FPGA/DSP-based Implementation of a High-Performance Multi-Channel Counter

F. Baronti, A. Lazzeri, R. Roncella, R. Saletti

Dip. Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni
Università di Pisa, Via G. Caruso 16, I-56122 Pisa (Italy)

Abstract

A high-performance configurable multi-channel counter is presented. The system has been implemented on a small-size and low-cost Commercial-Off-The-Shelf (COTS) FPGA/DSP-based board, and features 64 input channels, a maximum counting rate of 45 MHz, and a minimum integration window (time resolution) of 24 µs with a 23 b counting depth. In particular, the time resolution depends on both the selected counting bit-depth and the number of actually used channels: indeed, with a 8 b counting depth, the time resolution reaches the value of 8 µs if all the 64 input channels are enabled, whereas it lowers to 378 ns if only 2 channels are used. Thanks to its flexible architecture and performance, the system is suitable in highly demanding photon counting applications based on SPAD arrays, as well as in many other scientific experiments. Moreover, the collected counting results are both real-time processed and transmitted over a high-speed IEEE 1394 serial link. The same link is used to remotely set up and control the entire acquisition process, thus giving the system a even higher degree of flexibility. Finally, a theoretical model which immediately provides the overall system performance is described. The model is subsequently validated by the reported test results.

Key words: Counting circuits, Digital measurements, Digital signal processors, Digital systems, Field programmable gate arrays

1. Introduction

A multi-channel counter is an electronic system which is able to count the number of digital pulses detected on each input channel in a given time integration window.

Such a system plays a key role in many scientific and industrial set-ups and, most of all, in photon counting applications based on single-photon avalanche diode (SPAD) arrays. For instance, the number of photons collected by an element of the SPAD array in astronomical observation experiments is related to the luminance of the celestial surface focused on that particular element. Since a SPAD generates a digital pulse for each detected photon, the astronomical image can be displayed on a screen by mapping the counting result into the brightness of a pixel [15]. As a consequence, the quality of these experiments strongly depends on the performance of the involved pulse-counting system [9], i.e. the number of input channels, the minimum time integration window width (that is the highest time resolution) and the maximum input pulse rate.

In the last few years SPADs performance has been greatly improved [3], thanks to the development of high-speed quenching circuits [17] and the successful implementation in CMOS technology [10], [8]. For this reason, high performance pulse counters are needed to make the exploitation of these new devices possible and to significantly improve the applications in which they are used.

Nowadays there are several commercial counters

Email address: r.saletti@iet.unipi.it (R. Saletti).

that best meet the requirements of these killer-applications. In particular, besides the basic function of counting the incoming pulses, they are also able to process the collected results in real-time, thus significantly extending their field of use. However, since they are mainly based on ASICs, they are very expensive and only suitable to large volume applications.

In this paper, we present a high performance and low-cost 64-channel pulse-counting system, which has been implemented on a Commercial-Off-The-Shelf (COTS) FPGA/DSP-based board. A basic and brief description of it has been given in [1]. Here, we provide an expanded and more general theoretical analysis of the counting architecture, which can be a valuable and general tool for designing multi-channel counters. Besides, a larger experimental characterization of our system is reported and its performance is compared with state-of-the-art counterparts. It results that the proposed approach is very appealing because of its competitive performance [4], [6], along with small-size and low-cost [13], [7].

The main feature of this counting system is its flexible architecture. This allows us to easily vary the counter parameters and to adapt them to the different application requirements. For instance, the width of the integration window can vary in an extremely wide range, from few microseconds to some hundreds of milliseconds, with a step size of few nanoseconds. This is particularly useful in astronomy, since it allows to observe fast and slow phenomena, with either high or low luminescence. Moreover, the width of the integration window determines the time interval between two consecutive images, i.e. the frame rate of the imaging system, and hence the time-resolution with which the experiment is observed.

The counter can operate in a continuous mode, where a long term acquisition without dead-times between two consecutive integration windows is performed. It is also capable to process in real-time the acquired data for filtering or other elaboration. Then, it sends both the raw counting data and the processed results over a IEEE 1394 network [5] for recording and further off-line processing. As an example, Figure 1 shows a typical application scenario, in which the counter is used for astronomical observations.

A high degree of flexibility is obtained by allowing a remote user to set up and control the entire acquisition process. Indeed, it is possible to choose

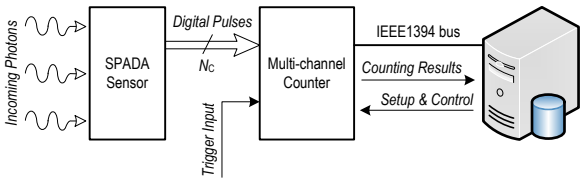

Figure 1. A typical application of the proposed system in astronomical experiments.

the width of the integration window from 8 µs to 186 ms with a step size of 11 ns, the counting depth from 8 to 23 b, the acquisition mode (continuous or triggered by a dedicated input signal) and – in the case of triggered acquisition – also the post-trigger delay and the acquisition length. Moreover, the time resolution of the measurements depends on both the number of channels used and the counting depth selected. In particular, if only 2 channels are enabled, a resolution of 378 ns with a 8 b counting depth is achievable.

