

PSeuDoFFIL: Power Saving Datapath FiFo Insertion Logic

John Lofgren, Synopsys, Inc.
3501 Quadrangle Blvd. St 200, Orlando, FL 32817, jlofgren@synopsys.com

Josefina Hobbs, Synopsys, Inc.
1301 South Mopac Expressway Bldg 4 Suite 200, Austin TX, 78746, josefina@synopsys.com

John McCardle, ATI Technologies, Inc.,
3501 Quadrangle Blvd. St 375, Orlando FL 32817, jmccardl@ati.com

1. Abstract

A technique is described to dynamically limit power dissipation in heavily pipelined digital circuits. Stages in the pipeline are clock gated based upon validity of data at that stage. As the pipe approaches a stalled or inactive state, branches of the clock tree are progressively disabled stage by stage. The technique has the additional advantage of removing gaps or “bubbles” in the data streams, improving the latency of the system.

Keywords: power bubble circuit pipeline logic

2. Introduction

High throughput processing engines necessitate constant data input to deliver peak performance. In complex systems the type of data needed to complete a processing task is varied and thus requires fetches from multiple sources to begin execution. The system architect attempts to balance the system by scheduling fetch requests hundreds of cycles before the data is required. The goal is to provide just in time arrival of data at the processing unit thus maximizing the throughput and minimizing the area overhead. Data caches and FIFO's are used as an area efficient method of hiding the fetch latency in data flow designs. These caches and FIFO's are linked by pipeline stages that transform raw input formats into a format suitable for computation. The figure below shows a simple example of such a piped system. Data flowing into pipes A and B is buffered, transformed, re-buffered and then combined to generate a result that is further transformed and buffered for output to the next stage.

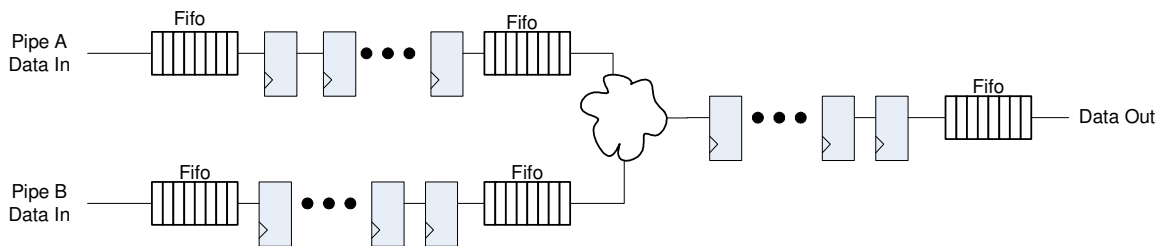


Figure 1: Pipelined Data Architecture

Data dependencies often occur between pipes and fetch latencies vary depending upon the data type. Locally, this can result in complete or intermittent stalls of one or more pipelines. For instance, long latencies getting data to the input of Pipe B will cause Pipe A to stall once its FIFO's are at capacity(see in Figure 1). Even though Pipe A is not moving data, it is constantly clocking the registers and FIFO

controllers. When data begins flowing in Pipe B, it can come in finite bursts resulting in null (lack of data) bubbles in the output data pipeline. The bubbles in the pipeline can draw the same amount of power as valid data while transitioning from one stage to the next.

At the system level, multiple threads are being executed in parallel with the issuance of new threads dependent upon the completion of current threads. Latencies at this level can leave complete sections of the architecture fully and/or intermittently inactive due to either starvation or stall conditions.

Power consumption in data flow systems has many sources: leakage current of the transistors, registers changing state, propagation of new state values through random logic between register stages and the switching current of the clock tree. Typical power reduction techniques attempt to minimize the switching activity in a system. These techniques are applied to different parts of the circuit. One method requires a priori knowledge of the activity level in a functional unit block. If a unit has no scheduled activity, the clock tree to the block is disabled. This is the most efficient method since it disables both the local clock tree as well as the functional logic. In typical semiconductor circuits, the clock tree accounts for 40% to 75% of the power budget. In heavily pipelined designs, advanced notice of inactivity is generally available. However, the threshold is binary: active or inactive. There are a number of circuits that function somewhere in between. Another technique is to use clock gating at the local level. This method identifies existing logic signals as clock enable inputs to flip flop primitives. The power reduction capability of this technique is lower since not all registers are converted to clock enable flip flops and the clock trees are still active.

