \frac{\Lambda \mathbb{E}[S^2]}{2(1 - \rho)}. \quad (3.6)$$

With deterministic service $S \equiv T$,

$$\mathbb{E}[W_q] = \frac{\Lambda T^2}{2(1 - \Lambda T)} = \frac{\rho T}{2(1 - \rho)}. \quad (3.7)$$

Combining (3.1) and (3.7) and noting $\mathbb{E}[U] = T$, we obtain

$$\mathbb{E}[S_t] = \tau_0 + T + \frac{\Lambda T^2}{2(1 - \Lambda T)}. \quad (3.8)$$

Equations (3.5)–(3.8) form a *self-consistent system* linking the mean sojourn time $\mathbb{E}[S_t]$, the admitted rate Λ , the pileup probability P_{loss} , and the utilization ρ .

3.5 Closed-form solution

The coupled equations above can be solved algebraically to obtain an explicit, *closed-form* expression for $\mathbb{E}[S_t]$ without iterative numerical methods.

Let $a = N\lambda$, $b = T$, and $\tau = \tau_0$. Substituting (3.5) into (3.8) leads to a quadratic equation in a transformed variable $u = 1 + \lambda \mathbb{E}[S_t] - ab$:

$$u^2 - (1 - ab + \lambda(\tau + b))u - \frac{\lambda ab^2}{2} = 0. \quad (3.9)$$

The physically valid (positive) root is

$$u = \frac{C + \sqrt{C^2 + 2\lambda ab^2}}{2}, \quad C = 1 - ab + \lambda(\tau + b). \quad (3.10)$$

Finally,

$$\mathbb{E}[S_t] = \frac{u - 1 + ab}{\lambda}, \quad (3.11)$$

and the remaining quantities follow directly:

$$P_{loss} = \frac{\lambda \mathbb{E}[S_t]}{1 + \lambda \mathbb{E}[S_t]}, \quad \Lambda = \frac{N\lambda}{1 + \lambda \mathbb{E}[S_t]}, \quad \rho = \Lambda T. \quad (3.12)$$

In the next section, we will prove that this closed-form, self-consistent model accurately predicts experimental measurements across both light and heavy traffic regimes, providing a quantitative bridge between asynchronous arbitration and queueing-theoretic performance metrics.

3.6 Stability and asymptotics

The $M/D/1$ queue is stable provided that the effective service rate exceeds the admitted arrival rate, $\rho = \Lambda T < 1$. Substituting (3.5) gives

$$N\lambda T < 1 + \lambda \mathbb{E}[S_t]. \quad (3.13)$$

This inequality is inherently satisfied by the self-consistent solution of Eqs. (3.5)–(3.8), since $\mathbb{E}[S_t]$ increases with load in a way that limits Λ such that $\rho < 1$.

For **light traffic**, i.e., $\lambda \rightarrow 0$ (when arrivals are sparse), the system behaves as a collection of independent sources. The total admitted rate is $\Lambda \approx N\lambda$, the utilization grows linearly with the rate, and the mean sojourn time converges to the single-source timing:

$$S_{t,\min} = \tau_0 + \frac{T}{2}, \quad S_{t,\max} = \tau_0 + \frac{3T}{2}, \quad \bar{S}_t = \tau_0 + T. \quad (3.14)$$

These relations can directly calibrate the effective propagation delay τ_0 from one-pixel measurements.

For **heavy traffic i.e.**, $\lambda \rightarrow \infty$ as the input rate increases, the throughput saturates at one completion per acknowledge period: $\Lambda \rightarrow 1/T$, $\rho \rightarrow 1$, and nearly all sources hold active requests ($P_{loss} \rightarrow 1$). In this limit, the system behaves like a round-robin scheduler that services each of the N sources once per cycle. Consequently, the mean sojourn time per source does not diverge as in a conventional queue but remains bounded by the time needed to serve all sources once,

$$\boxed{\mathbb{E}[S_t]_{\max} \approx NT + \tau_0,} \quad (3.15)$$

which matches the observed saturation of S_t at high load. This finite upper bound reflects the fairness of the arbitration tree: each source is guaranteed service within approximately N acknowledge periods, avoiding unbounded queueing delays typical of centralized buffers. The bounded behavior ensures predictable latency even at full occupancy, a key property for architectural scaling.