Going into more details, the board used (Orsys MicroLine C6713Compact [11]) features a 1 M gate Xilinx Virtex-II Field Programmable Gate Array (FPGA) [16], a 225 MHz floating point Texas Instruments TMS320C6713 Digital Signal Processor (DSP) [14], a general purpose Texas Instruments TSB12LV32 Link-Layer Controller (LLC) [14] and a high-speed IEEE 1394 serial interface [5] (see Figure 2). The 32 b DSP External Memory IF (EMIF) Data Bus is connected to both the FPGA and the LLC. The FPGA also receives a 90 MHz clock source from the DSP EMIF. The LLC manages both the isochronous and the asynchronous transactions of the IEEE 1394 bus [5] by means of the Data Mover and µController interfaces respectively, as shown in Figure 2. The Orsys MicroLine Connector routes the 64 input channels directly to the FPGA, which implements the acquisition and counting logic. Whenever a packet of counting results is ready, the FPGA sends it to the DSP and to the LLC Data Mover IF. In this way, the DSP can process the data in real-time, while the LLC transmits them over the IEEE 1394 isochronous link, so that they can be recorded on a remote device for off-line processing. Once the DSP has processed a proper number of packets depending on the configured application, it sends the results to the LLC µController IF, so that the processed data can be transmitted over the IEEE 1394 asynchronous link.

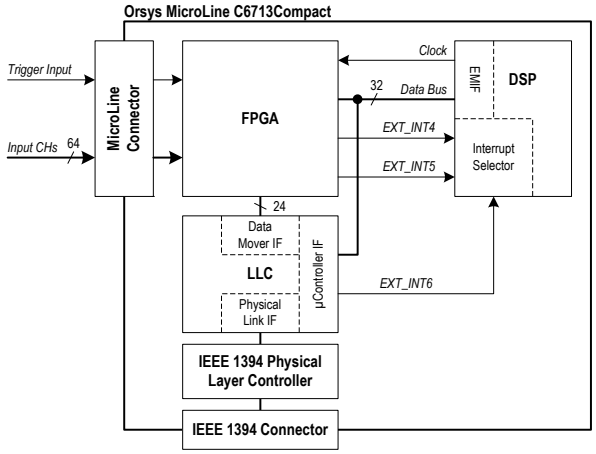

Figure 2. The architecture of the proposed system as part as the Orsys MicroLine C6713Compact architecture.

2. System Design Considerations

A counter array is generally characterized by the following parameters: the number of input channels N_{C}, the maximum value f_{INmax} of the incoming pulse-rate f_{IN}, the width T_{W} of the time integration window in which the pulses are accumulated, and the maximum counting bit-depth L_{C} available for each channel. These parameters are set according to the specific application in which the system is used.

In particular, the time integration window is often required to vary in an extremely wide range, from microseconds to hundreds of milliseconds. The upper bound T_{Wmax} of this range is related to L_{C} and f_{INmax} by the following equation

$$L_{\mathrm{C}} = \lceil \log_2(T_{\mathrm{Wmax}} \cdot f_{\mathrm{INmax}}) \rceil. \tag{1}$$

The greater T_{Wmax} and f_{INmax} are, the more logic resources on the FPGA are needed to implement each channel counter. As a consequence, being the available resources limited by the chosen implementation platform, the maximum number of channels is constrained by the desired T_{Wmax} and f_{INmax} values.

Since a smaller T_{W} value produces a larger measured data rate, the lower bound of it depends both on the available bandwidth between FPGA and DSP and on the maximum input bit-rate sustainable by the DSP for real-time processing. In particular, given the hardware platform, the minimum T_{W} depends on how the available bandwidth is used and how complex is the DSP firmware. Thus, the main design challenge and goal is to address these issues and to reach the minimum integration window possible, i.e. the maximum system time resolution.

To this end, data formatting is an important issue. Let us assume $16 < L_{\mathrm{C}} \leqslant 32$, as usually happens in many commercial counters. Let also L be the number of bits actually used to represent each result, thus it is $L \leqslant L_{\mathrm{C}}$. Since the DSP handles 32 b data, we may format – when it is possible – more than a single result in a 32 b word. Indeed, if $L = L_{\mathrm{C}}$, then each 32 b word can hold a single result only, so that the data format must be 1×32 b. Instead, if $L = 16$ or $L = 8$, then 2 or 4 consecutive results may be packed in a 32 b number respectively, i.e. a 2×16 b or a 4×8 b data format is adopted.

For each of these data formats, the DSP receives a data packet consisting of N_{C} 32 b-words every T_{W}, $2T_{\mathrm{W}}$ or $4T_{\mathrm{W}}$ respectively. In order to sustain real-time data processing, the DSP must process each data packet before the next one is ready. Generally, the time needed by the DSP to process a packet depends on the packet data format, and increases with the number of counter readings contained in it. For this reason, even if a compacted data formatting improves the bandwidth allocation between FPGA and DSP, this technique does not guarantee an improvement of the minimum T_{W} possible. A better system time resolution is only achieved under particular circumstances, as outlined by the following remarks.

For each of the above mentioned data formats, let B_{32}, B_{16} and B_8 respectively be the highest input bit-rate sustainable by the DSP in real-time. As explained above, the amount of processing needed to extract the single count results from a compacted data packet makes

$$B_{32} > B_{16} > B_8. \tag{2}$$

The values of B_{32}, B_{16} and B_8 are determined once the FPGA is hardware programmed for the given number of input channels N_{C} and the DSP is programmed for the real-time processing requested by the application.

Using the 1×32 b data format, real-time processing is possible only if

$$\frac{32 \cdot N_{\mathrm{C}}}{T_{\mathrm{W}}} \leqslant B_{32}, \tag{3}$$

that is

$$T_{\mathrm{W}} \geqslant \frac{32 \cdot N_{\mathrm{C}}}{B_{32}} \equiv T_{\mathrm{W32}}. \tag{4}$$

In other words, T_{W32} is the minimum allowed value of T_{W} when the 1×32 b data format is used.

