PAPER

Event driven readout architecture with non-priority arbitration for radiation detectors

Published by IOP Publishing for Sissa Medialab

Received: *October 24, 2021*
Accepted: *February 10, 2022*
Published: *April 21, 2022*

Topical Workshop on Electronics for Particle Physics 2021
20–24 September, 2021
Online

Event driven readout architecture with non-priority arbitration for radiation detectors

D.S. Gorni,* G.W. Deptuch, S. Miryala, D.P. Siddons, A. Kuczewski, A.K. Rumaiz and G.A. Carini

Brookhaven National Laboratory, Upton, NY, U.S.A.

E-mail: dgorni@bnl.gov

Abstract: A novel event driven readout architecture, EDWARD (Event Driven with Access and Reset Decoder) architecture, for highly granular pixel detectors is presented. It incorporates, inter alia, an asynchronous arbitration tree based on Seitz' arbiters, removing the need for an imposed prioritization scheme. It also provides protection against glitches during readout. The system allows not only reading pixel activities, but also retrieving additional data, both analog and digital, from the pixels. A novel in-channel logic allows the entire readout process to be split into consecutive phases for additional flexibility. All operations are controlled by only one edge of the clock signal, seen as an acknowledge token, so there is no dead time between readouts.

Keywords: Electronic detector readout concepts (solid-state); Modular electronics; Detector control systems (detector and experiment monitoring and slow-control systems, architecture, hardware, algorithms, databases); Digital electronic circuits

*Corresponding author.

https://doi.org/10.1088/1748-0221/17/04/C04027

2022 JINST 17 C04027

Contents

1 Introduction

The growing number of data channels in radiation detectors requires robust, reliable, and preferably compact readout system on the chip to collect data from each channel and then transmit it out of the chip to communicate with acquisition systems. There, further processing and storing of data takes place. However, readout systems strive for optimal usage of the available bandwidth of the links. The number of channels, ranging from tens to hundreds of thousands, precludes the use of fixed links from the very beginning. Given the fact that the data from the channels are sparse, this would be far from an optimal solution anyway. For readout operations from this type of data source, a dynamic link, set up after a transmission request is received, is more interesting. That type of link utilizes a TDMA (Time Division Multiple Access) method for managing usage permission of the shared resources. Over the years different applications of such approach were presented. It is worth mentioning some of them, beginning with DPA (Data Push Architecture) [1] with metastability issues and quite complicated logic used to build the readout procedure pipeline. The next interesting approach is TPA (Token Passing Architecture) [2] which is still widely used to build systems of granting permission to use the shared resources in a form of a daisy chain, where the length of this chain can be a bottleneck in the larger chips. One other architecture that has gained recognition is AERD (Address-Encoder and Reset-Decoder) [3] with a hierarchical arbitration tree built based on priority logic. To overcome these issues and implement some additional useful features when, e.g. charge sharing comes into account, a new EDWARD architecture was designed and is presented in this document. In section 2 the EDWARD architecture is presented from a general overview to more detailed description in the following subsections. Section 3 summarizes the EDWARD architecture using an implementation example, along with simulation results that confirm the correctness of the idea.

2022 JINST 17 C04027

2 Structure of a system

The EDWARD architecture (figure 1) is adapted for the efficient transmission of data from a plurality of data sources (channels), that can be arranged in one-dimensional structures, two-dimensional structures, and/or any other form. This is thanks to the hierarchical design of the arbitration tree, mating in-channel logic and developed digital implementation flow that allows the system to be quickly adapted to the architecture at the chip design level. The logic provides a universal interface to the back-end circuitry in each channel. The interface further enables sending additional data, beyond merely the digital bits, e.g. analog voltage, and provides a reliable mechanism to prevent collisions between channels accessing the common data bus. Because of the synchronous nature of most of the data acquisition systems, synchronization circuitry is provided. The synchronization takes place at the global logic level, so there is no need to distribute any clock signal to all the channels. This saves area, power, and complexity of the system since no clock tree is required. All these features and improvements are advantageous for not only integrated readout of strip and pixel radiation detectors but also for building neuromorphic or other event-driven processing circuits.

