CHAPTER 6

QUASI-DELAY INSENSITIVE TEMPLATES FOR
DISCRETE COSINE TRANSFORM MATRIX VECTOR
MULTIPLICATION

6.1 INTRODUCTION

This research apparently explains low power design techniques for
discrete cosine transform (DCT) and inverse discrete cosine transform
(IDCT) circuits which are conspicuously applicable for low bit rate wireless
video systems. The techniques includes skipping IDCT computation of
blocks, skipping DCT computation of low energy macroblocks with all
coefficients equal to zero, by using lower precision gating the clock, reducing
transitions and constant multipliers in the data path. The proposed DCT and
IDCT circuits reduce power dissipation on an average 94\% over baseline
reference circuits. This research also demonstrates the design of efficient
asynchronous bundled data pipeline for the matrix vector multiplication core
of discrete cosine transforms. The architecture is so designed for both zero
and small valued data which yields high average performance and low
yielding power. Novel data dependent delay lines with integrated control
circuit to efficiently bring out speculative completion sensing. In order to gain
confidence in the timing margin, extensive post layout backend timing
analysis was performed. Usually the asynchronous circuit consist of a set of
functional component that communicate using a set of handshaking protocols. Its uniqueness in the market is well known due to its versatile features. Parallelism in the handshaking protocol, data encoding across the channels and the degree of timing assumptions needed to ensure correctness are the predominant properties (Salomon and Klar 1993), (Sparso and Nielsen 1993).

In the several data compression and decompression standards are essential, for the two-dimensional (2-D) discrete cosine transform (DCT) and inverse DCT (IDCT). This will include H.261, H.263, JPEG, and MPEG. Power and performance are respectively low and higher during implementations of the DCT and IDCT which supports increase in demand for faster data rates and longer battery lifetimes in portable multimedia devices.

In the input vector core operation carried within both of these applications will be the multiplication of a constant matrix. For example a matrix–vector multiplier, typically implemented with a set of multiply-accumulation units. High-performance, low-power or together getting explored in many of the DCT/IDCT synchronous and asynchronous designs. IDCT input data is zero-valued while observing the typically significant fractions. By skipping operations that involves zero-value data, the motivated development of a data-driven IDCT saves power. Proposed data compression methods that maintain only significant data portions is an additional advantage of small-value data also suppress the remaining insignificant portions. It offers no average-case performance benefit since they are synchronous and these designs yield only low average power. The position of sign adjusts dynamically because an asynchronous width adaptive data architecture that activates operations only in significant data portions.
The power consumption is reduced in the architectural approach with good average-case performance. It is well known for being area of expensive since it represents implementations based on the Quasi-delay-insensitive asynchronous design style (Teymourzadeh et al 2010). DCT designs are effectively optimized in both zero and small numbers yielding both good average-case power and performance by proposing an area-efficient asynchronous (Sunan Tugsinavist and Peter Beerel 2005).

To save the power in groups of bit-slices involving only sign-extension bits are dynamically turned off by partitioning the datapath into a staircase bit-sliced bundled-data architect is the key idea. To achieve the high average case performance, it has been proposed efficient speculative completion sensing delay lines with integrated control circuitry that activate shorter delay lines when various bit slices are turned off. To control the complex nonlinear pipelines efficiently and simplifying the logic design problems three novel control circuit templates to be proposed.

The first control template, it is a straightforward adaptation from pre-charged full buffer (PCFB). While applying to bundled-data designs it suffers from large control overhead inherent in the underlying handshaking protocol. Reduced stack pre-charged half buffer (RSPCHB) is the second control template which will significantly reduce this overhead.

All designs were laid out with an identical datapath in a 0.35- m CMOS process. The best asynchronous design has 30% higher average throughput with comparable energy consumption by simulating 2.3 V with typical DCT input statistics. This chapter thus demonstrates that for matrix-multipliers within DCTs, full-custom gated-clocking may achieve similar
power savings as that of synchronous techniques. In the standard synchronous techniques it is not significant increase in average throughput however the asynchronous techniques can yield.