3.7 Summary of computable outputs

Given the system parameters (N , λ , T), the fixed delay is obtained either from experimental calibration using (3.3) or (3.14) or be estimated based on technology timing using (3.2). Solving Eqs. (3.9)–(3.11) yields the mean sojourn time $\mathbb{E}[S_t]$, from which the loss probability P_{loss} , the admitted rate Λ , and the utilization ρ follow via (3.12). These closed-form relations enable rapid exploration of design trade-offs (tree size, serialization rate, acknowledge period) while reproducing the observed system behavior across a wide range of loads.

4 Model validation

The self-consistent $M/D/1$ model with per-source admission control was validated against (i) post-layout simulations of an EDWARD-class prototype [13] and (ii) software-based discrete-event simulations of the asynchronous arbiter tree. For comparison, two classical abstractions were also included: the $M/G/1/K$ model (finite central buffer with $K=N$) and the Engset $M/G/1//N$ model (finite population). Additionally, an analytical $M/G/1$ variant was evaluated assuming a uniform service-time distribution over $[\tau_0 + T/2, \tau_0 + 3T/2]$, while preserving the same admission rule as the $M/D/1$ core.

4.1 Model parameters

To ensure comparability with post-layout simulation data, the analytical and simulated models were parameterized according to the prototype specifications. For the sake of reference, the prototype comprises $N = 1,024$ pixels and operates with an external serialization clock of 250 MHz, which is internally divided by 14 to generate the acknowledge and data-latch signals in the serializer. This configuration yields a deterministic service quantum of $T = 56$ ns. The prototype allows pixel-level activation control and, by enabling a single pixel, the intrinsic service latency τ_0 can be extracted from relatively fast post-layout simulations. Figure 5 presents the distribution of the service latency S_t for an isolated pixel. From these results, the intrinsic request/acknowledge path delay was estimated as $\tau_0 = 6.05$ ns.

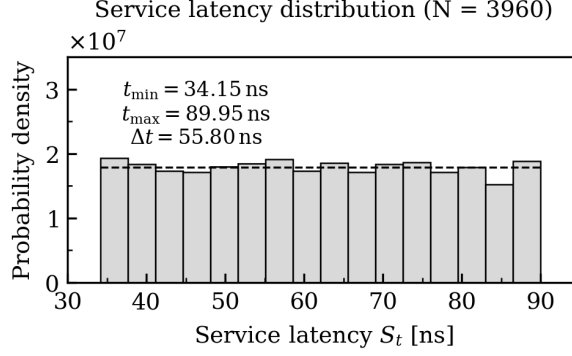


Figure 5: Distribution of the service latency S_t for a single isolated pixel. The histogram’s position and width provide estimates of the intrinsic request/acknowledge path delay τ_0 and timing jitter, independent of inter-pixel contention [14].

4.2 Post-layout reference data

To obtain reference data for model validation, three full-scale post-layout simulations of the EDWARD prototype were performed under distinct input-rate conditions [14]: low, medium, and high. Each scenario corresponds to a different per-pixel event-generation rate λ and an effective total arrival rate Λ^* , calculated as a sum of all pixel rates in the absence of contention.

In the *low-rate* scenario, the per-pixel rate was $\lambda = 948.2 \text{ s}^{-1}$, giving an aggregate $\Lambda^* = 970.9 \text{ ks}^{-1}$. The corresponding mean inter-arrival time $1/\Lambda^* \approx 1.02 \mu\text{s}$ was much larger than the service quantum $T = 56 \text{ ns}$, i.e., $1/\Lambda^* \gg T$. Under such sparse traffic, almost all pixel requests were served immediately by the next available acknowledge pulse, with minimal queueing effects.

In the *medium-rate* scenario, $\lambda = 15,169.2 \text{ s}^{-1}$ and $\Lambda^* = 15.53 \text{ Ms}^{-1}$, yielding $1/\Lambda^* \approx 64 \text{ ns}$, which becomes comparable to T . In this intermediate regime, contention between pixels starts to play a significant role. The average service delay increases as multiple requests compete for acknowledge, resulting in a broader latency distribution with occasional quantized delays corresponding to integer multiples of T .

