C-015

Energy Efficient Motion Capturing System Using Functional Vision Sensor

Georgios Dikaros†‡ Ioannis Papakostas†‡ Shuhei Maeda† Takuro Ohmaru† Takayuki Ikeda† Shunpei Yamazaki†

1. Introduction

Limited power environments such as sensor networks require all components to be as effective as possible while minimizing power consumption. There is high interest in developing efficient vision sensors for motion capturing and some different approaches, namely frame-based [1] and event-driven [2, 3] vision sensors have been reported. Neither of these can give satisfying results, due to the difficulty, of the former to capture slowly moving objects and of the latter to perform both motion and image capturing because of its complicated pixel configuration.

To overcome these problems, a new vision sensor for motion capturing was proposed in [5]. It featured in-pixel non-volatile analog memory utilizing a c-axis-aligned a-b-plane-anchored crystal In-Ga-Zn-O (CAAC-IGZO), that constitutes the base for a FET with extremely low off-state current [4] and can easily retain captured data of a given reference frame. The sensor had three operational modes, imaging, motion capturing and standby, which could be set up and selected through a GUI that was developed for this exact purpose. Although many of the initial issues are addressed, the proposed system was still not practical. In this paper we take this implementation a step forward, first, by creating a circuit that makes this transition among the three modes completely automatic, ensuring the proper usage of the whole device as an independent motion capturing system and then with a second implementation to make it even more efficient.

2. Operating Modes – Implementation 1

The system we created was designed in a way that it allows for maximum reduction of power consumption through transitioning among the three operating modes. Figure 1 shows the block diagram of the sensor as well as which blocks are activated in each mode. The controller is the component that handles the operation of all the parts of our sensor, namely the analog processor, the row/column drivers, the pixel array and the ADC depending on the mode. In the motion capturing mode all the components are activated except for the column driver and the ADC. After a target frame is captured, the analog processor is responsible for evaluating the differential data between the target and a reference frame generated by each pixel of the pixel array and signaling if motion has been detected or not. If there is indeed some motion detection, the controller drives the system to the standby mode by deactivating all the components until the next rising edge of the system clock, when the system enters the imaging mode. In the imaging mode all blocks but the analog processor are once again reactivated, including the column driver and the ADC. The frame, right after the target frame obtained during the motion capturing mode, is captured, digitized and stored as the new reference frame. In Figure 2 we present the flowchart of the entire process of motion capturing including all three operating modes. The only component that remains constantly active as long as our system is powered on and orchestrates the transition among the three modes is the controller, thus we activate it only once in the beginning. Since the core goal of our system is to detect motion, there should be a triggering signal that is set once the sensor captures motion. In our flowchart this signal is defined as “MT” and its value is critical for switching between the modes. This signal is evaluated on every rising edge of the system clock and in case it is set, our system transitions to the imaging mode whereas if no motion has been detected, the motion capturing mode is selected. In the imaging mode, apart from capturing a new frame and digitizing it, this new frame is also stored as the new reference frame in order to be compared with the target frame that is captured during the motion capturing mode. Since without a stored reference frame the whole process cannot be initiated, “MT” is initialized as “True” so that during the very first period of our system clock, all the operations taking place in the imaging mode, including the capturing and storage of a new reference frame, are executed.

From that point on our system follows an iterative process passing through the different modes depending on the result of the comparison between the voltage levels of the pixels of the target and the reference frame. More specifically in the motion capturing mode, the target frame is captured and its rows are scanned and compared one by one with the corresponding rows of the reference frame. Through this comparison, differential data are computed, which in turn are compared with the predefined positive and negative margins. In case the computed difference in voltage exceeds the positive or negative allowed deviation, this means that motion has been detected and no further scanning of the target frame’s rows is needed. The sensor is switched to the standby mode, the MT signal is set and all the components are switched off until the next rising edge of the system clock, minimizing the power consumption.

The next rising edge of the system clock will guide the process through the imaging mode (MT is True) that will capture the next frame (the one right after the target frame that featured voltage difference in its pixels), store it as the new reference frame, digitize it and output it.

