TWO-DIMENSIONAL MICROPIPELINES:
FOR PARALLEL TO SERIAL DATA CONVERSION

George S Taylor and Gerard M Blair

The two-dimensional micropipeline is introduced − for data conversion between bit-parallel and bit-serial formats at a constant input data rate. By using micropipelines, the architecture has independent clock rates for the data input and output, and can be designed with level sensitive latches.

1. Introduction

This letter introduces the two-dimensional micropipeline architecture in the context of a converter between bit-parallel and bit-serial format, which may be used, for instance, as an interface between a bit-parallel environment and a bit-serial accelerator module. In general, the two environments may be independently clocked or purely asynchronous.

Figure 1 shows a possible system application architecture with a synchronously clocked bit-serial module. Bit-parallel data (an n-bit word) is pushed vertically onto a stack at a rate determined by the signal *inClk*; when sufficient (m) words are accumulated to feed the central, bit-serial module, the data is shifted-out horizontally - while the bit-parallel data continues to be added to the buffer region at the top of the micropipeline stack. When the n bits of the data words are output, the data accumulated in the buffer is allowed to fall into the lower section. A similar arrangement allows the bit-serial output from the central module to be accumulated and output at a regular data rate from the system. The clock which drives the central module need not be the same as *inClk* (or *outClk*) − and the sizes of the micropipeline regions are a function of the word-size, the block-size and the relative rates of the *push* and *pull* signals.

In this letter, we will describe the parallel-to-serial converter in detail; the corresponding serial-to-parallel converter can be designed along similar lines.

2. Micropipeline controller

The design for the two-dimensional micropipeline is based upon the two-phase latch controller recently published by the authors[1] and shown in Figure 2. The controller consists of two transparent latches (one positive enabled and one negative enabled) and an *xnor* gate. The inputs: *Rin* and *Aout*, are the handshaking signals for the previous and next controller; and the outputs: *response* and *enable*, are the handshake output (in both directions) and the control signal for the transparent data latches.

3. Two-dimensional micropipeline

The parallel-to-serial converter is shown in Figure 3 for transforming a stream of (n =) 3-bit words in blocks of (m =) 4 words into bit-serial format. The central array is composed of (positive enabled) transparent latches in three distinct regions:

A. forming a micropipeline buffer with inputs from only above

B. performing the conversion using connections from both left and above

C. holding the most-significant bit of the data words from the last conversion while the next block of data is being assembled; this may be unnecessary in some applications.

The use of transparent latches with micropipeline control, rather than edge-triggered flipflops with a synchronous clock and enable signals, leads to high saving in area particularly for large arrays.

The latch array is controlled by two sets of micropipeline controllers: the left and bottom arrays producing the horizontal and vertical latch control signals respectively. Regions A and B together form a vertical micropipeline driven by the left controller array; regions B and C form a horizontal micropipeline driven by the bottom controller array.

The following description applies to the case when the number of words in region B is even; otherwise a slightly different design is necessary.

Initially, the vertical micropipeline is set to *empty* so that data can enter: this is achieved by setting all the internal latches in the left controller array to *zero* (signal not show in Figure 3) which raises all the horizontal *enable* signals. Conversely, the horizontal micropipeline is set to *full* to preclude data movement along it: this is achieved by setting alternate pairs of latches in the bottom array to *zero* and *one* (as indicated in Figure 3) which lowers all the vertical *enable* signals.

Data is now input (on both phases of the *push* signal) until region B is *full*. Because of the handshaking protocol, this condition cannot be tested simply by the *and* of the low *enable* lines since this occurs on two or more adjacent registers when data is being transfered between them. Instead, the condition: *request pending* (RP), is monitored on all but the top register in region B. When a latch controller is in RP, it has not only received data but also acknowledged its receipt and the preceding controller has requested to pass on new data. As a feature of the controller design in Figure 2, this state is uniquely indicated by different values on the input and output of the first (positive) latch in the controller. This could be tested using $(Rin \oplus x)$, but the delay of the latch may cause glitching when the latch is transparent; instead it is tested by $(\overline{enable}.(Rin \oplus x))$ which can be optimized in alternate controllers as $\overline{(enable + Rin + \bar{x})}$ and $\overline{(enable + \overline{Rin} + x)}$ due to the way in which the signals evolve as region B "fills up". The *full* signal is thus tested as the logical *and* of the RP functions. Since the array fills up from the bottom, a simple chain of *and* gates implements the function (as shown) without a delay penalty dependent upon the array size.