In the *high-rate* scenario, the pixel-level rate reached $\lambda = 239.98 \text{ ks}^{-1}$, giving a total $\Lambda^* = 24.57 \text{ Ms}^{-1}$. Here, $1/\Lambda^* \approx 40.7 \text{ ns} \ll T$, implying that new events arrive faster than acknowledges can be issued. In this saturation regime, nearly every pixel experiences waiting time, and the mean latency asymptotically approaches the full-matrix readout time – as it was a frame-by-frame acquisition process where each pixel must wait for all others to be serviced.

The recorded distributions of the service (sojourn) times for these three operating regimes are presented in Figure 6(A). The evolution of the distribution, from a narrow, nearly uniform shape at low λ to a right-skewed, quantized profile at high λ , clearly reflects the transition from sparse, asynchronous operation to contention-limited throughput.

Complementary insight is provided by the per-pixel maps shown in Figure 6(B–C). Panel (B) presents the spatial distribution of the mean service latency S_t across all 1,024 pixels for each input-rate regime. At low rates, the map is essentially uniform, confirming that all handshake paths within the arbiter tree contribute nearly identical propagation delays. Minor spatial variations ($<1 \text{ ns}$) can be attributed to systematic routing differences and transistor-level parasitics in the physical layout.

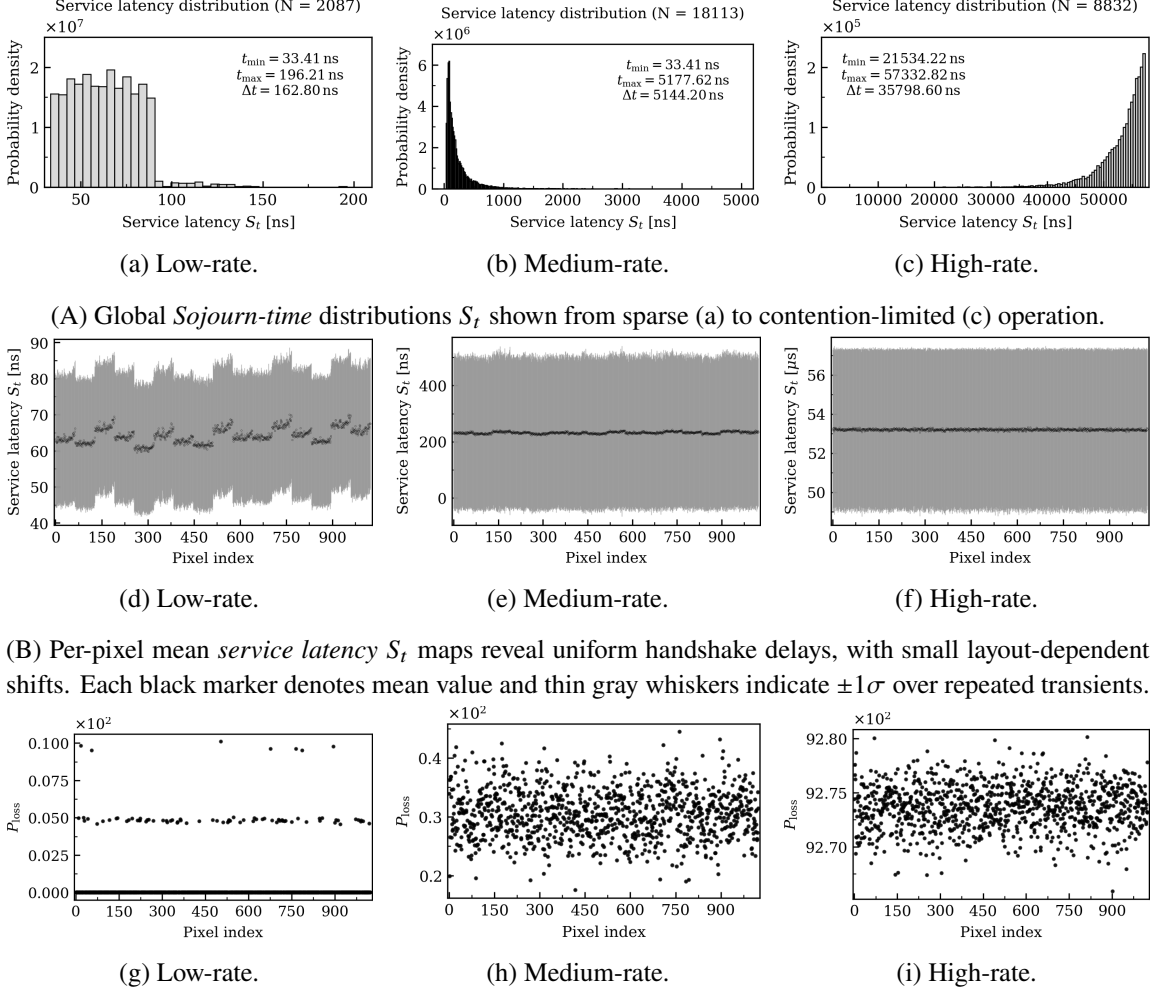


Figure 6: Comprehensive post-layout validation under three input-rate regimes [14].

As the input rate increases, these latency maps remain globally flat, indicating that contention is evenly distributed among the pixels and that the arbitration network operates without introducing geometric bias or priority artifacts.

Panel (C) illustrates the corresponding per-pixel pile-up probability P_{loss} , defined as the ratio of lost (not handled) requests to the total number of the generated events. As expected, the loss probability grows monotonically with λ : it is practically zero in the low-rate regime, reaches the sub-percent level at medium rates, and approaches unity in the high-rate case. The near-uniform spatial distribution of P_{loss} further confirms that the admission process depends solely on global load and not on pixel position or electrical distance within the matrix.