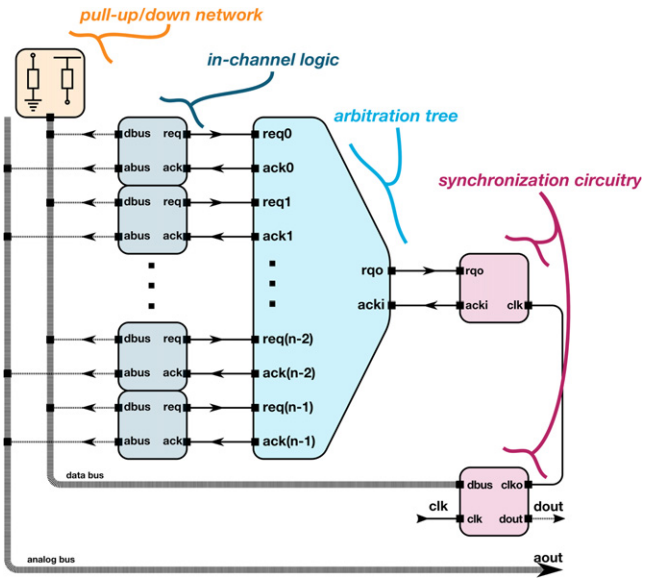

Figure 1. General structure of the EDWARD architecture with highlighted functional blocks.

2.1 In-channel logic

The in-channel logic (figure 2) is present in each channel and its function is to manage readout transactions between the channel and global peripheries. When the data ready flag *rdy* is set by the back-end electronics (peak found, ADC done flag, etc.), the controller block issues the read request *req* promptly.

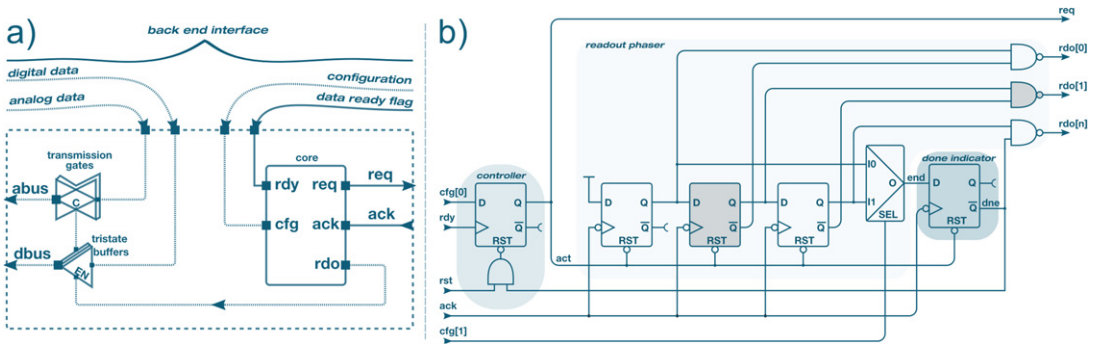

Figure 2. (a) In-channel logic internal structure, (b) in-channel core detailed structure.

During the active state on request line, the readout phaser block in the in-channel core is enabled and sensitive to transition on the channel acknowledge input *ack*. The transition to active state can be described as receiving an acknowledge token with assigned expiration time, after which the acknowledge input switches back to the inactive state. The first token initiates a readout transaction. A single transaction may consist of multiple readout phases in which different data packages may be transmitted sequentially and uninterrupted by requests from other channels. These data packages can contain subsequent chunks of data from a single channel or can be used to send additional information i.e. data from adjacent channels, e.g. the charge that was shared between the channels.

The maximum number of phases is determined by the number of flip-flops in the phaser chain and the chain length can be programmed by configuration *cfg*. The multibit readout signal *rdo* is used to switch between banks of tristate buffers and transmission gates. Only one bank is active at any given time thanks to the one-hot encoding provided by the output gates located in the phaser. After the last phase is processed, the end flag *end* is set. Consequently, the next token initiates the reset procedure for the in-channel logic. During the procedure, request is cleared and consequently acknowledge is withdrawn from the channel. Permission to use the shared data bus can be granted to a different requesting channel if there is one.

2.2 Pull-up/down network

The request output *rqo* from the arbitration tree is effectively the logical sum of requests from all channels. This signal, however, is not synchronized in any way with the incoming acknowledgement tokens. The request may come when the token is already expired or come too close to the expiration moment. As a result, the token is not able to trigger the transaction in the channel due to the duration of the triggering signal seen by the flip-flop in the in-channel logic core being too short. For this reason, a mechanism, that is not based on the output request, should be provided to distinguish between data derived from a channel and an empty state on the data bus. This mechanism has been implemented as a network of up and down pulls that delineate empty data. The pattern thus determined can then be discarded either on-chip by a peripheral circuit or off-chip in the acquisition system.