In the same way, using the 2×16 b data format we must have

$$\frac{32 \cdot N_{\mathrm{C}}}{2T_{\mathrm{W}}} \leqslant B_{16}, \tag{5}$$

that means

$$T_{\mathrm{W}} \geqslant \frac{16 \cdot N_{\mathrm{C}}}{B_{16}} \equiv T_{\mathrm{W16}}. \tag{6}$$

Comparing (6) to (4), the system time resolution improves only if $T_{\mathrm{W16}} < T_{\mathrm{W32}}$, i.e. only if $B_{32} < 2B_{16}$.

Finally, using the 4×8 b data format, real-time processing is possible only if

$$\frac{32 \cdot N_{\mathrm{C}}}{4T_{\mathrm{W}}} \leqslant B_{8}, \tag{7}$$

i.e. only if

$$T_{\mathrm{W}} \geqslant \frac{8 \cdot N_{\mathrm{C}}}{B_{8}} \equiv T_{\mathrm{W8}}. \tag{8}$$

Again, comparing (8) to (6), the system resolution improves only if $T_{\mathrm{W8}} < T_{\mathrm{W16}}$, i.e. only if $B_{16} < 2B_8$.

In conclusion, packing 2 or 4 results in a single 32 b word determines an improvement of the system time resolution only if the sustainable bit-rate is larger at least 2 or 4 times that of 1×32 b format, i.e.

$$B_{32} < 2B_{16} < 4B_8. \tag{9}$$

The main design goal is thus to satisfy (9).

Let us assume (9) is satisfied. It is now interesting to see if a generic working condition $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ will lead to counters saturation once the data format, i.e. the value of $L = L_{\mathrm{C}}, 16, 8$ is chosen. In the log-log plane $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ we can plot the curves given by (see Figure 3):

$$\begin{cases} \gamma_{\mathrm{C}} \ : T_{\mathrm{W}} \cdot f_{\mathrm{IN}} = 2^{L_{\mathrm{C}}} \\ \gamma_{16} \ : T_{\mathrm{W}} \cdot f_{\mathrm{IN}} = 2^{16} \\ \gamma_{8} \ : T_{\mathrm{W}} \cdot f_{\mathrm{IN}} = 2^{8} \end{cases} \tag{10}$$

The curve γ_{C} represents those pairs $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ which exactly cause L_{C} bits-wide counting results. In the same way, the curves γ_{16} and γ_8 represent those pairs which cause exactly 16 and 8 bits-wide counting results respectively.

If the input parameter pair $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ sets a point above γ_{C}, the counter will saturate. Instead, if the array works below γ_{C}, we will have three different cases:

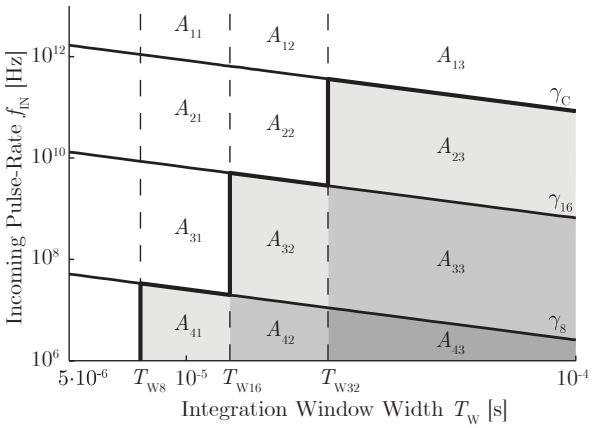

Figure 3. The $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ plane partitioning.

A_{jk}	1	2	3
1	4×8 b	2×16 b 4×8 b	1×32 b 2×16 b 4×8 b
2	4×8 b	2×16 b 4×8 b	1×32 b $\star$ 2×16 b 4×8 b
3	4×8 b	2×16 b $\star$ 4×8 b	1×32 b $\star$ 2×16 b $\star$ 4×8 b
4	4×8 b $\star$	2×16 b $\star$ 4×8 b $\star$	1×32 b $\star$ 2×16 b $\star$ 4×8 b $\star$

Note: a $\star$ means a non-saturating format

Table 1
The $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ plane partitioning summary.

– if $L = L_{\mathrm{C}}$, it must be $T_{\mathrm{W}} \geqslant T_{\mathrm{W32}}$, and saturation will never occur;
– if $L = 16$, it must be $T_{\mathrm{W}} \geqslant T_{\mathrm{W16}}$, and we have to consider γ_{16}: saturation will only occur above it, and not below it;
– if $L = 8$, it must be $T_{\mathrm{W}} \geqslant T_{\mathrm{W8}}$, and we have to consider γ_8: saturation will only occur above it, and not below it.

As a consequence, the plane $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ is partitioned into twelve zones, named A_{jk} $(j = 1, \ldots, 4;\ k = 1, \ldots, 3)$ as shown in Figure 3. The data formats available in each zone and the related saturation occurrence are summarized in Table 1.

Finally, the same diagram extended up to show T_{Wmax} is reported in Figure 4. This diagram resumes the above theory and gives an immediate view of the performance obtainable by the system for different input parameters. In particular, Figure 4 gives:
– the narrowest T_{W} for a given f_{IN};
– the highest f_{IN} for a given T_{W}.