Table 1: Comparison of post-layout, simulation, and analytical models at three representative per-source rates ($N = 1,024$, $T = 56$ ns, $\tau_0 = 6.05$ ns).

λ [s ⁻¹]	Model	$\mathbb{E}[S_t]$ [s]	P_{loss}	ρ	Λ [s ⁻¹]
948	Post-layout	6.420×10^{-8}	0.000%	5.437%	9.710×10^5
	<i>M/D/1</i>	6.366×10^{-8}	0.006%	5.437%	9.709×10^5
	<i>M/G/1</i>	6.417×10^{-8}	0.006%	6.024%	9.709×10^5
	Simulation	6.403×10^{-8}	0.000%	5.347%	9.547×10^5
	<i>M/G/1/K</i>	6.403×10^{-8}	0.000%	5.856%	9.427×10^5
	<i>M/G/1//N</i>	6.485×10^{-8}	0.006%	6.077%	9.713×10^5
15,169	Post-layout	2.320×10^{-7}	0.300%	86.725%	1.549×10^7
	<i>M/D/1</i>	2.440×10^{-7}	0.369%	86.665%	1.548×10^7
	<i>M/G/1</i>	7.367×10^{-7}	1.105%	95.319%	1.536×10^7
	Simulation	2.451×10^{-7}	0.389%	86.889%	1.552×10^7
	<i>M/G/1/K</i>	8.436×10^{-7}	0.000%	96.076%	1.549×10^7
	<i>M/G/1//N</i>	6.323×10^{-7}	0.956%	95.713%	1.548×10^7
240,000	Post-layout	5.320×10^{-5}	92.700%	100.000%	1.786×10^7
	<i>M/D/1</i>	5.321×10^{-5}	92.737%	99.947%	1.785×10^7
	<i>M/G/1</i>	5.941×10^{-5}	93.445%	99.944%	1.611×10^7
	Simulation	5.316×10^{-5}	92.732%	99.999%	1.786×10^7
	<i>M/G/1/K</i>	6.346×10^{-5}	93.422%	100.000%	1.613×10^7
	<i>M/G/1//N</i>	5.912×10^{-5}	93.440%	100.000%	1.618×10^7

4.3 Modeling results

Table 1 reports mean sojourn time $\mathbb{E}[S_t]$, per-source pile-up probability P_{loss} , utilization ρ , and aggregate throughput Λ at three representative per-source rates. Figures 7(a–d) summarize the full-range trends.

At **low rate** all models agree that the system is lightly loaded, with $\rho \approx 5\text{--}6\%$ and essentially with zero loss. The prediction **M/D/1** matches the post-layout Λ and ρ to within $<0.01\%$ absolute and $\mathbb{E}[S_t]$ within $<1\%$. The software **simulation** is statistically coincident. Small systematic offsets in Λ for **M/G/1/K** and **M/G/1//N** reflect their different admission mechanisms (central blocking vs. finite-population suppression), not the source-level pile-up.