2022 JINST 17 C04027

2.3 Arbitration tree

The arbitration cells are used to build an arbitration tree as shown in figure 3. A single cell upon receiving read request signals *req[X]*, accepts it and creates a path from acknowledge token input *acki* to one of the acknowledge outputs *ack[X],* corresponding to the accepted request number. The arbitration cell also generates a request output which is sent to the next stage of arbitration, where the same process takes place. In that way a hierarchical structure is built and once the request is accepted in the top arbitration cell, the complete path for acknowledge token, from the top of the arbitration tree to the selected channel, is set and, if there is an active token, it is sent to the channel immediately.

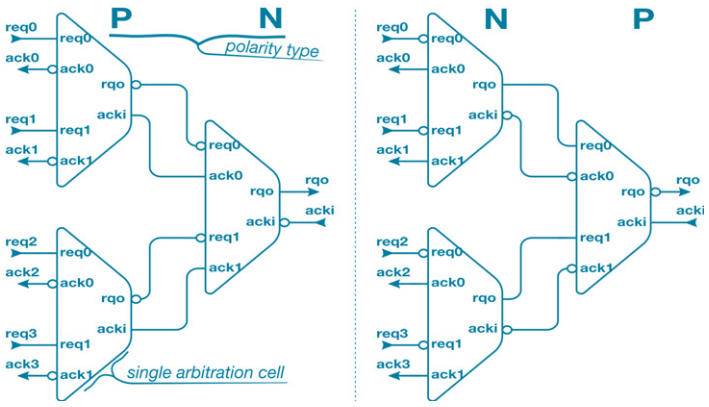

Figure 3. Hierarchical arbitration tree built using arbitration cells with different logic polarity. The use of the cells of different types of polarity avoids the use of additional inverters.

When there are multiple readout requests, each arbitration cell accepts only one of them. To distinguish which one is selected, arbiters, based on the Seitz' arbiter [4], are used (figure 4(a)). The arbitration process is based on the requests' arrival times. In the case when these times are indiscernible, the arbiter needs additional time to be moved out of metastable state. A metastability filter is used so as not to disrupt the operation of other logic blocks. An arbiter with a metastability filter also guarantees mutual exclusion of the grants *gnt[X]* signal, even during the metastable state of the arbiter. These signals are the flags that store information about which request is accepted.

A single arbitration process in the arbitration cell, solely between requests, is not enough because it may lead to unpredictable disruptions on the request output line. As a result of this interference, the acknowledge token may be withdrawn from the arbitration cell even after it has already been routed to the other acknowledge output. As a result, a channel may receive a partial acknowledge token that is not exclusive to it. Consequently, multiple channels may initiate readout transactions, resulting in a conflict on the data bus.

2022 JINST 17 C04027

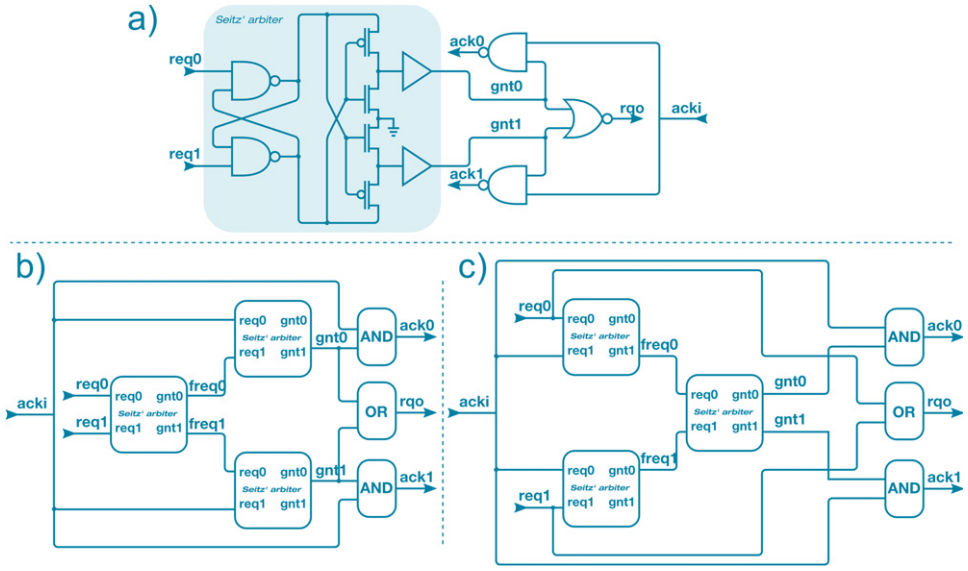

Figure 4. Various types of arbitration cells: (a) with one Seitz' arbiter, (b) fair, (c) unfair.