The method introduced in this paper allows fine grained clock gating of pipelined circuits based upon the propagation of data valid bits. The technique disables the clock tree for entire rows of data bits. In addition, the clock gating logic automatically squishes bubbles in a data pipeline.

3. Description of Technique

PSeuDoFFIL stands for Power Saving Datapath FiFo Insertion Logic. PSeuDoFFIL is implemented by automatically inserting complementary logic into the system that recognizes where valid data is placed and takes steps to move the valid data forward a register at a time until a stall condition is reached, rather than stalling the whole pipeline with bubbles and valid data left in static, interleaved positions.

The algorithm begins by taking advantage of a data valid bit that exists in every pipeline and travels in lock-step with the data. Figure 2 is an example of a data pipeline. Between each block is an arrow with "Pipe Data" that represents the movement (and typically a transformation of some sort) of data at each clock edge. Above that are the DV (data valid) bits that indicate where interesting and "valid" information exist. The Pipe Data and DV values travel forward as a matched pair, left to right, in lock step at each clock edge.

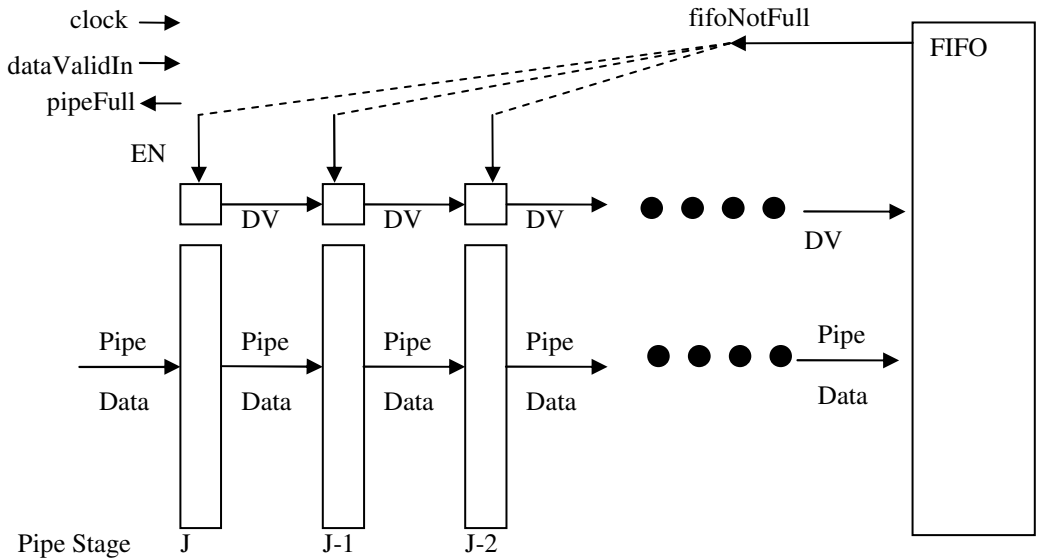


Figure 2: Pipeline Without Bubble Removal

In the case of Figure 2, the `fifoNotFull` signal indicates the pipeline can still move data forward into available FIFO memory. `fifoNotFull` is connected directly to the `EN` (enable) signals for each stage. The inverse of this signal becomes `pipeFull`, which tells the source of the pipeline data to stop sending data. When the FIFO is full, every stage in the pipeline freezes. Now consider this same generic circuit after logic is inserted to independently disable or enable the forward movement of data in the pipe.