6.2 ASYNCHRONOUS CHANNEL

In the protocol for taking information in terms of tokens like data, control or mixing of these two, a communication channel will consist of bundle of wires from sender to receiver. One wire per bit of information will be encoded in tokens of hand shake protocols. A token is validated when a request line (req) is used to convey the receiver. Once the token is received an acknowledge line (ack) is used to convey the sender on information. Request lines are not needed since in a 1-of-N channel uses N wires to encode log2N bits. To encode one bit of information in dual-rail encoding two wires will be used. In the communication channels both two and four phase handshaking protocols will be present.

6.3 QDI Templates

As shown in Figure 6.1 Reduced Stack Pre-Charged Half Buffer (RSPCHB) was referred from the new QDI templates. Instead of an OR gate NAND gate will be used before the output inverter RCD block is optimized by tapping its inputs has been well noted. The non-improved performance in the concurrency will be reduced by RSPCHB templates by facilitating the removal of internal enable signals. The LCD and RCD are companied using a C-element to generate acknowledgment signal Lack in PCFB templates. This shows the results of validity and neutrality of both input and output data is supported in the integrated handshake protocols. This system removes the
need for function block to be weak conditioned. Use of the en signal replaces the requirement from necessary to more concurrent. In the case of a join, output may get generated from non-weak conditioned function block while data received from one of the input channels. Therefore the RCD of the join asserts its output.

Reception of data, evaluation of asserts in both its LCD and RCD outputs and asserts acknowledgement signal in any of the subsequent stages. Once the en is asserted it will precharge and receive acknowledgement through the join. Once acknowledged in the input stages it will assert en signal get delayed in the precharge of the circuit. This delay prevents the precharge from triggering the RCD to de-assert. Once generation of acknowledgment signals from any stage subsequent to the join has arrived and been acknowledged, this will safely remove the en signal. The performance will not be impacted due to the delay of the acknowledgement even though the join been the performance bottleneck for its subsequent stages. The overall performance will improve significantly and reduces the capacitive load due to the lack of an LCD and reduced stack size of the function block is the major advantage.
In Figure 6.2 shows the STG of (signal transition graph) when a left token arrives (L+), the evaluation of the R_gen dynamic logic blocks occur, produces a valid output token, firing of the local clock (GR i - ri R i+), and simultaneously the ILCD block sense the token arrival(ilcd-). Following that, the IRCD block locates the right data is valid (ircd-), which results in the desertion(Le-) of the left enable, and the internal state is to be reset(en-). After the resetting of the left data (L-), the ILCD block detects that the data are null (ilcd+) and along with the reset of enable, it causes the left surroundings, which allows it to send a new token, though the right environment is slow or stalled, hence avoiding a important presentation penalty. The right enable will be DE asserted by the right situation (Re-), allowing the R_gen blocks to pre-charge. The right enables (Re+) is re asserted by the right environment and concurrently, the internal enable to be reasserted (en+). This in turn allocate the R_gen blocks to reevaluate in respond to a new token.
As shown in Figure 6.3 the abstract STG of the RSPCHB is the derivative of the analytical expression for the timing margin associated with this isochronic fork. In which $F_1^e$ denotes the Function $F_1$ at evaluation state. At pre-charge state, $F_1^P$ represents the Function $F_i$, $C_1^+$ depicts the completion detector $C_1$ state is changed to 1, $C_1^-$ signifies the completion detector $C_1$ state is changed to 0, the Right Completion Detector represents the RCD+, RCD state is changed to 1 and RCD- denotes the Right Completion Detector RCD state is changed to 0 of pipeline stage 1.
Figure 6.3 STG of the abstract RSPCHB protocol

6.4 MATRIX–VECTOR MULTIPLICATION ARCHITECTURE

The discussions of the proposed architecture and the review of matrix multiplication are dealt in detail in this section.

6.4.1 Matrix–Vector Multiplication

In this chapter a detailed study of matrix multiplication operation and the proposed architectural designs are clearly discussed.

\[
\begin{bmatrix}
  z_0 \\
  z_1 \\
  z_2 \\
  z_3
\end{bmatrix} =
\begin{bmatrix}
  x & x & x & x \\
  v & u & -u & -v \\
  x - x & -x & x & x \\
  u - v & v & -v & u
\end{bmatrix}
\begin{bmatrix}
  y_0 \\
  y_1 \\
  y_2 \\
  y_3
\end{bmatrix} =
\begin{bmatrix}
  (x \cdot y_0) + (x \cdot y_1) + (x \cdot y_2) + (x \cdot y_3) \\
  (v \cdot y_0) + (u \cdot y_1) - (u \cdot y_2) - (v \cdot y_3) \\
  (x \cdot y_0) - (x \cdot y_1) - (x \cdot y_2) + (x \cdot y_3) \\
  (u \cdot y_0) - (v \cdot y_1) + (v \cdot y_2) - (u \cdot y_3)
\end{bmatrix}
\]

