# Fault Tolerant Display Image Data Manipulation Unit for SOP

***Jaehee You, Hyungoo Lee***
**School of Electrical Eng., Hongik Univ., Seoul, Korea**
**Tel: +822-320-1657; E-mail: jaeheeu@hongik.ac.kr.**

## Abstract

*A display panel image data manipulator for SOP or SOG is presented. It is capable of all the shift operations for MPEG decoders, graphic processors and controllers as well as data pack, merging, bit split and reformation operations to improve speed and memory utilization. To alleviate poly-Si low yield, redundancy based fault recovery scheme is introduced utilizing regular structure.*

## 1. Introduction

Mobile and portable devices such as mobile handset, PDA, PMP, portable DVD, mobile PC and visual MP3 have display panels with tight constraints on size, power, thickness, cost, weight and high reliability. This leads to system on panel (SOP) and system on glass (SOG) integration as one of promising technical directions next to system in package (SIP). LTPS is known to be a suitable processing technology candidate for SOP. However, it has inherent drawbacks such as large channel length, low mobility, large leakage current, kink effect problems, and the lack of uniformity on threshold and mobility. So far, SOP cases such as charge pump, scan driver and analog data driver in Sanyo and Sony, scan driver and digital data driver in Philips, scan driver and digital data driver in Toshiba, 6 bit digital data driver in LG, and DC-DC converter in NEC, have been reported for past years. This year, 9 bit DAC [1] and LTPS panel sized column driver [2] are integrated in display panels. [3] shows a complete SOP on glass substrate including a simple Z-80 like processor and a timing controller. In the future, frame memories, a graphic controller, interfaces and an image processor or engines for MPEG are expected to be realized with SOP.

This paper describes a fault tolerant SOP image data manipulation unit architecture and implementation issues for mobile display panel systems. It can process all the conventional shift operations and data pack, merging, bit split and reformation operations for SIMD (Single Instruction Multiple data) image computing with low hardware overhead. So far, these are computed with software as in ARM by several ALU and shifter operations or multimedia processors such as VIS in SUN UltraSparc [4], MMX in Intel, AltiVec in Motorola, which are not suitable for SOP due to tremendously high hardware complexity regardless of 5X computing gain. However, the proposed manipulator can improve computation speed more with less hardware for bit split arithmetic and save memory space for image data with bit widths not aligned with the word-length of arithmetic unit such as 24 bit pixels. In addition, it can reduce pixel data transfer rate between a frame memory and a data driver, which causes one of major power sources in mobile display system. In addition, to overcome inherent low yield problem of TFT device technology, the proposed architecture is designed to have the regular structure of barrel shifter with newly developed redundancy methodologies mainly composed of interconnections having relatively high yield compared to TFT devices.

## 2. Instructions, Architecture and Operations

To enhance computation speed with low hardware complexity, SIMD is required since image processing is basically composed of identical operations on many pixels. This requires data manipulations to overcome the difference in data width between arithmetic unit and pixels with multiple precisions. To efficiently process the above, the following instructions are proposed and implemented. They are PACK4B (PACK4HW): packing four bytes (half word) from four words with 32 bits/word at a fixed position to generate a word (two words), PUSH24P: packing four 24 bit data from four words at a fixed position to generate three words, POP24P: the reverse operation of PUSH24P and finally PACK64: shifting a 64 bit

data with arbitrary distances and extracting a 32 bit data. The applications of the proposed instructions are as follows. PACK4B (PCK4HW) facilitates 8 (16) bit pixels storage into a frame buffer to save memory space and display controller read from the frame buffer. PUSH24P and POP24P also save frame buffer space and ease the display controller pixel fetch especially for 24 bit pixel precision. As shown in Fig. 1, each of four 32 bit words contains 24 bit pixels right justified with 8 bit waste leading to 25% waste of total memory space. PUSH24P assembles most significant bytes of four words and places them in a word and the remaining four 16 bits of pixels are packed with two 16 bits in a word style, which gets rid of memory waste completely and achieves 100% frame or line memory utilization for SOP. The manipulator can again reformat the data back to four 24 bit pixels to be sent to display panel through display controller by POP24P.