To overcome the problem of interference during network switching for the arbitration token, an additional arbitration stage must be introduced in the arbitration cell (figures 4(b) and (c)). This stage should ensure that the acknowledgement token is redirected only if the request output is guaranteed to remain stable during this process or if the acknowledgement token is not present in the arbitration cell. The former leads to an unfair arbitration cell, in which the next channel to be read is predetermined by its relative position to the channel currently being read out, as the token is rerouted locally, from one acknowledge output to another. In the second case, we have a fair arbitration cell, in which the token must always be first withdrawn from the cell, and only then may be redirected to another channel, starting from the top of the arbitration tree.

2.4 Synchronization

Data from the data bus are latched inside the output circuit by the clock *clk*. Data latching synchronizes the readout with the data acquisition system. Data are latched before each new token is generated, resulting in a new set of latched data for each of them. Data can then be sent serially from the chip. The serialization clock is thus used to generate acknowledge tokens by appropriately dividing it, whereby the duty cycle and frequency of the divided clock *clko* can be determined with considerable degrees of freedom based on width of data bus and maximum propagation delay for requests and acknowledge tokens passing through the arbitration tree.

3 Summary

The physical design of a pixel array, consisting of 8×8 channels, containing back-end pixels electronics and EDWARD architecture is realized and shown in figure 5. Corresponding simulation results present different number of readout phases for two channels, due to different preloaded

2022 JINST 17 C04027

configurations. The design was implemented with the use of the tools for automatic P&R and CMOS 65 nm Standard Cell Library with added designs for Seitz' arbiters. During a readout transaction, each channel sends its address (6 bits) and group sends its own address (8 bits) to the shared data bus. It is worth noting how a token is passed from one channel to other after a transaction is done. Thanks to that feature the token is reused, and no dead time is observed.

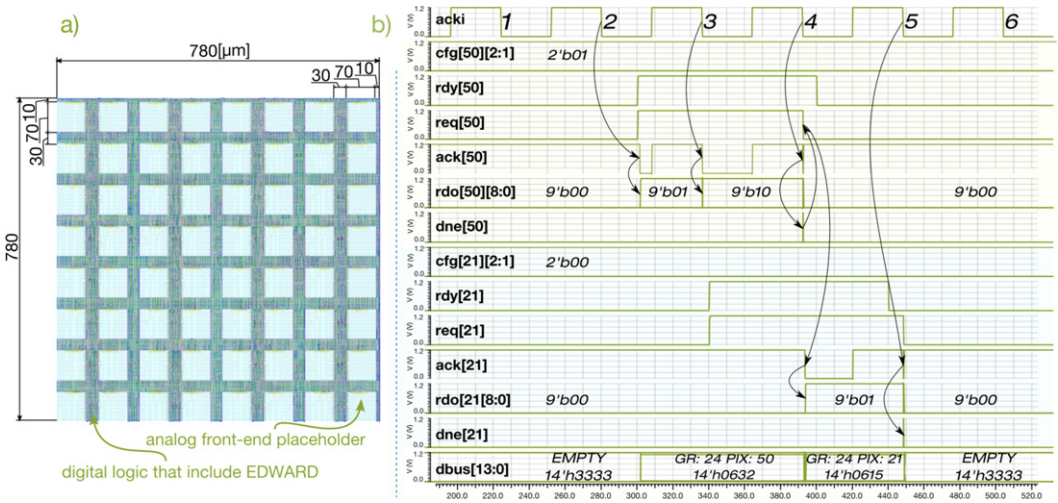

Figure 5. (a) Layout of the pixel matrix with digital logic that includes, but is not limited to, the EDWARD architecture, (b) corresponding transistor level simulation results.

Acknowledgments

This manuscript has been authored by employees of Brookhaven Science Associates, LLC under Contract No. DE-SC0012704 with the U.S. Department of Energy. The publisher by accepting the manuscript for publication acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

[1] S. Mani et al., *An asynchronous data acquisition ASIC with a data-push architecture, Nucl. Instrum. Meth. A* **360** (1995) 345.

[2] G. Deptuch et al., *A vertically integrated pixel readout device for the vertex detector at the international linear collider, IEEE Trans. Nucl. Sci.* **57** (2010) 880.

[3] P. Yang et al., *Low-power priority address-encoder and reset-decoder data-driven readout for monolithic active pixel sensors for tracker system, Nucl. Instrum. Meth. A* **785** (2015) 61.

[4] C.L. Seitz, *Ideas about arbiters, Lambda* **1** (1980) 10.

2022 JINST 17 C04027