The matrix–vector specification that the implementations can be expressed as follows
6.4.2 Asynchronous Pipelined Architecture

In Matrix Vector multiplication two novel speculative delay matching controllers are involved. One of the delay matching controllers is asymmetric delay line controller (ADLC) and one for a symmetric delay line controller (SDLC). Based on the select control lines each controller functions same as to one of its input signals. The power is significantly reduced since both the delay line controllers are compact. At the algorithmic level, the basic methodology of implementing each matrix vector multiplication is strategized into four iterations that have been taken, one per column of the matrix. In iteration, $i^{th}$ element of $y$ is multiplied by the $i^{th}$ column. This involves multiplying an input with three varied coefficients and possibly inverting the result, thereby it is clearly shown as the use of three distinct hardwired multipliers. The results of each iterations are stored in accumulators, after the fourth iteration, these results are written and then it is reset in preparation of the forth coming input vector. The Figure 6.4 shows the proposed the novel five-stage pipelined architecture at the architectural level.

The upper portion of the picture explains asynchronous controllers communicated with the datapath and other controllers with four-phase handshaking signals rather than a global clock. To obtain low power, the datapath is executed using single-rail static logic. Numerous power optimizations looking advantages of small-valued input statistics are applied. The general idea is to dynamically deactivate groups of bit-slices that contain only sign extension bits (SEBs). In the datapath, the MASK unit identifies the bit-slices of input data that contains non-SEBs and the zero data unit (ZDU) detects the special case in which the data is zero. If the input data is zero, the ZDU unit asserts a zero_detect signal that disables the entire computation operations. To handle nonzero input data, the multipliers and accumulators consist of groups of partitioned bit-slices that are selectively activated by the MASK unit.
Figure 6.4  Matrix multiplications with five stage asynchronous pipeline
Figure 6.5  Proposed asynchronous fine grained carry-save hardwired multiplier
In particular, the mask signals are used to deactivate non-SEBs by forcing them to zero via the input ANDing logic and are sent to control matched delay lines in the multiplier stage. Additionally, the same mask signals when latched are ORed with their previously registered versions. The resulting mask signals (OR) identify the bit-slices of the accumulators that contain non-SEBs and matched delay lines in the accumulator stage. The result of the accumulator stage is fed to the partial sign bit recovery (PSBR) logic which expands the sign bit of newly activated bit-slices in the accumulator to ensure that both inputs to the accumulator have the same number of activated bit-slices. In the last iteration, the full sign bit recovery (FSBR) logic improves the suppressed zero bits of accumulator results to get the correct final result. The fine-grain hardwired multiplier is based on a bit-partitioned carry-save multiplier, illustrated in Figure 6.5.

The carry-save multiplier’s critical path is mainly by the side of the final, vector-merging adder, which has been executed as a bit-partitioned ripple-carry adder for two reasons. First, ripple-carry adders consume significantly lower power than faster (e.g., carry select or bypass) adders. Secondly, while ripple-carry adders have relatively long worst case delay, the bit-partitioning of the multiplier array (including the ripple-carry adder) leads to very good average case delay for this application. The staircase-patterned bit-slices, as illustrated by the dotted lines in Figure 6.5, agree to the adders to be dynamically configured for different input bit-widths. For example, if the first two bit-slices are make active, the multiplier behaves exactly as a typical multiplier that handles 9-bit inputs.
Figure 6.6  Asynchronous controller of DCT matrix-vector multiplication
6.5 CONTROLLER ALTERNATIVES

Both synchronous and asynchronous controllers can be integrated with the same data path. Implementation of Clocking conditioned asynchronous design as same as a gated-clocking synchronous controller in order to compare with this asynchronous design. As shown in Figure 6.6 the asynchronous controllers are implemented using the PCFB and RSPCHB templates, which yield two varied asynchronous designs for comparisons. Since the DCT matrix–vector multiplier explained is a medium-grain pipeline, over the RSPCHB template and power-efficient asymmetric delay lines are used along with the PCFB based design in order to compare, at the same time symmetric delay lines are used with both RSPCHB based design.