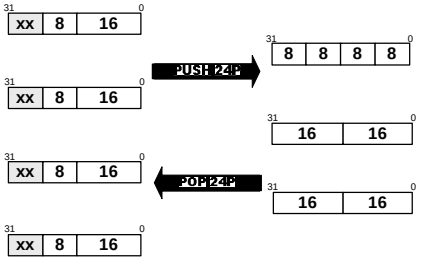



Fig.1. PUSH24P and POP24P operations.

In addition, the conventional data manipulation instructions are supported in the proposed data manipulator.

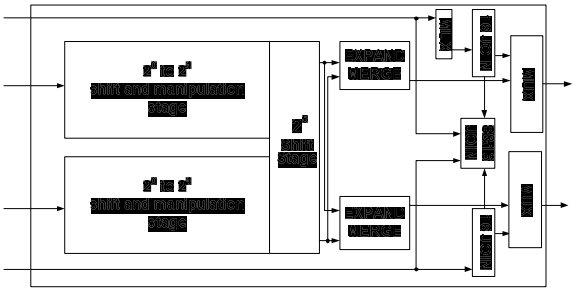


Fig. 2. Data manipulator architecture.

They are additional pack instructions, such as extracting four bytes at a fixed position of 64 bit data to generate a 32 bit word (FPACK16), taking a byte from a 32 bit word and 24 bits from another word to generate a 32 bit word (FPACK32), extracting 16 bits from two 32 bit word to generate a 32 bit word (FPACKFIX), taking a byte pixel and expanding to higher precision pixels into a double word (EXPAND) and merging two words into a double word (MERGE) to process the pixels with SIMD as well as bit split addition and multiplication. All the above instructions are implemented in proposed data manipulator with little overhead added to a conventional barrel shifter.

Fig. 2 shows proposed data manipulator architecture capable of processing 64 bit data. It consists of barrel shifters capable of $2^0$ to $2^4$ shift stages and a $2^5$ shift stage for conventional shifts as well as the proposed data manipulation commands. It can align the pixel with all bit widths in a word or double word format as well as put pixels in arbitrary bit position for comparison, partitioned addition and multiplication arithmetic with 100% utilization. Some of the required data manipulation paths conflict with conventional barrel shifter paths, which require additional shifts with fixed distances. Although it has a few additional interconnections for the data manipulations, the paths are designed not to degrade the regularity in a barrel shifter to facilitate redundancy replacements for fault tolerance. This can be done with MUXes and additional interconnections, which does not degrade the SOP yield greatly.

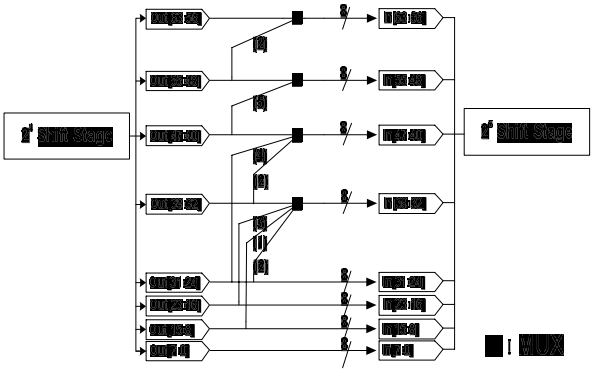


Fig. 3. Extra interconnection paths.

Extra shift path $2^3$ for FPACK16, FPACK32, $2^4$ for FPACK16, FPACKFIX and $2^3 + 2^4$ for FPACK16 are designed. In Fig. 3, the additional interconnections for the extra fixed shifts paths between $2^4$ and $2^5$ shift stage are shown. (1), (4), (5) are for FPACK16, (2), (5) are for FPACK32 and (3), (4) are used for FPACKFIX. For example, FPACK32 uses four (2), (5) paths to shift 8 bit distance for the corresponding four fixed $2^3$ shifts required.

The yield of poly Si devices is lower compared to conventional CMOS. To cope with this major obstacle for SOP, efficient redundancies are included in each shift stage with short paths without significant additional delay with regularity. The each shift stage shown in Fig. 1 is composed of 4 cross point switch blocks and each block has an additional switch array block, which can be efficiently shared since they are identical switch array. Fig. 4 shows fault tolerant cross point shifter for only $2^0$ shift stage.