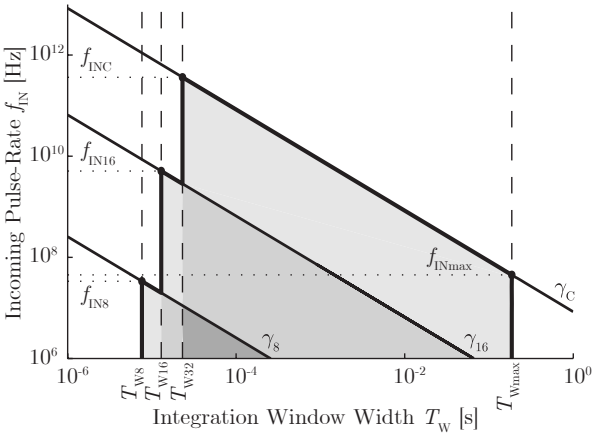

Figure 4. The diagram for the overall system performance.

Consider, for instance, the first point. Let f_{INC}, f_{IN16} and f_{IN8} be the values of γ_{C}, γ_{16} and γ_8 at T_{W32}, T_{W16} and T_{W8} respectively. From 10 we have:

$$\begin{cases} f_{\mathrm{INC}} = \dfrac{2^{32}}{T_{\mathrm{W32}}} = \dfrac{2^{32}}{32} \cdot \dfrac{B_{32}}{N_{\mathrm{C}}} \\ f_{\mathrm{IN16}} = \dfrac{2^{16}}{T_{\mathrm{W16}}} = \dfrac{2^{16}}{16} \cdot \dfrac{B_{16}}{N_{\mathrm{C}}} \\ f_{\mathrm{IN8}} = \dfrac{2^{8}}{T_{\mathrm{W8}}} = \dfrac{2^{8}}{8} \cdot \dfrac{B_{8}}{N_{\mathrm{C}}} \end{cases} \tag{11}$$

If $f_{\mathrm{IN}} \leqslant f_{\mathrm{IN8}}$ then the narrowest integration window is T_{W8} and is obtained with a 4×8 b data format. Instead, if $f_{\mathrm{IN8}} < f_{\mathrm{IN}} \leqslant f_{\mathrm{IN16}}$ then the narrowest window is T_{W16} and is obtained with a 2×16 b data format. Finally, if $f_{\mathrm{IN16}} < f_{\mathrm{IN}} \leqslant f_{\mathrm{INC}}$ then the narrowest window is T_{W32} and is obtained with a 1×32 b data format. For $f_{\mathrm{IN}} > f_{\mathrm{INC}}$ every working condition $(T_{\mathrm{W}}, f_{\mathrm{IN}})$ will lead to counters saturation. However, it should be noted that the input parameter pairs that represent a real working condition are those with $f_{\mathrm{IN}} \leqslant f_{\mathrm{INmax}}$ and with $T_{\mathrm{W8}} \leqslant T_{\mathrm{W}} \leqslant T_{\mathrm{Wmax}}$.

3. FPGA Architecture

The FPGA architecture consists of two main modules: the Data Processing Module (DPM) and the Data Mover Port IP Core (DMPORT), which is an IP module provided by Orsys [11]. The DMPORT is directly connected to the LLC Data Mover Interface, which manages the IEEE 1394 isochronous data transfers, as shown in Figure 5. The DPM consists of the N_{C} Channel Modules (ChMs) and the DPM Control Module (DCM).

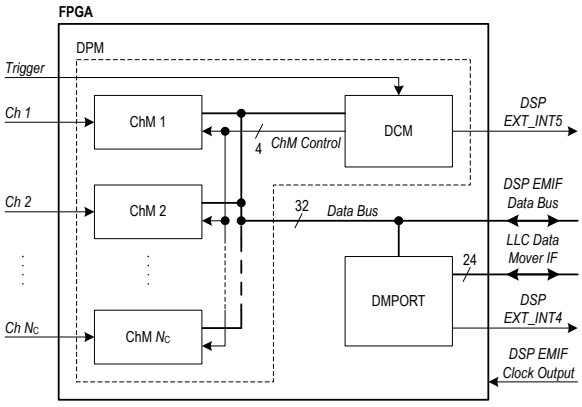

Figure 5. The proposed FPGA architecture.

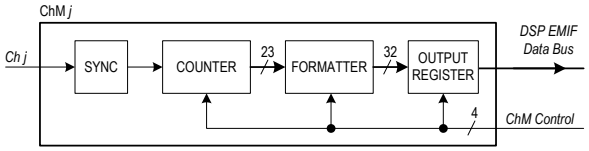

Figure 6. The proposed ChM architecture.

The DPM control module handles the operation of the counter array according to the acquisition mode (continuous or triggered) and the related parameters, which can be configured by the DSP. In fact, the DCM configuration registers can be accessed by the DSP through read/write operations on its external memory space. In this way it is possible to set the values of T_{W} and L, i.e. the format of the output data. Moreover, the post-trigger delay and the duration of the acquisition can also be set in the case of triggered acquisition mode.

The DCM drives each ChM with the same control signals, so that the synchronous operation of all the ChMs is achieved. Each ChM (Figure 6) implements the logic necessary to synchronize and count the incoming events from its related input channel, and is able to format 1×32 b, 2×16 b or 4×8 b consecutive counting results in a single 32 b output word. When a new word is ready (i.e. a new packet is ready), it is immediately stored in the ChM Output Register, without suspending the counting process. This allows a continuous input acquisition without dead times. Then, the DCM sends an interrupt signal to the DSP for the retrieval of the packet.