At **medium rate** contention becomes appreciable: ρ rises to ~ 0.87 in the post-layout data and **M/D/1**, with modest loss ($P_{\text{loss}} \approx 0.3\%\text{--}0.4\%$). Here **M/D/1** slightly *overestimates* delay: 2.44×10^{-7} vs. 2.32×10^{-7} s, by $\sim 5\%$ and P_{loss} by ~ 0.07 percentage points (pp), while still tracking Λ and ρ closely. This may be the result of limited simulation time of the post-layout run, especially when the software **Simulation** shows a comparable deviation. In contrast, the uniform-service **M/G/1** inflates both ρ and delay ($\mathbb{E}[S_t]$ is larger by a factor ~ 3), because it folds τ_0 and alignment into the server time. **M/G/1/K** shows negligible blocking at this $K=N$ (loss $\approx 0\%$) but still lengthens $\mathbb{E}[S_t]$ via central buffering. Engset **M/G/1//N** behaves similarly to **M/G/1** in delay

and utilization owing to finite-population suppression effects.

At **high rate** the system saturates as expected: $\Lambda \rightarrow 1/T \approx 1.786 \times 10^7 \text{ s}^{-1}$, $\rho \rightarrow 1$, and $P_{\text{loss}} \rightarrow 92.7\%$, with $\mathbb{E}[S_t] \approx 53.2 \mu\text{s}$. **M/D/1** and **Simulation** coincide with the post-layout data within the reported spread. Models that treat τ_0 as server occupancy (uniform-service **M/G/1**, **M/G/1/K**, and Engset **M/G/1//N**) underestimate throughput by $\sim 9\text{--}10\%$ and exhibit longer mean sojourn times, being consistent with their inflated effective service times and, for **M/G/1/K**, central blocking.

Across all regimes, the dominant behaviors are dictated by the architectural facts established in Section 3: (i) a *deterministic* server at the root (one completion per T when backlogged); (ii) *per-source one-slot* admission, which places loss at the sources rather than in a central queue; and (iii) τ_0 and phase alignment adds to sojourn time but does *not* consume root service. The **M/D/1** abstraction preserves all three, hence it remains quantitatively predictive from light to heavy traffic. By contrast, uniform-service **M/G/1**, **M/G/1/K**, and Engset **M/G/1//N** each violate at least one of these architectural constraints, leading to the observed overestimates in delay and shortfalls in saturation throughput.

5 Using the Model During System Design

The analytical framework developed in Section 3 can be employed to predict system performance under different architectural configurations. Because the relations are algebraic and self-consistent, they allow rapid evaluation of latency, loss probability, utilization, and throughput without time-domain simulation. The model is suitable for:

- preliminary dimensioning of readout architectures under specified limits on latency, loss, and throughput;
- analysis of the impact of acknowledge period T , arbitration depth, and intrinsic delay τ_0 ;
- selection of the number and size of independent subsystems (tiles) for a given I/O and power budget.

In large matrices, the readout system may be divided into c independent tiles, each containing N_j sources and its own arbiter and serializer. Substituting $(N, \lambda, T, \tau_0) \rightarrow (N_j, \lambda_j, T_j, \tau_{0,j})$ in the analytical relations yields tile-specific quantities $\mathbb{E}[S_t]_j$, $P_{\text{loss},j}$, and ρ_j . The total throughput is the sum of tile throughputs, $\Lambda_{\text{tot}} = \sum_j \Lambda_j$. Reducing the tile size decreases the arbitration depth $L_j = \lceil \log_2 N_j \rceil$ and consequently the propagation delay $\tau_{0,j}$, improving latency and reducing local pile-up at constant T_j . A typical use of the model involves:

1. defining the array size N , expected per-pixel rate λ , and acknowledge period T ;
2. estimating per-tile path delays $\tau_{0,j}$ from layout or measurement;
3. computing per-tile metrics $\mathbb{E}[S_t]_j$, $P_{\text{loss},j}$, and ρ_j using the closed-form relations;
4. verifying that all tiles satisfy target limits $P_{\text{loss},j} \leq P_{\text{loss}}^{\text{max}}$, $\mathbb{E}[S_t]_j \leq \mathbb{E}[S_t]^{\text{max}}$, and $\rho_j < 1$;
5. iterating on tile size N_j or acknowledge period T_j if constraints are not met.