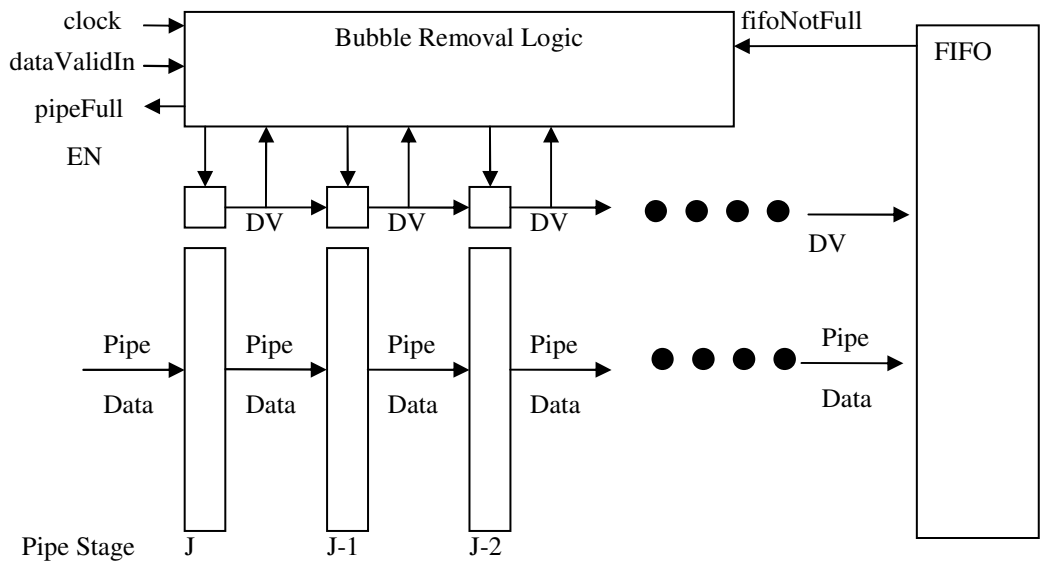


Figure 3: Pipeline with Bubble Removal Logic

Figure 3 shows how the Data Valid (DV) bits and their paired data values are controlled in their forward movement one stage at a time by a block called “Bubble Removal Logic”. The basic function of the bubble

removal logic is to enable data to move forward during a FIFO full condition if there is NULL data in the next stage of the pipeline.

The meaning of pipeFull changes in this case too, now it is a function of fifoNotFull and the existence of any bubbles in the pipe. As long as there are bubbles or the fifoNotFull signal is active, the pipeFull signal will not go active, so the pipeline can continue to absorb new data until all bubbles are gone.

Now the biggest advantage of bubble removal may be introduced: by employing clock gating as a means of enabling data movement, the largest source of power drain, the clock tree distributed to the data registers, may be switched off. There are three general cases:

1. **EMPTY:** Valid data does not exist in the pipeline
2. **STALL:** The destination is not accepting data and the pipe is full.
3. **BUBBLE:** The register ahead of a bubble is disabled

Bubbles contain no interesting information, so clocking the register ahead of a bubble is unnecessary. Only registers ahead of valid data that can move forward need be clocked. In essence, local clock tree power dissipation is data driven.

Detailed Logic

The logic that is built to do bubble removal considers the global FIFO full signal and the local data valid bits as it determines clock gating. Realize the last register in the pipe is numbered as register 1, and the rest are numbered in ascending order for each step upstream. The clock enable is defined by:

```
assign clkEn[j] = reset
  || ( dataValid[j+1] && ((~fifoFull) || (~&dataValid[j:1]))); (Eq.1)
```

The logic in Equation 1 tells each register to capture data if there is a bubble downstream or the FIFO is not full, and there is a valid piece of data in the register upstream. Reset overrides all of these conditions and turns the clock on regardless.

The Data Valid bits looks like this:

```
assign preValid[j] = reset
  || ( dataValid[j+1] && ((~fifoFull) || (~&dataValid[j:1]))
  || ( dataValid[j] && ~((~fifoFull) || (~&dataValid[j:1]))); (Eq.2)
```

The Data valid bit is more complex as shown in Equation 2. It is set if there is a valid data bit upstream, and bubbles or the FIFO is not full downstream, or it retains its value if there are no bubbles and the FIFO is full. Reset overrides all of these conditions.