The only FPGA clock source is the DSP EMIF Clock Output, the frequency f_{CLK} of which is 90 MHz. The design style adopted is fully-synchronous, so that we chose to represent T_{W} as a multiple of the FPGA clock cycle with a 24 b value. T_{W} thus varies from few microseconds to hundreds

of milliseconds with the clock cycle time as the narrowest step. In this way, T_{W} can reach $T_{\mathrm{Wmax}} = 2^{24}/f_{\mathrm{CLK}} \simeq 186$ ms, with a $1/f_{\mathrm{CLK}} \simeq 11$ ns step size.

A conventional two flip-flop synchronizer (in which the first flip-flop is asynchronously reset by the second one) provides the synchronization of the incoming events in each ChM. For this reason, the maximum frequency of the synchronized signal which drives the counter is $f_{\mathrm{CLK}}/2 = 45$ MHz. Incoming asynchronous events with a pulse rate $f_{\mathrm{IN}} > f_{\mathrm{CLK}}/2$ are not recognized by the system, thus $f_{\mathrm{INmax}} = f_{\mathrm{CLK}}/2$. As a consequence, being $T_{\mathrm{Wmax}} = 2^{24}/f_{\mathrm{CLK}}$, we have from (1) $L_{\mathrm{C}} = 23$. In other words, the highest achievable count result is 2^{23}, so that a 23 b counter is implemented in each ChM.

Finally, the ChM Formatter prepares the related 32 b output word according to the chosen data format. In particular, should the count exceed the maximum representable value, the result is fixed to that value, avoiding the counter roll-over and the transmission of wrong results.

4. DSP Firmware

The DSP provides the real-time data processing and the board communication to the external world. It receives the acquisition setting and control commands from the remote user by means of the IEEE 1394 asynchronous link, and sets up the DPM and the DMPORT. Moreover, the same link is used to periodically send back to the user the results obtained during the real-time processing.

The DSP main program manages the asynchronous transfers by means of a callback functions queue, i.e. a pointer-to-function queue implemented in the DSP internal memory. Whenever the LLC has to interact with the processor, it triggers an interrupt request. Once started, the related Interrupt Service Routine (ISR) reads the LLC interrupt code and inserts in the callback queue a pointer to that specific function which performs the operations requested by the LLC. However, this function is not immediately called, and the main program continues its normal execution when the ISR returns. It is up to the main program to periodically check the callback queue and, if it is not empty, to extract the first pointer and call the pointed callback function.

The packet transfers from FPGA to DSP are managed in background by the processor Enhanced DMA (EDMA), using a Ping-Pong buffering approach. Two buffers are allocated in the internal processor memory. Each of them is able to store M incoming packets. While one buffer is filled by the EDMA, the other is processed by the DSP, and vice versa. In particular, the EDMA continuously listens to the interrupt requests coming from the DPM. Whenever the DPM triggers a request, the EDMA downloads a new packet. Finally, when one of the two buffers is filled, the EDMA triggers an interrupt request to the processor. The related ISR simply tells the main program which of the two buffers is ready to be processed, by setting an appropriate flag. Again, it is up to the main program to periodically check this flag and start the buffer processing if the flag is set.

Anytime the DSP is interrupted, it spends a fixed amount of time to set up the ISR and another time to execute it, the amount of which is variable and depends on the interrupt routine complexity. Therefore, the body of each ISR should be rather small, in order to reduce the time during which the main program execution is suspended.

In our real-time system, the critical code section is represented by the Ping-Pong buffer processing routines. Let T_{PRC} be the time needed by the DSP to process a single packet, and let $T_{\mathrm{PKT}} = T_{\mathrm{W}} \cdot F$ be the lifetime of each packet, where F – according to the data format chosen – is the number of consecutive results in each 32 b ChM output word, i.e. 1, 2 or 4. In order to sustain real-time data processing we must obviously have

$$T_{\mathrm{PKT}} > T_{\mathrm{PRC}}. \tag{12}$$

Let T_{ISR} be the time needed by the DSP to handle an EDMA interrupt request. Since the EDMA interrupts the processor just when a buffer is filled, the DSP must complete the ISR and process an entire buffer before the next EDMA interrupt occurs. As a consequence, we must have

$$T_{\mathrm{ISR}} + M \cdot T_{\mathrm{PRC}} < M \cdot T_{\mathrm{PKT}}, \tag{13}$$

hence, by (12),

$$M > \frac{T_{\mathrm{ISR}}}{T_{\mathrm{PKT}} - T_{\mathrm{PRC}}}. \tag{14}$$

Moreover, if one or more LLC interrupt requests occur while processing a buffer, we have also to consider the overall time τ spent by the processor to handle the callback queue. Hence

$$M > \frac{T_{\mathrm{ISR}} + \tau}{T_{\mathrm{PKT}} - T_{\mathrm{PRC}}}. \quad (15)$$

It is worth noting that the LLC interrupt requests should never be disabled, otherwise the callback queue layer – as the asynchronous link itself – would fail. This means that it is not possible to guarantee $\tau = 0$.

The real-time processing routines are entirely application-dependent. We chose to implement a medium-complex algorithm, with a complexity of 200-650 elementary instructions per packet depending on the data format, in order to estimate the performance of our system when used in a particular application. This algorithm performs, for each channel, a periodic accumulation of the counting results coming from the FPGA, and stores the accumulated data in a dedicated Accumulation Buffer. This buffer is allocated in the internal DSP memory, and it is $N_{\mathrm{C}} \times 32$ b wide. Every time $P \cdot M$ packets have been processed (i.e. Ping and Pong buffers have been processed P times), the Accumulation Buffer is uploaded to the user by means of the asynchronous link. To do this, the transfer between the DSP and the LLC µController IF is carried out in background by the processor Quick DMA (QDMA). Then the Accumulation Buffer is re-initialized and the algorithm restarts. In this way, the DSP performs a further integration of the counting values in a window $T_{\mathrm{ACC}} = P \cdot M \cdot T_{\mathrm{W}} \cdot F$. Once the user has set T_{W} and F, the value of P is chosen by the DSP to have for T_{ACC} the smallest value greater than 100 ms. Figure 7 shows the main program flow-diagram that the DSP executes.