These steps allow for an early architectural exploration before a detailed simulation.

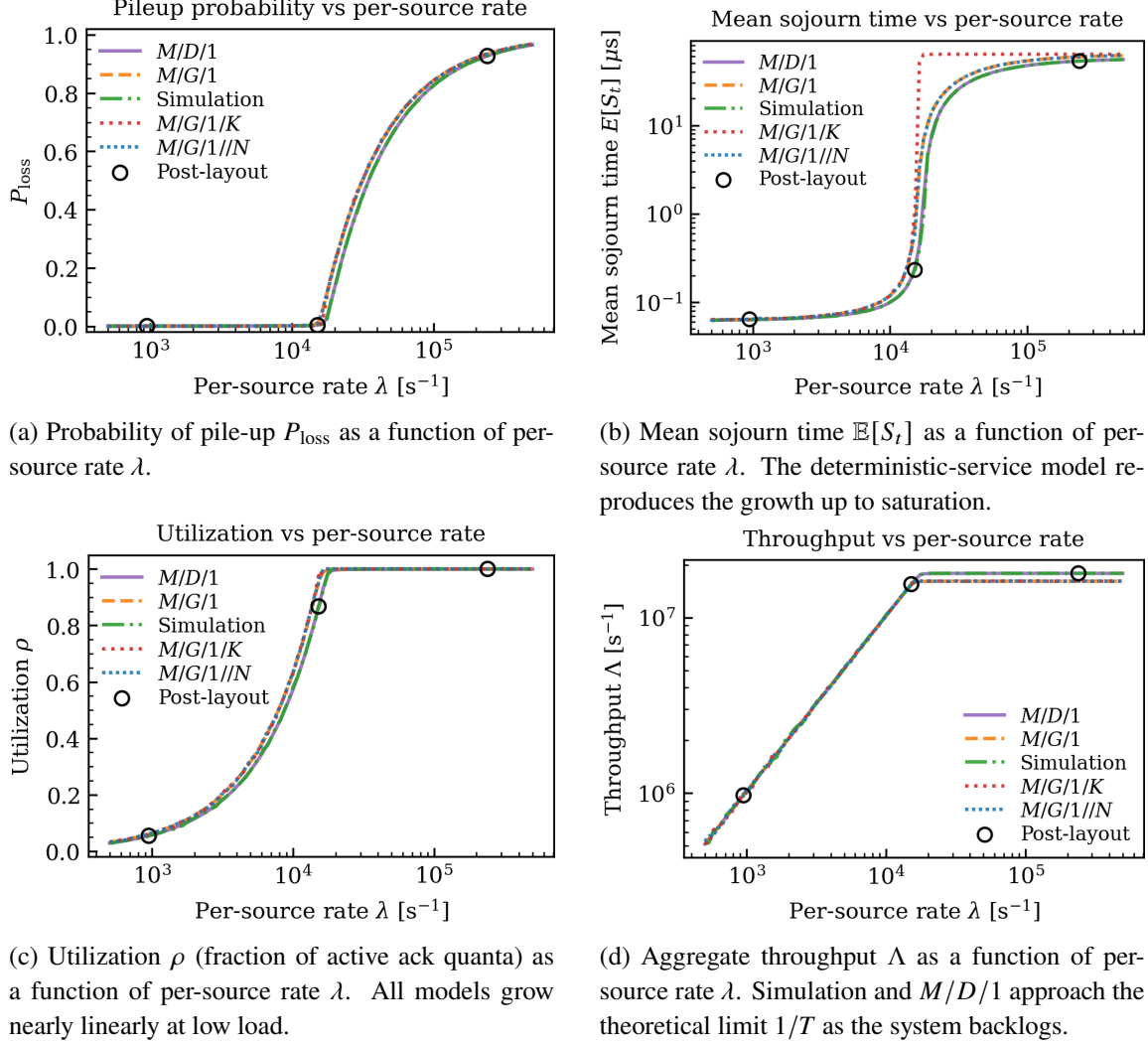


Figure 7: Comparison of analytical models, simulation, and measurements as a function of per-source rate λ .