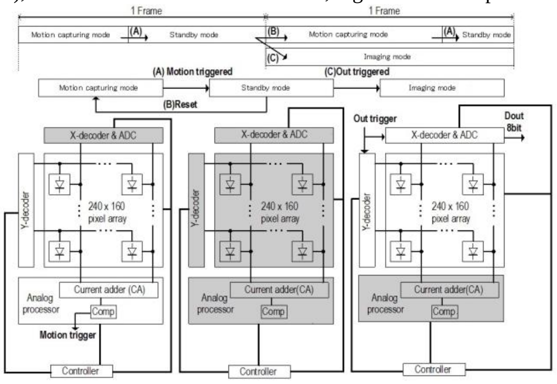

Fig.1 Motion capturing system block diagram

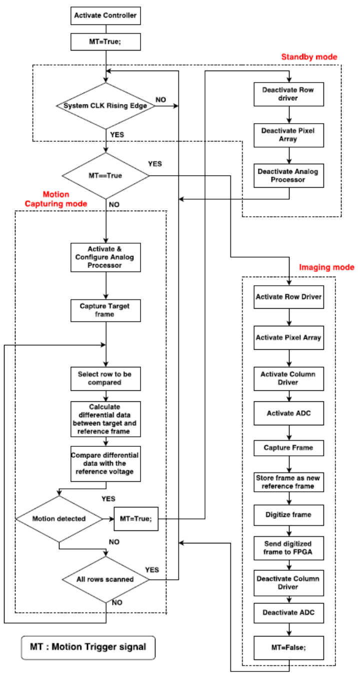

Fig.2 Flowchart for Implementation 1 (60fps)

3. Implementation 2

As it has already been mentioned above, power consumption has always been a critical factor for the effectiveness of our system. Taking into account that choosing a lower frequency for the system clock could significantly reduce power consumption, we started investigating how we could proceed with such a modification in a way that we wouldn’t lose so much in terms of accuracy as well. Our sensor in the previous implementation was configured to capture 60 frames per second, frequency high enough (at least for objects moving in reasonable speed) to allow for selection of a lower value. With that being said we decided to reduce this frequency in half (30fps), however in order to compensate for

† Semiconductor Energy Laboratory Co., Ltd.
‡ KTH Royal Institute of Technology

losing half of the frames we had to come up with a different algorithm so as to capture as many images as possible during this time period of 30 frames.

Our algorithm is based on the following three basic features.

1. Reduction of the system frequency in half.

We chose to reduce our system frequency in half so that the operation is performed in 30fps (initial implementation was 60 fps). This was the major design decision for better power consumption but we also had to consider not altering the functionality.

2. Motion detection is followed by N frames in the imaging mode.

Our thoughts behind this choice were that if motion is detected in one frame, there is high probability that we will detect motion in the next N frames as well. By keeping the system in imaging mode for more than 1 frame we compensate for the reduction in frequency, while capturing consecutive images of a moving object.

3. No motion detection, is followed by M frames in Standby mode.

When not detecting motion, there is high probability that no motion will be detected in the next M frames either, or even if motion starts on one of the next M frames it will probably last long enough for the system to capture it the next time it operates in motion capturing mode.

In figure 3 we present the flow chart of the second implementation. The basic procedures taking place in each mode are the same, however in order to integrate the aforementioned logic we replaced the MT signal with two counters, the Standby Counter (SC) and the Imaging Counter (IC). The former is assigned the value M (we want to drive the system to the standby mode and lock it there for M frames) whenever there is no motion detection, while the latter is assigned the value N straight after motion detection takes place. This creates a temporary loop of N times and consequently equivalent number of digitized frames, during which the system is locked in the imaging mode until IC becomes zero and it returns to motion capturing mode. The system operates as described in an iterative way. Finally one change that helps towards further increasing the efficiency of our system and is probably worth mentioning, is that since our system is configured in such a way that stays for N times in the Imaging mode, the deactivation of the Column Driver and the ADC is not performed inside the Imaging mode as before but instead this happens once the system enters again the Motion Capturing mode.

4. Discussion

In this section we will present our estimations and calculations which make both of the created algorithms very efficient. What is worth mentioning though, is that for the second algorithm, the number of frames for which the system remains either on imaging mode or standby mode is totally dependent on the application our system is used for. However, in order to illustrate the difference in power consumption between the two implementations, we had to choose two proper values for a more general case. After careful analysis we concluded on **N = 3** and **M = 1** which we consider good enough so as to drop energy needs but also not lose so much in accuracy**.** Of course these values are yet to be optimized.

According to [5], the power consumption for the different operating modes if we assume that the system stays on the same mode for 60 frames is 25.3μW for motion capturing, 1.88μW for standby and 3.6mW for imaging mode. Taking these values as reference we can adjust them to make estimations and compare the power consumption in the time period of 1 second between our initial algorithm and the new algorithm we have created.