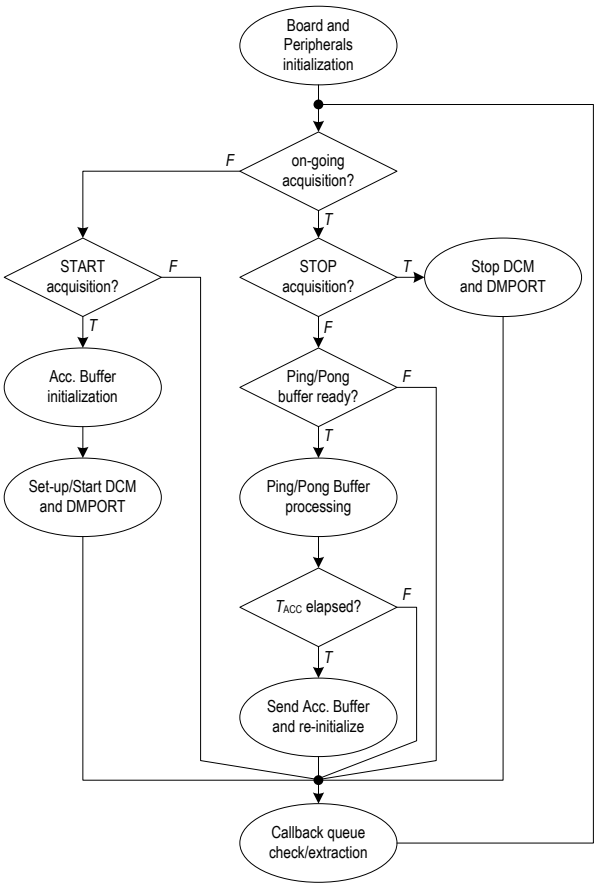

Figure 7. The DSP main program flow-diagram.

5. Design Implementation

The FPGA was designed with the Xilinx ISE Foundation development system [16]. With $N_{\mathrm{C}} = 64$, the FPGA logic resources occupation is 97%. Table 2 reports the occupation statistics for both the entire design and every single module.

It is worth noting that the entire design has an equivalent gate count larger than the number of gates available in the FPGA. Indeed, the synthesis was forced to map as many logic functions as possible in the internal FPGA RAM Blocks, to fit the design in the target device. Moreover, the total slice count is much lower than the sum of each module slice count, since many slices are shared between two or more modules. The same happens to LUTs. However, the values reported give a good comparative figure of the complexities of the different modules.

Design Module	Slices		LUTs		Equivalent Gates
	#	%	#	%	
Entire Design	5,003	97.71	9961	97.28	2,747,130
DCM	154	3.01	244	2.38	10,016
ChM Sync	2	0.04	0	0.00	32
ChM Counter	14	0.27	27	0.26	357
ChM Formatter	64	1.25	122	1.19	924
ChM Out Reg	16	0.31	0	0.00	355
DMPORT	327	6.39	403	3.94	275,141

Table 2
FPGA resources occupation statistics ($N_{\mathrm{C}} = 64$).

As far as the DSP firmware is concerned, the time-to-completion of the critical code section has been optimized by letting the compiler unroll the Ping-Pong buffer processing loop M times. In particular, in order to determine the value of M, we measured

Data Format	Channel Processing Time [ns]			Buffer Processing Time [µs]		
	Min	Avg	Max	Min	Avg	Max
4 × 8 b	459	461	465	470.300	471.891	476.046
2 × 16 b	388	389	394	397.596	398.708	397.596
1 × 32 b	345	345	346	353.195	353.601	353.963

Table 3
DSP critical code section statistics ($N_C = 64, M = 16$).

both the time spent by the DSP to process a single 32 b channel word – obtaining the values reported in the first section of Table 3 – and the average time needed to handle the callback queue and the DMA interrupt requests – obtaining a value of about 16 µs for each main program cycle. In this way, M has been chosen as a compromise between (15) and the DSP memory allocation, since its value determines the size of both the Ping-Pong buffer and the unrolled code section. As a good trade-off, we set $M = 16$, thus obtaining the values reported in the second section of Table 3.

Using the average channel processing time values reported in Table 3, the values of B_{32}, B_{16} and B_8 can be estimated as follows:

$$\begin{cases} B_{32} = \dfrac{M \cdot N_C \cdot 32}{(M \cdot N_C \cdot 345 \cdot 10^{-3} + 16)\ \mu s} \\ B_{16} = \dfrac{M \cdot N_C \cdot 32}{(M \cdot N_C \cdot 389 \cdot 10^{-3} + 16)\ \mu s} \\ B_8 = \dfrac{M \cdot N_C \cdot 32}{(M \cdot N_C \cdot 461 \cdot 10^{-3} + 16)\ \mu s} \end{cases} \quad (16)$$

thus obtaining

$$\begin{cases} B_{32} \simeq 88.73 \text{ Mbps} \\ B_{16} \simeq 79.09 \text{ Mbps} \\ B_8 \simeq 67.14 \text{ Mbps} \end{cases} \quad (17)$$

These values satisfy (9). Moreover, according to (4), (6) and (8) we have:

$$\begin{cases} T_{W32} \simeq 23.1\ \mu s \\ T_{W16} \simeq 13\ \mu s \\ T_{W8} \simeq 7.6\ \mu s \end{cases} \quad (18)$$

Finally, from (17) and (11) we also have:

$$\begin{cases} f_{IN16} \simeq 5.06 \text{ GHz} \\ f_{IN8} \simeq 33.57 \text{ MHz} \end{cases} \quad (19)$$

so that the 45 MHz threshold given by f_{INmax} can easily be traced on the diagram reported in Figure 4.