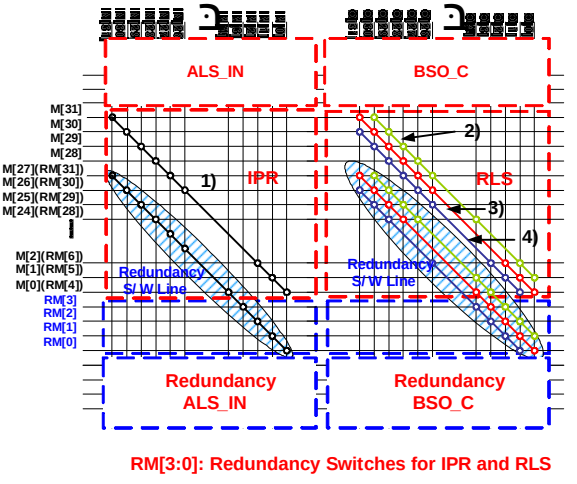



Fig. 4. Fault tolerance scheme for $2^0$ shift stage.

First block (RLS) processes conventional arithmetic and logic shifts. 2), 3), 4) in RLS are the crosspoint lines for right shift, bypass without the shift and left shift, respectively. Second block (IPR) reroutes RLS input paths (IN[31:0]) with 1). In case of four 8 bit or two 16 bit block shift, the data at the border of the blocks need not to be fed into neighbor blocks. For example, conventional 16 bit $2^0$ left shift routes IN[7] to O[8] across the border. 1) shifts and routes the data with 8 bit block-based style and the data at the border of the blocks are cut out. Third block (ALS_IN) generates 0 or MSB for 32, 16 and 8 bit partitioned shifts. Fourth block (BSO_C) reroutes 0 or MSB from ALS_IN to appropriate output ports. The above four blocks have redundancy blocks (redundancy ALS_IN, redundancy BSO_C in Fig. 4) to overcome faults inside each block as shown in Fig. 4 for low yield SOP process realization. ALS_IN and BSO_C as well as IPR and RLS (i.e. 1) in IPR and 2), 3), 4) in RLS) are functionally connected so if faults occurred in any of cross point switches, the faults are replaced with redundancy ALS_IN and redundancy BSO_C below. For example, if the cross point through 4) in RLS has faults at the intersection of O[29] and M[28], all of 2), 3), 4) are replaced with the redundancy cross point lines at RLS (shaded area). Since, the inputs into RLS should be rerouted, the cross points through 1) in IPR are replaced with the redundancies (shaded area at four grids below). In detail, faulty IN[28] → M[28] → O[29] path is replaced with IN[28] → M[24] (i.e. RM[28]) → O[29]. Therefore, overall path is IN[31:0] → {M[27:0] (i.e. RM[31:4]) : RM[3:0]} → O[31:0]. M[27:0] are shared with RM[31:4] and only the addition of RM[3:0] can completely recover the faults in IPR and RLS. Extra interconnections and MUXes added to the conventional barrel shifter are 376 and 300 (2:1 Mux), respectively. Fig. 5 shows the experimental data manipulator FPGA layout for functional verification purpose. The cross points implemented with full custom layout can further reduce the chip area of the redundancy overhead.

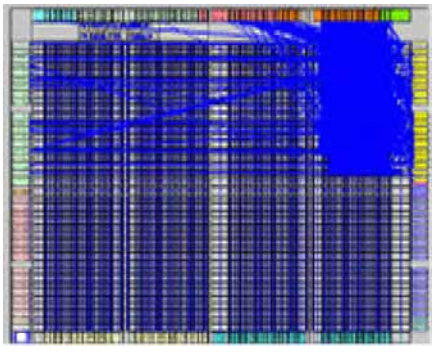
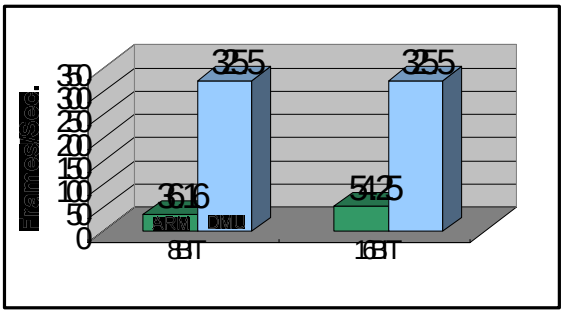


Fig. 5. Data manipulator FPGA implementation.