Once region B (as part of the vertical micropipeline) is *full*, the data may be read out in a word-parallel, bit-serial format using the bottom controller array on both phases of the *pull* signal. If the left-most *Rin* control signal is connected to *gnd* then the pipeline will *empty* as data is extracted - and this condition can be tested as the *and* of the vertical *enable* lines. In this case, the bottom array will need to be reset (to *full*) before new data can enter the region B in the vertical micropipeline. If, however, the "alternative ending" (shown in Figure 3) is used, then the array will remain full; this is the simpler arrangement if the "consumer" module calculates itself when to cease "pulling" data since it avoids the need to reset the bottom array.

Once the bit-serial data output is complete, the elements in the left controller array driving region B are reset to *zero* (to be empty) and the data in the buffer region A will start to fill region B. The most-significant bits of the data words remain in region C until the next set of *pull* signals.

During the reset of the left array, it is necessary to isolate the input to the top controller of region B (shown in Figure 3) to prevent contention on the adjacent resetting latch; an implementation by a wide pass-transitor (driven by the reset singal) may minimize this delay. In both controller arrays, the completion of the reset can be determined by changes in their respective output control signals: *full* and *empty*, provided that all elements are reset in parallel. Further the directions of the data flow and the *and*-chains shown in Figure 3, coupled with the handshaking protocol, ensure that these control outputs are glitch free.

4. Conclusions

We have demonstrated a two-dimensional micropipeline architecture in the application of parallel-to-serial data conversion between regions with unrelated clock signals. By using a micropipeline, the architecture allows for a constant input data rate despite the fact that the data is processed in blocks. Signals to indicate completion of each phase of the conversion are derived to provide a control interface.

References

1. G.S. Taylor and G.M. Blair, ''Reduced complexity two-phase micropipeline latch controller'', *ESSCIRC 97*, pp. 304-307, Sept 1997.

The authors are at The Department of Electrical Engineering, The University of Edinburgh The King's Buildings, Edinburgh, EH9 3JL, Scotland, UK.

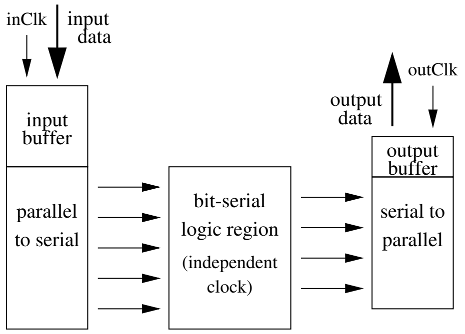

Figure 1: possible application for parallel-to-serial conversion

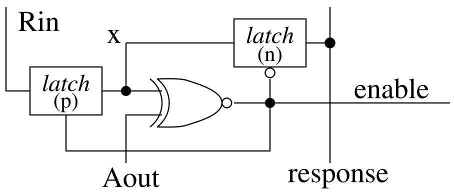

Figure 2: two-phase micropipeline latch controller[2]

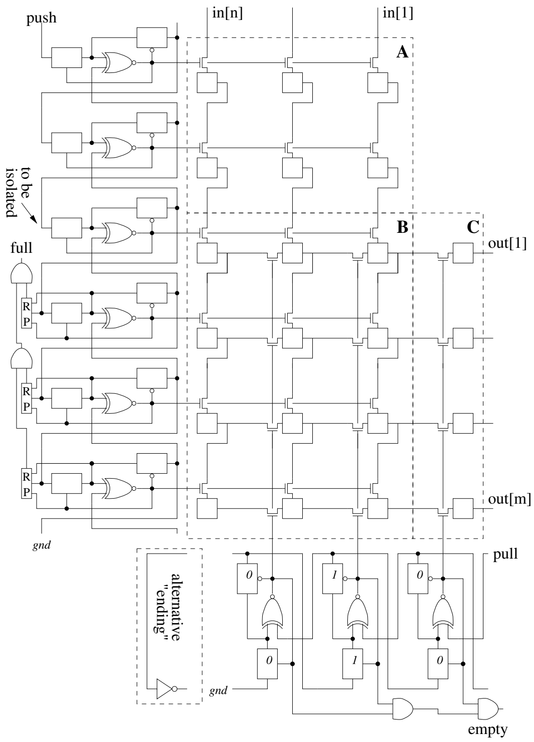

Figure 3: two-dimensional micropipeline architecture