6.6 DESIGN FLOW, EXPERIMENTAL RESULTS AND COMPARISONS

In this design the hierarchical design flow has been used. Each behavior block in verilog explains a constructed architectural specification behavioral specification of the design is précised. At this juncture, in order to verify functional correctness of this architecture, simulation is used. Each block is decomposed into gate level verilog. The next step is that transistor-level implementation is to map every gate in this library. In order to verify the constraints about timing and to ensure correctness level simulations are performed. The final step is that a layout is generated hierarchically. At this juncture, correctness and timing analysis are carried out by extracting wire capacitance and thus taking into consideration of the impact of interconnection delays.
The design methodology in this chapter targets fine grain pipelining and small cells, where the forward latency is two gate delays (Montek Singh et al. 2001). Fine grain pipelining is attained by dividing the processing blocks to even smaller cells where each cell has its own input and output completion detector. When the multiplier completes its processing and generates a 16 bit output, the output completion detector identifies it and combined with the input completion detector produces the acknowledgement. The multiplier will admit a new input only after the first stage of the multiplier has finished processing. Therefore the throughput is bounded to how fast the multiplier can multiply two numbers, produce the acknowledgement and then reset. As in the synchronous case the throughput of the multiplier can be increased by further pipelining the multiplier. In asynchronous design, this can be completed by constructing the multiplier using small number of cells such as adders and other logic gates which have their own input and output completion detectors.

6.6.1 Throughput and Energy Comparisons

The first work evaluates asynchronous designs using the PCFB control with two different delay lines, a standard asymmetric delay line (PCFB_{ASYM}) compared with one using the power efficient delay line (PCFB_{SYM}). In this design, five different inputs which activate zero to all bit-slices and the simulation results are verified.
Table 6.1 Comparison of PCFB based designs using Different Asymmetric Delay Lines

<table>
<thead>
<tr>
<th>Test Pattern</th>
<th>PCFB ASYM</th>
<th>PCFB SYM</th>
<th>% of Lower Overall energy</th>
<th>% of Lower controller energy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Power (mW)</td>
<td>E/Cyc (ns)</td>
<td>Power (mW)</td>
<td>E/Cyc (ns)</td>
</tr>
<tr>
<td>Zero</td>
<td>11.5</td>
<td>7.2</td>
<td>91.5</td>
<td>12.1</td>
</tr>
<tr>
<td>Bs1</td>
<td>42.5</td>
<td>16.4</td>
<td>721</td>
<td>41.7</td>
</tr>
<tr>
<td>Bs2</td>
<td>45.3</td>
<td>18.4</td>
<td>842</td>
<td>42.9</td>
</tr>
<tr>
<td>Bs3</td>
<td>44.5</td>
<td>21.6</td>
<td>1054</td>
<td>46.8</td>
</tr>
<tr>
<td>Bs4</td>
<td>45.4</td>
<td>23.7</td>
<td>1108</td>
<td>44.5</td>
</tr>
</tbody>
</table>

Table 6.1 exhibits average power, cycle time, and energy per cycle. The result propose that the performance of design using (PCFB\text{SYM}) controls and yields up to 2.2% lower energy than (PCFB\text{ASYM}) control. However the controller put in as little as 4.8 % of the overall energy, the (PCFB\text{SYM}) controller yields up to 46% lower energy than the (PCFB\text{ASYM}) controller. Therefore, it has been preferred the (PCFB\text{SYM}) control as the nominated design using PCFB control for the follow up evaluations.
Next, Table 6.2 shows the design compares two different asynchronous designs: one using PCFB controllers and another one using RSPCHB controllers. Table 6.2 illustrates the worst case forward latency (FL).

