Parallel pipelined histogram architecture via c-slow retiming


It is advisable to refer to the publisher’s version if you intend to cite from the work. See Guidance on citing.

To link to this article DOI: http://dx.doi.org/10.1109/ICCE.2013.6486871

Publisher: IEEE

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.
www.reading.ac.uk/centaur

CentAUR
Central Archive at the University of Reading
Reading’s research outputs online
Abstract—A parallel pipelined array of cells suitable for real-
time computation of histograms is proposed. The cell architecture
builds on previous work to now allow operating on a stream of
data at 1 pixel per clock cycle. This new cell is more suitable for
interfacing to camera sensors or to microprocessors of 8-bit data
buses which are common in consumer digital cameras. Arrays
using the new proposed cells are obtained via C-slow retiming
techniques and can be clocked at a 65% faster frequency than
previous arrays. This achieves over 80% of the performance of
two-pixel per clock cycle parallel pipelined arrays.

I. INTRODUCTION

Image analysis based on histograms is abundant and well
used in many consumer applications [1]. An array of cells to
perform the computation of \( m \)-bin histograms that takes \( k \)
pixels per clock cycle offers to gain a speedup factor of \( k \).
Such a design was proposed [2], but required a sensor or
processor supplying four pixels per clock cycle to get a
speedup of four. Many embedded microprocessors consist of
8-bit data buses and consequently are able to supply one pixel
per clock cycle [3,4]. In order to exploit this property, a
histogram solution using C-slow retiming to create two sub
streams of computation derived from a dataset arriving at one
pixel per clock cycle is proposed here.

This paper briefly explains the principle of C-slow retiming
and applies C-slow to fully develop the proposed cells in
section II before presenting final conclusions. The essential
result is that the proposed design provides speed-up while also
facilitating easier interfacing to camera sensors or
microprocessors compared to other designs.

II. C-SLOWING RETIMING

C-slow retiming is a method used to reduce the critical path
delay in digital circuits especially when feedback loops exist
[4]. Every register in the datapath is replaced by \( C \) registers
and then all registers are moved around on the critical data
paths using a retiming algorithm. C-slow retiming separates
the calculation performed in the original datapath into \( C \)
instances. Fig. 1 shows an excerpt of the datapath of a
histogram cell previously presented [2] that includes a
feedback path (left), its C-slow version by a \( C \) factor of two
(center) and after retiming to get a C-slow retimed version
(right). A simple example using Fig. 1 illustrates the principle
of retiming. For input sequence \( u = 3, 5, 4, 1 \) the left diagram
in Fig. 1 produces \( r = 0, 3, 8, 12, 13 \); the leading zero reflects
the register delay with output \( r \) being the running accumulation
on input \( u \). The diagram on the right of Fig. 1 gives \( r = 0, 0, 3,
5, 7, 6 \) for the same input \( u \). The output corresponds to the
accumulation as if there were two separate input streams: \( u_0 =
3, 4 \) and \( u_1 = 5, 1 \) and as such the output has been separated
into \( r_0 = 3, 7 \) and \( r_1 = 5, 6 \); and the two interleaved into output
\( r \). In general C-slow retiming creates \( C \) interleaved streams of
computation and as such also requires \( C \) input data streams.
For practical reasons related to the design, only the factor \( C =
2 \) is considered in the rest of the discussion.

Fig. 1. Pipelined datapath with feedback (left), C-slow with \( C = 2 \) (center)
and C-slow retimed (right).

A. Discussion on the C-slow effects

Fig. 1 demonstrates that the process of re-retiming reduces the
critical path delay from the cost of a binary adder and the
associated logic to being either the time of the binary adder or
the logic time whichever is longer. The downside is that the
register count may increase significantly (by a factor of \( C \)). For
example, compare the diagrams in Fig. 1 as retiming proceeds
from left to right. The final architecture is also influenced by
the specific places within the datapath where the registers are
finally placed (due to datapath widths.) So, \( r' = 0, 0, 0, 3, 5, 7 \)
(Fig. 1 right) and \( r + r' = 0, 0, 3, 8, 12, 13 \) implies the cost of
an extra adder is required to merge the two streams; this is
unavoidable in the context of the example and also applied to
computation of histograms.

B. C-slow retimed histogram processing cell

A C-slow retimed (\( C = 2 \)) processing cell for the computation
of histograms is presented in Fig. 2. This follows straightforwardly
from the above discussion and the histogram
cell presented [2]. The new registers introduced by C-slow
retiming are shown in gray. The mechanism to read bins out
from the cell in a pipelined fashion has been omitted for
simplicity.

The cell structure above the Logic block has been preserved
except for the fact that C-slowing by a factor of two replicates
the pipeline registers moving data left to right in the original
design. It should be appreciated that the structure looks very much like an instance of Fig. 1. It follows that, the separation of the computation into two streams does require the use of the extra adders as seen at the very bottom of Fig. 2. The critical path delay for the cell is now either the comparison followed by the block of Logic or the adder. Without C-slow retiming the critical path is due to the compare-logic-accumulation chain.

### C. Results and analysis

A design was tested using ASIC technology of 35 microns giving the results in Table 1. Although the C-slow cell is only around 25% faster than the standard pipelined cell [2] the real advantage comes when the cells are arranged as an array. A pipelined array accepting 2 data items per clock cycle computes the histogram in \( n/2 + m/2 \) clock cycles with each cell processing two bins; \( m/2 \) is the latency. The C-slow cell in Fig. 2 requires two data items per clock cycle. Assume the cell of Fig. 2 is fed with every other data item (from an input dataset of \( n \) items) every clock cycle: half the items go into the array stream piped through \( x^1in \) input and the other half into through \( x^2in \) input. As a result an array processes a single data item per clock cycle. Thus, the histogram is computed in \( n + m \) clock cycles (the latency is \( m \) even for cells computing two bins since each cell in Fig. 2 has a latency of two clock cycles.) As \( n \gg m \) for typical image sizes, latency can be ignored for a quick analysis.

### III. CONCLUSIONS

A new array of cells computes \( m \)-bins histograms on streams of one pixel per clock cycle at over 80% of the performance of a pipelined array, working on streams of two pixels per clock cycle. This is due to arrays of C-slow cells achieving 65% faster clocks than previous pipelined arrays. The proposed array is consequently better suited for when camera sensors or microprocessors are limited to supply one pixel per clock cycle.

### REFERENCES