4. Power Saving Opportunities

The figure below shows an example of a data underflow condition:

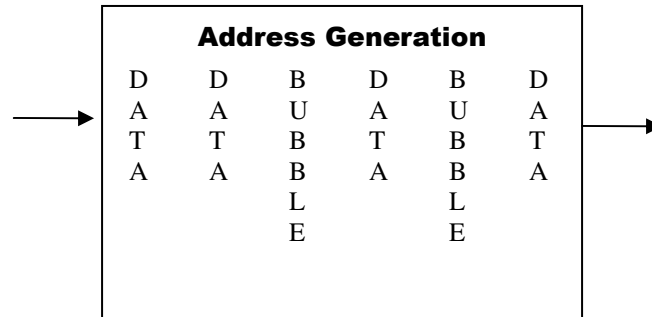


Figure 4: Data Underflow

In the data underflow condition, the registers to the right of each bubble do not need a clock, so 2 registers meet the BUBBLE case and need no clock. The power needed to charge and discharge the capacitance on the clock wires that feed 2 of the 6 registers may be saved. Some register stages can be 1000's of bits wide in complex systems. Depending on the clock network layout, the process technology and the use of leakage-saving standard cells, turning off 33% of the clock distribution paths can save as much as 20-30% or more of the total power of the circuit. The power savings may be much higher or lower in cases where the width of the pipeline registers varies widely from stage to stage.

In summary, as the number of bubbles in the pipe increases, quantum reductions in power are realized until nearly zero clock switching power is manifested at the EMPTY case. At any given moment clock power is minimized in quantum steps proportional to register size as data rates decrease.

After adding PSeuDoFFIL, if the datapath downstream from a pipeline is stalled, the pipeline will gradually fill with values, from right to left. Simultaneously quantum decreases in power are realized as many, then few, then 1, then no new values enter the pipe at the point it is full. Fully stalled, the clock switching power would again approach zero. At any given moment clock power is minimized as data rates overload the pipe.

Example

A sample circuit was devised to demonstrate the concept. A multiplier-accumulator with 2x32 bit inputs and 64 bits of output is synthesized into 6 pipe stages to run at 600Mhz. It contains over 8K combinational cells, and more than 500 flip flops. A clock buffer tree is inserted in the physical floorplan. A Vera simulation test bench exercised the circuit through thousands of clock cycles and the power is estimated based on switching activity metrics.

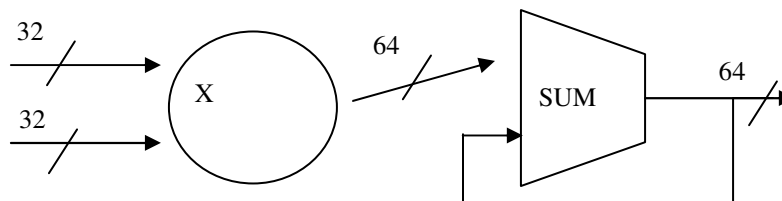


Figure 5: Demonstration Circuit

Using the technique described herein, the dynamic power is reduced by 23% when the data rate has a Valid to Null ratio of 50% (**1.3183 mW** down to **1.0118 mW**).

Pipeline Bubble Removal

Figure 6 illustrates the state of a 6-stage pipeline as a series of data values are interleaved with bubbles. Each DATA word represents a rank of registers holding valid data. Each BUBBLE word represents a register containing Null data. The DATA and BUBBLE words move in lock step from left to right:

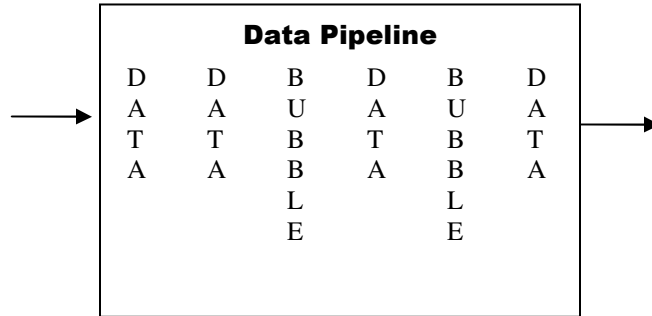


Figure 6: Pipeline Bubbles

Now suppose the processing element located downstream (to the right) reached a stall condition, at which time the address generator had to be stopped for a moment. While stopped, if the bubbles could be squished, the data would move closer to the end of the pipeline, shortening the time it takes to reach the output and thereby reducing latency, as shown in Figure 7:

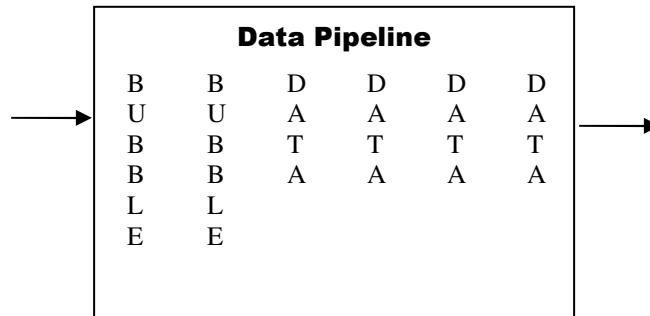


Figure 7: Bubbles Squished

Furthermore, these bubbles at the back of pipeline or data allow additional opportunities for data to enter the pipeline until the address generator is full of valid data, effectively allowing upstream processors to work further ahead of the flow, hiding latency and increasing throughput, as shown in Figure 8.

Since pipelined logic blocks like those described here can comprise up to 50% or more of image generation or image processing systems, the opportunity to take advantage of bubble removal latency reductions can be large. However, the manual insertion of the bubble removal logic into the pipeline logic is too complex, time consuming and error prone.

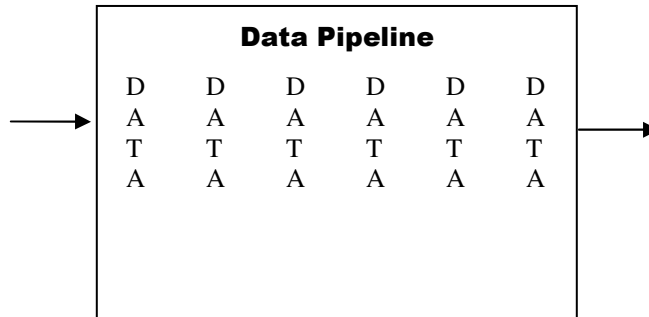


Figure 8: Bubbles Removed

The use of automated design techniques to implement bubble removal provides rapid high quality insertion into the system. This automated approach can also realize quantum power reductions in low data rate conditions (static scenes in image generators) and quantum power reductions in data overload conditions (rapidly changing scenes in image generators). System designers can also perform area-reduction/performance-improvement trade-offs which allow dedicated FIFO sizes to be reduced because the pipeline flip-flops may become storage devices that behave like FIFOs.

PSeuDoFFIL implementation in the tool flow is slightly more complex but can be 100% automated through synthesis scripts and results in identical pin-out to any stall-able pipeline with a data valid bit. There are limitations imposed on the pipe length. As with any stall signal, the fanout cone of the STALL signal back down the pipeline will eventually exceed the ability to settle within a clock period. This condition should be detected by application of a static timing tool. The verification flow is also slightly more complex. Equivalence checking will need to deal with additional clock gating logic using pre-verified libraries or vector-based checking. An RTL simulation model can possibly be synthesized using a FIFO- and counter-based wrapper/model that limits the number of values in the pipeline and output FIFO to the same number of registers in the pipelined logic.

5. Conclusion

PSeuDoFFIL is a generally applicable, simple and automated means for manifesting power reduction and performance enhancements in pipelined circuit designs. The technique has the biggest impact on wide data paths that require multiple clock buffers per rank. Overall peak power is not reduced by the PSueDoFFIL technique since the peak is generally determined by worst case system usage: full data pipes, continuously active IO. PSueDoFFIL will reduce the Total Dynamic Power for the system since it automatically disengages the clock tree to register banks in stasis.

References

http://www.ece.ucsb.edu/Faculty/Parhami/text_comp_arch.htm

<http://my.opera.com/allbits/blog/show.dml/167704>