6 Summary and Conclusions

This work formulated a tractable analytical description of arbiter-tree, event-driven readout with a synchronous acknowledge quantum. The readout core was modeled as an $M/D/1$ server (one completion per acknowledge period T when backlogged), while losses were placed at the sources via one-slot gating. The resulting self-consistent relations link admitted rate Λ , mean sojourn time $\mathbb{E}[S_t]$, utilization ρ , and loss probability P_{loss} , with a closed form for $\mathbb{E}[S_t]$ that separates fixed path delay τ_0 and queueing.

Validation against an EDWARD-class prototype and software simulations showed agreement across low, medium, and high load, including saturation at $1/T$. In contrast, abstractions that centralize blocking or fold τ_0 into service time deviated at moderate and high occupancy. The model provided quantitative guidance for design: tile-level partitioning reduces arbitration depth

and τ_0 , lowers latency and pile-up at fixed T , and scales aggregate throughput additively across tiles.

Future extensions include percentile metrics for S_t , explicit burst models beyond Poisson, non-uniform rate maps with tiling optimization, and co-design of serializer timing and arbitration for energy-latency trade-offs. The proposed formulation bridges physical implementation and high-level performance modeling, offering a practical analytical tool for future event-driven readout designs.

References

- [1] R. He, X.-Y. Niu, Y. Wang, H.-W. Liang, H.-B. Liu, Y. Tian et al., *Advances in nuclear detection and readout techniques*, *Nuclear Science and Techniques* **34** (2023) 205.
- [2] J.P. van Schayck, Y. Zhang, K. Knoops, P.J. Peters and R.B.G. Ravelli, *Integration of an event-driven timepix3 hybrid pixel detector into a cryo-em workflow*, *Microscopy and Microanalysis* **29** (2022) 352.
- [3] D. Gorni, G. Deptuch and S. Miryala, *Investigation of timing properties for an event driven with access and reset decoder readout architecture for a pixel array*, in *2022 17th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, pp. 113–116, 2022, DOI.
- [4] C. Yang, C. Feng, J. Liu, Y. Teng, S. Liu, Q. An et al., *A prototype readout system for the alpide pixel sensor*, *IEEE Transactions on Nuclear Science* **66** (2019) 1088.
- [5] D. Gorni, G. Deptuch, S. Miryala, D. Siddons, A. Kuczewski, A. Rumaiz et al., *Event driven readout architecture with non-priority arbitration for radiation detectors*, *Journal of Instrumentation* **17** (2022) C04027.
- [6] K. Boahen, *Point-to-point connectivity between neuromorphic chips using address events*, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* **47** (2000) 416.
- [7] C. Zamarreño-Ramos, *Modular and scalable implementation of aer neuromorphic systems*, 2011, <https://api.semanticscholar.org/CorpusID:193527387>.
- [8] U.N. Bhat, *An Introduction to Queueing Theory: Modeling and Analysis in Applications*, Statistics for Industry and Technology, Birkhäuser, Boston, MA, 2 ed. (2015), 10.1007/978-0-8176-8421-1.
- [9] J. Keilson and L.D. Servi, *The m/g/1/k blocking formula and its generalizations to state-dependent vacation systems and priority systems*, *Queueing Systems* **14** (1993) 111.
- [10] T. Takine, H. Takagi and T. Hasegawa, *Analysis of an m/g/1/k/n queue*, *Journal of Applied Probability* **30** (1993) 446–454.
- [11] C.L. Seitz, *Ideas about arbiters*, *Lambda* (1980) 10.
- [12] R.W. Wolff, *Poisson arrivals see time averages*, *Operations Research* **30** (1982) 223.
- [13] D. Gorni, G. Deptuch, P. Maj, S. Mandal and G. Pinaroli, *Event-driven readout development: testing of the edward65p1 chip with integrated event generators*, *Journal of Instrumentation* **20** (2025) C03009.
- [14] D.S. Gorni, G. Carini, G.W. Deptuch, A. Kuczewski, S. Mandal, G. Pinaroli et al., *Event-driven, arbitrated protocols implemented in integrated readout circuits for segmented sensors*, in *XII Front-End Electronics Workshop*, (Torino, Italy), INFN Torino, June, 2023, <https://agenda.infn.it/event/36206/contributions/202624/>.