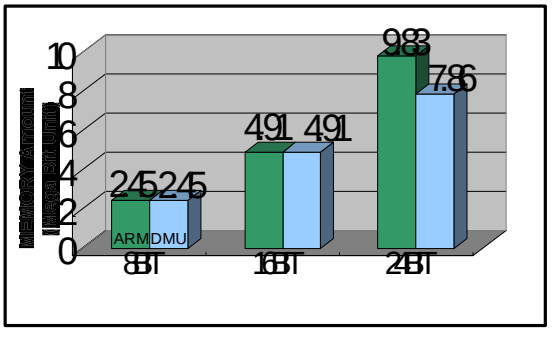
The image data flow is as follows: Image raw data → DMU →→ Repeated (Frame buffer → Graphic operations → Frame buffer) →→ DMU (only for 24 bit pixels) → Graphic controller & Line memory → Display Panel.

## 3. Performance and Advantages

The proposed data manipulator has many application areas in SOP as well as in conventional display systems. It can be used as a separate or embedded unit inside SIMD style double 32 bit MPEG decoding engine, and graphic processing unit or controller in SOP. All the data are tightly packed with the manipulator then they can be processed with bit partitioned image processing unit, transferred by reduced data rate through buses and stored in frame memory, which enhances memory and bus utilizations with low power in case image processing unit, frame memory and DMU are implemented with SOP.




(a) Frames/sec. comparisons




(b) Memory amount for 8, 16, 24 bit pixels

Fig. 6. Performance comparisons.

The proposed manipulator can be extended to process the shuffling commonly used in FFT or DCT and the edge handling for a frame boundary processing. Combined with customized load and store of embedded frame memory, image processing capability is greatly enhanced with little hardware.

Assuming that byte or a half-word, 24 bit pixels are stored in a 32 bit frame buffer and fetched by a display controller, the performance of the proposed data manipulation unit (by PACK4HW or PACK4B) is evaluated and compared with that of ARM alone (by software). The time for the process: 8, 16 bit pixel fetch from the memory – packing – frame buffer write – display controller - display panel is as shown in (1).

$$T_{total} = T_{dmu} + T_{memory} + T_{lcd} \quad (1)$$

$T_{toal}$: Total time

$T_{dmu}$: Time for pixel packing

$T_{memory}$: Time for frame buffer pixel fetch and write

$T_{lcd}$: Time for display controller

Since $T_{memory}$ and $T_{lcd}$ are the same for both ARM and the proposed data manipulator, only $T_{dmu}$ is evaluated and compared as shown in Fig. 6. 8, 16 bit pixel processing performances are increased 6 and 9 times respectively, which alleviates the drawback of low speed SOP process with low hardware overhead. As shown in Fig. 6(b), 25% memory saving can be obtained for 24 bit pixel case with the proposed PUSH24P and POP24P. Furthermore, the proposed DMU has a hardware-efficient redundancy scheme based on regular cross point structure for fault tolerance, which are mainly composed of interconnections having higher SOP process yield.

## 4. Acknowledgements

This research was supported by a grant (F0004110) from the Information Display R&D Center, one of the 21st Century Frontier R&D Program funded by the Ministry of Commerce, Industry and Energy of the Korean Government.

## 5. Reference

[1] I. Knausz et. al., A 250uW 0.042mm2 2MS/s 9b DAC for Liquid Crystal Display Drivers, pp. 172-173, ISSCC (2006).

[2] O. Ishibashi et. al., Amorphous Si SOG 15inch XGA 20MHz LTPS CMOS TFT Panel sized column driver, pp. 176-177, ISSCC (2006).

[3] T. Matsuo et. al., CG Silicon Technology and Development direction on System on Panel, pp. 856-859, SID (2004).

[4] Z. Mou et. al., VIS-based native video processing on UltraSPARC, Int. Conf. on Image Processing, Vol. 1, pp. 16-19, (Sep. 1996).