**Table 6.2** Comparison of PCFB and RSPCHB based on timing analysis, forward latency, Overhead and Cycle Time

<table>
<thead>
<tr>
<th>Test Pattern</th>
<th>SYNC</th>
<th>ASYNC-PCFB</th>
<th>ASYNC-RSPCHB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \Gamma (\text{ns}) )</td>
<td>( E/\text{Cyc} )</td>
<td>( E^2 )</td>
</tr>
<tr>
<td>Zero</td>
<td>20</td>
<td>95</td>
<td>39</td>
</tr>
<tr>
<td>Bs1</td>
<td>20</td>
<td>671</td>
<td>270</td>
</tr>
<tr>
<td>Bs2</td>
<td>20</td>
<td>775</td>
<td>311</td>
</tr>
<tr>
<td>Bs3</td>
<td>20</td>
<td>981</td>
<td>394</td>
</tr>
<tr>
<td>Bs4</td>
<td>20</td>
<td>1015</td>
<td>407</td>
</tr>
</tbody>
</table>
Table 6.2 illustrates the worst case forward latency (FL), cycle time (t) and controller overhead (OH) of two designs for each type of inputs from zero to all bit-slices activated. The result implies that the RSPCHB template is consistently operate 19–38% faster than PCFBs. Furthermore, the worst case latency of the accumulators is 18.8 ns, and it has been set the synchronous cycle time to 20 ns. To quantify performance–power tradeoff, it has to set up ten test cases as follows. The first of seven test cases, each having 20 input vectors, are simulated using Nanosim on the extracted layout. Of these, the first five test cases demonstrate average cycle time and energy comparison of zero data and four different bit-slices activated starting from zero data and then bit-slice one (bs1) to bit-slice four (bs4).
Table 6.3  Comparison of PCFB and RSPCHB based design (Control and Datapath) with detailed Timing and energy Analysis

<table>
<thead>
<tr>
<th>Test Pattern</th>
<th>PCFB</th>
<th>RSPCHB</th>
<th>% FAST VS PCFB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FL(Ns)</td>
<td>Δ(ns)</td>
<td>OH(Ns)</td>
</tr>
<tr>
<td>Zero</td>
<td>3.2</td>
<td>6.9</td>
<td>3.5</td>
</tr>
<tr>
<td>Bs1</td>
<td>12.5</td>
<td>16.6</td>
<td>3.9</td>
</tr>
<tr>
<td>Bs2</td>
<td>14.5</td>
<td>18.6</td>
<td>3.9</td>
</tr>
<tr>
<td>Bs3</td>
<td>17.3</td>
<td>21.5</td>
<td>4</td>
</tr>
<tr>
<td>Bs4</td>
<td>19.6</td>
<td>23.8</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 6.3 describes the experimental results. The cycle time (τ) and energy per Cycle (E/cyc) is clearly described in the first two columns for each design. The third column for every design indicates the Eτ² product when compared to the synchronous design. The results are displayed in the following points. Primarily, since the similar data path is applicable to all designs, the differences in energy are due to the difference in energy utilized by the controllers. The least energy is consumed by the clock gating synchronous controller, subsequently followed by the asynchronous RSPCHB controller, and the asynchronous PCFB is the next step in it. Added to this, the results display the efficiency of bit-slice partitioning in that less energy is consumed by smaller bit slice than a larger one.
Specifically, a zero input data utilizes far less energy compared with the others. Secondly, it is quite clear that in the asynchronous designs a larger bit-slice operates slower than a smaller one. Consequently because of its large control overhead, the PCFB controller is not advantageous considering the speed aspect over the synchronous design while more than two bit-slices are in active pace at the same time, when all the bit-slices are active RSPCHB controller is slower. Moreover, compared to the synchronous design the results of the bound analysis clearly states that, the cycle time is between 16%–35% faster for the RSPCHB design and the cycle time is between 6% slower and 13% faster for the PCFB. Finally, the simulation along with the three real images depicts that the characteristic performance gain over synchronous design is around 35% for the RSPCHB-based design and it is 10% for the PCFB-based design. The next step is that for low power the asynchronous designs can tradeoff performance. In this design provides 14%–35% higher performance with a 3.5%–10.5% energy penalty without voltage scaling. Energy can be quadratically reduced when the power supply is scaled.
6.7 SUMMARY

The use of a competent asynchronous bundled-data pipeline design methodology on matrix–vector multiplication for DCTs is clearly demonstrated in this chapter. Both high average performance and low power can be obtained by architectural optimizations that take advantage of zero and small-valued data, typical in DCT and IDCT. Low overhead integrated control circuits capable of handling nonlinear pipelines and enabling high average throughput are supposed to be created by novel control circuit templates and data-dependent delay lines. When compared with comparable gated-clocking synchronous counterpart recommend that 30% higher throughput with negligible energy overhead and it has a 49% better metric that can be obtained in the proposed asynchronous design.