6. Test

The multi-channel counter has been tested by connecting the board to a Linux PC with an IEEE 1394 standard cable [2]. A user interface developed in C++ allows the user to choose and send to the DSP the acquisition parameters, to monitor the accumulation data coming from the DSP every T_{ACC} seconds, and to store on the hard-disk each data packet coming from the FPGA. The counter-array channel inputs have been connected to a waveform generator that provides the input pulses to be counted.

Several experiments have been carried out, by changing the input parameters (input data rate and integration window) and reading the counted results. The experimental tests confirms the expected system performance. In particular, the actual values $\tilde{T}_{W32}$, $\tilde{T}_{W16}$ and $\tilde{T}_{W8}$ for the minimum integration windows have been measured. The values are the following

$$\begin{cases} \tilde{T}_{W32} \simeq 24\ \mu s \\ \tilde{T}_{W16} \simeq 14\ \mu s \\ \tilde{T}_{W8} \simeq 8\ \mu s \end{cases} \quad (20)$$

and they are very close to those given by (18).

In conclusion, we can state that the innovative and flexible architecture adopted makes this 64 channel counter array capable of acquiring and counting incoming events with the following correspondence between the highest time resolution and the maximum counting depth: 8 µs - 8 b, 14 µs - 16 b, and 24 µs - 23 b. In particular, for a given pair (T_W, f_{IN}) with $f_{IN} \leqslant f_{INmax}$ and $T_{W8} \leqslant T_W \leqslant T_{Wmax}$, the diagram in Figure 4 let us immediately know the allowed bit depths and the possibility of count saturation.

In order to further show the power of the architecture used, the system has also been implemented and tested for a lower number of channels, i.e. for $N_C = 32, 16, 8, 4, 2$. The results are reported in the first section of Table 4. It is worth noting that, as N_C decreases from 64 to 32, M increases from 16 to 32, so that the Ping-Pong buffer size remains the same, whereas the unrolled code section size doubles. Instead, as N_C decreases from 32 to 16, M cannot be further increased from 32 to 64, since the size of the unrolled code section would become too large to fit in the DSP internal memory. The same happens in

Real-time processing	N_C	M	$\tilde{T}_{W8}$	$\tilde{T}_{W16}$	$\tilde{T}_{W32}$
enabled	64	16	8 µs	14 µs	24 µs
	32	32	4 µs	7 µs	12 µs
	16	55	2 µs	4 µs	6 µs
	8	90	1 µs	2 µs	3 µs
	4	93	789 ns	1.4 µs	2.1 µs
	2	95	378 ns	744 ns	1.4 µs
disabled	64	16	2.7 µs	5.4 µs	10.5 µs
	2	95	88 ns	189 ns	378 ns

Table 4
Maximum system performance.

the other cases. However, by lowering the number of channels and increasing the value of M, the overall performance increases. In particular, the system is able to reach the maximum time resolution of 378 ns, when configured for only 2 channels.

Moreover, we disabled the real-time processing routines and measured the system performance in the two cases for $N_C = 64$ and $N_C = 2$. In these cases, the DSP only handles the out-board communications, by configuring the DPM and the DMPORT with the acquisition parameters supplied by the remote user, and it is not asked to execute any processing algorithm. This set up is very similar to the typical working condition of many commercial counters, for which real-time processing, if any, is provided only by the remote PC and not by on-board resources. The results are reported in the second section of Table 4. It is worth noting that, with only 2 channels, a value of 88 ns for the time resolution is achieved.

Let us now compare this counting system with other state-of-the-art counters. To the best of our knowledge, this is the only implementation of a general purpose and flexible multi-channel counter based on COTS components, like FPGAs or DSPs. Most advanced counters are generally based on ASICs designed to satisfy the specific requirements of the targeted application. Usually they feature very high time resolutions thanks to integrated Time-to-Digital Converters (TDC). This is paid in terms of high development times and costs and very low flexibility. Also, their performance parameters are related to the specific application they are designed for, and result rather inhomogeneous from counter to counter. As a consequence, it is first necessary to extract form the performance reported in their datasheets a set of parameters like the one used in our analysis, in order to set-up an effective comparison.

As a significant example, let us consider one of the most advanced commercial counters for photon counting applications [12]. From its features we extracted the following equivalent parameters: $N_C = 2$, $T_{Wmin} = 260$ ns, $T_{Wmax} = 33$ µs, 4 ps T_W step size, $f_{INmax} = 10$ MHz, $L = 16$, no real-time processing which is performed by an external PC. It can be noted that our counter, configured with $N_C = 2$, $L = 16$, and without any on-board processing to make a fair comparison, can reach a lower T_W.