Assuming that alternating current dominates both the motion capturing as well as the imaging mode, this means that power is proportional to frequency according to: $P = C * V^2 * f$ *(1)*, thus we can estimate that lowering the frequency in half will end up in lowering the power in half for these two modes. As far as the standby mode is concerned, there is only DC current flowing, which means power remains the same regardless of the change in frequency. For our calculations we assume that motion detection is always triggered at the end of each motion capturing period and no transition to standby mode is performed during the same frame.

<u>Case number 1:</u> continuous motion in the time period of 1 second.

- Algorithm 1: $\frac{30}{60} * P_{MC} + \frac{30}{60} * P_{IC} = 1.92mW$, where P_{MC} and P_{IC} are the power consumptions measured in [5] for the Motion Capturing mode and the Imaging mode respectively.
- Algorithm 2: $\frac{7}{30} * \frac{P_{MC}}{2} + \frac{23}{30} * \frac{P_{IC}}{2} = 1.39mW$

Our efficient algorithm succeeds in lowering the power consumption by : $1.92 - 1.39 \approx 0.43mW$

<u>Case number 2:</u> motion is never detected in the time period of 1 second.

- Algorithm 1: $P_{MC} = 25.3\mu W$
- Algorithm 2: $\frac{15}{30} * \frac{P_{MC}}{2} + \frac{15}{30} * P_{ST} = 7.27\mu W$, where P_{ST} is the power consumption measured in [5] for the Standby mode.

Our efficient algorithm succeeds in lowering the power consumption by : $25.3 - 7.27 \approx 18.03\mu W$

The power consumption we calculated above corresponds to a time period of 1 sec. Considering that a vast majority of moves we encounter in nature, even the sharpest ones rarely take less than 1 sec to be completed, our system is highly likely to be consuming power that bounces between the two estimated values for each case for the whole duration it is powered on. Of course we cannot exclude the case that motion is detected for less than 1 sec but it's not so easy to be repeated periodically with a period of less than 1 sec so that our system operates in a different way for a significant amount of time.

5. Conclusion

Any system, no matter how efficient it might be, can still be improved. In this paper, a device presented in [5] was chosen to be modified in order to take advantage of its high potential for efficient motion capturing due to its image sensor which featured, in-pixel non-volatile analog memory utilizing a c-axis-aligned a-b-plane-anchored crystal In-Ga-Zn-O (CAAC-IGZO) that constitutes the base for a FET with extremely low off-state current. This state of the art hardware needed to be complemented with an automatic process for the transition among different operating modes.

To summarize, in this paper we propose two algorithms for the aforementioned automatic process, using a controller that switches between the modes by activating and deactivating the appropriate components. The major goal that is reached is the transformation of a machine with very good characteristics to a complete motion capturing system. The initial approach was rather simplistic but still according to our estimations the results can be very pleasing. The second algorithm we proposed is a more sophisticated solution that can minimize power consumption while maintaining almost the same functionality.

As a proposal for future work, the authors believe there is interest in investigation regarding the optimization of the second algorithm which if modified correctly can be even more efficient for specific environments and applications.

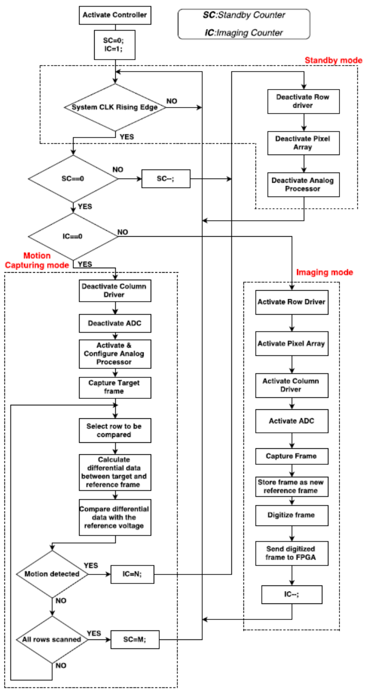

Fig.3 Flowchart for Implementation 2 (30fps)

References

[1] P. Lichtsteiner et al., JSSC, Vol. 43, pp. 566-576, (Feb. 2008)
[2] U. Malik et al., JSSC, Vol. 42, pp. 2187-2196, (Oct. 2007).
[3] S. Park et al., ISSCC, pp. 126-127, (Feb. 2014)
[4] K. Kato et al., JJAP, Vol. 51, No2, 021201, (Feb 2012).
[5] T. Ohmaru et al., ISSC, pp.118-119, (Feb. 2015).