Instead, the outstanding T_W step size feature is significantly lower than ours, thanks to a dedicated internal TDC. However, the comparison demonstrates that our counting system shows comparable performance with the state-of-the-art multi-channel counters and adds to them the flexibility that makes it adaptable to many different applications and configurations. Furthermore, the use of programmable COTS hardware makes possible to build a reconfigurable counting system, the parameters of which could be adapted on-the-fly to the application requirements.

7. Conclusion

A high-performance architecture for multi-channel counter implementation has been presented. It is a flexible architecture where the computing power provided by a DSP is joined to an efficient synthesis of the counting logic on an FPGA. The proposed system features up to 64 input channels and is able to acquire incoming events with a pulse rate up to 45 MHz.

The main feature of this counter is the flexibility, that allows us to choose the width of the time integration window between 8 µs and 186 ms with a step size of 11 ns, and a counting depth of 8, 16 or 23 b. The acquisition mode can be continuous or triggered by a dedicated input signal, for which the post-trigger delay and the acquisition length can be set. In any case, the counter results are real-time processed to realize a further integration step over consecutive windows, and the count results are uploaded on an external PC. If the counter is programmed to manage only 2 channels, the minimum integration window width – i.e. the system time resolution – lowers to 378 ns. The counter is realized with COTS programmable hardware and could be

designed to be reconfigurable on-the-fly, by adapting its parameters to the application needs.

The raw counting and the processed results are transmitted over a high-speed IEEE 1394 serial link, for remote data recording and off-line processing. A remote user controls the acquisition parameters with commands uploaded over the same link, by means of a user interface on the external PC.

Finally, a theoretical model which allows us to calculate the overall system performance has been provided. This model is validated by test results, and can easily be reused for every other FPGA/DSP-based counter array.

A comparison with state-of-the-art multi-channel counters shows that the performance obtained is comparable, but the flexible architecture and the theoretical model developed allow the designer to efficiently adapt our counter to many applications which differ for number of channels, time integration window, input data rate and amount of real-time processing requirements.

References

[1] D. Audino, F. Baronti, A. Lazzeri, R. Roncella, R. Saletti, FPGA/DSP-based configurable multi-channel counter, in: Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), 2007.

[2] D. Audino, F. Baronti, R. Roncella, R. Saletti, S. Tisa, F. Zappa, M. Belluso, G. Bonanno, 60-channel 10 µs time-resolution counter array for long term continuous event counting, IEEE Trans. Nucl. Sci. 54 (3) (2007) 549–554.

[3] M. Ghioni, A. Gulinatti, I. Rech, F. Zappa, S. Cova, Progress in silicon single-photon avalanche diodes, IEEE Journal of Selected Topics in Quantum Electronics 13 (4) (2007) 852–862.

[4] S. Hansen, T. Jordan, T. Kiper, D. Claes, G. Snow, H. Berns, T. H. Burnett, R. Gran, R. J. Wilkes, Low-cost data acquisition card for school-network cosmic ray detectors, IEEE Trans. Nucl. Sci. 51 (3) (2004) 926–930.

[5] IEEE standard for a high performance serial bus, 445 Hoes Lane, Piscataway, NJ08855 (aug 1996).

[6] D. R. Lee, S.-W. Lee, J. W. Jeon, Frame grabber circuit for IEEE1394 image transfer, in: Proceedings of the International Conference on Control, Automation and Systems, 2007.

[7] M. D. Lepage, G. Léger, J. Cadorette, J. D. Leroux, M. Otis, S. Rodrigue, R. Lecomte, FPGA/DSP-based coincidence unit and data acquisition system for the sherbrooke animal PET scanner, in: Proc. of IEEE Nucl. Sci. Symp., vol. 2, 2000.

[8] C. Niclass, M. Gersbach, R. Henderson, L. Grant, E. Charbon, A single-photon avalanche diode implemented in 130 nm CMOS technology, IEEE Journal of Selected Topics in Quantum Electronics 13 (4) (2007) 863–869.

[9] C. Niclass, A. Rochas, P. Besse, R. Popovic, E. Charbon, A 4 µs integration time imager based on CMOS single photon avalanche diode technology, Sensors and Actuators A: Physical Volumes 130-131 (2006) 273–281.

[10] C. Niclass, M. Sergio, E. Charbon, A CMOS 64x48 single photon avalanche diode array with event-driven readout, in: Proceedings of the 32nd European Solid-State Circuits Conference (ESSCIRC 2006), 2006.

[11] Orsys Orth Systems GmbH, D-88677 Markdorf, Germany, datasheet available at http://www.orsys.de.

[12] PicoQuant PicoHarp300, datasheet available at http://www.picoquant.com.

[13] L. Pinot, R. Sellem, J. C. Cuzon, A. Lasage, R. Mastrippolito, L. Valentin, A compact data acquisition system for TOHR multidetectors: Time encoding for both time and energy measurements, IEEE Trans. Nucl. Sci. 50 (2) (2003) 272–277.

[14] Texas Instruments Inc., Dallas, TX 75243-4136, USA, datasheet available at http://www.ti.com.

[15] S. Udo, M. Fukushima, K. Kadota, T. Matsuda, S. Ogio, H. Sagawa, A. Taketa, Y. Tameda, M. Tanaka, Signal digitizer-finder system prototype for TA fluorescence telescope, in: Nuclear Science Symposium Conference Record, vol. 1, 2004.

[16] Xilinx Inc., San Jose, CA 95124-3400, USA, datasheet available at http://www.xilinx.com.

[17] F. Zappa, A. Gulinatti, P. Maccagnani, S. Tisa, S. Cova, SPADA: single-photon avalanche diode arrays, Photonics Technology Letters, IEEE 17 (3) (2005) 657–659.