300MHz design methodology of VU for Emotion Synthesis

Takayuki Kamei Hideki Takeda Yukio Ootaguro Takayoshi Shimazawa
Kazuhiko Tachibana Shin'ichi Kawakami‡ Seiji Norimatsu† Fujio Ishihara
Toshinori Sato Hiroaki Murakami Nobuhiro Ide Yukio Endo
Akira Aono†† Atsushi Kunimatsu

System ULSI Engineering Lab. TOSHIBA Corp.
† EDA and DFT Grp. Micro & Custom LSI Design Dept. Micro & Custom LSI Div. TOSHIBA Corp.
†† System LSI development Grp. Toshiba Microelectoronics Corp.
‡ Information System Dept. TOSHIBA Microelectoronics Corp.
580-1, Horikawa-cho, Saiwaiku, Kawasaki 210-8520 JAPAN
Tel: +81-44-548-2523 Fax: +81-44-548-8324 Email: kamei@sdel.toshiba.co.jp

Abstract— With advance in silicon technology and system integrating technology, utmost care for RC–delay must be taken in high performance LSI design. This paper describes the new design method applied to the Vector Unit(VU) implemented in the recently announced high-performance 3D-graphics engine, 'Emotion Engine'. The key features of the design method are the followings:

- **careful functional design of the VU architecture**
- **design hierarchy consistency between RTL descriptions and floor plan**
- **accurate estimation of wire load in pre–layout static timing analysis**

With the above design features, the VU is has achieved the operating frequency of 300MHz.

I. Introduction

A new 300MHz 3D-graphics engine called 'Emotion Engine' (EE) has been developed by TOSHIBA and Sony Computer Entertainment, which has 13Mtr. and 15.02mm × 15.04mm die area with 0.18μm CMOS technology [1],[2]. With advance in silicon technology and system integrating technology, special attention needs to be paid to RC delay in high performance LSI design.

Several design methodologies for estimating accurate wire capacitances in pre–layout stages have been proposed[3] to improve speed quality of synthesized logics and reduce the time required for design iteration. But they deal with wire capacitance only and lack of wire resistances falls in inaccurate pre–layout timing analysis.

We have developed a new method which can estimate not only wire capacitance but also wire resistance by predicting a wire shape and the estimated RC delay is effectively used for pre–layout timing analysis. This method enables us to obtain accurate pre-layout timing result and reduce the design turn-around time (TAT). The improved TAT for the pre–layout timing analysis is one sixth the post–layout analysis.

In this paper, we discuss the methodology employed for the design of the Vector Unit (VU) embedded in EE, the next-generation video game console. First, we describe the VU architecture and the outstanding features of VU design in Section II, the basic concept of our timing design in Section III, the details of our timing design in Section IV, and the effectiveness of our design method in Section V.

II. Overview of VU architecture

VU is a vector calculation engine implemented in EE , the next generation video game console. For realizing "Emotion Synthesis" and high quality 3D-graphics, VU has two processing modes.

In the co–processor mode, VU is tightly connected to the core processor as a co–processor by 128bit bus . VU has capability of executing enhanced core processor instructions which are prepared for 4-parallel floating point vector calculations, and of processing massive floating point calculations.

The other mode is called VLIW mode in which VU works as a stand-alone VLIW processor and all functional units operate fully in parallel. The mode makes high performance 3D-geometry calculations possible.

The VU block diagram is shown in Fig.1

VU has 4 fMACs, 1 fDIV, 1 iALU, 32×128bit floating point registers(fGPR), 16×16bit integer registers(iGPR), a random number generator unit(RANDU), and a load–store unit(LSU). An elementary function unit(EFU) can be attached optionally. Every floating point register consists of 4 fields(x,y,z,w), each of which stores a 32bit single-precision floating–point number. VU has a flexible bypassing and broadcasting network between the fGPR and the arithmetic units. Hence, any floating point operand can be transferred to every functional unit through the bypass(f–Bypass) and broadcasted to all fMACs.

The VU pipeline is shown in Fig.2.

The pipeline consists of 6 stages for all operations except for those of fDIV(9 stages). The number of stages is decided so that programmers and compilers can code optimized programs by

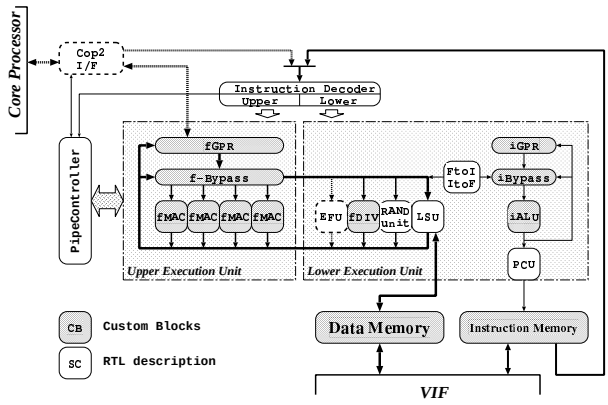

Fig. 1. a VU block diagram

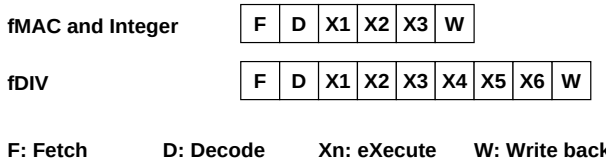

Fig. 2. a VU pipeline

pipelining method easily. The software pipelining is essential for high quality 3D-graphics and Emotion Synthesis. The throughput of each operation except for that of EFU and fDIV is only one cycle and instructions are executed efficiently. On geometry transformation, for example, 4×4 matrix and 4 elements vector are multiplied in eight cycles.

III. Basic concept of our timing design

In order to achieve the target frequency, it is important to study the effect of wire loading on timing design.

With respect to conventional logic synthesis tools, wire loading is modeled by static distribution of R factor and C factor as a function of fanout. This works fine as far as RC delay for one fanout is not very dominant. However, the RC delay factor is becoming larger and larger as the process technology evolves.

This is a vital issue because wires with heavy RC parasitics have a serious impact on timing design, but, at the same time, the number of such heavy wires occupies only a small fraction of the total wires counts. Even though the effect of RC delay of these wires cannot be neglected, it is not appropriately reflected in wire load models simply because these wires are minorities.

Fig.3 shows RC distribution in VU, where gray bars and white bars stand for the number of wires inside functional blocks and the total number of wires, respectively. Most of the wires tends to have smaller resistance values as the connection

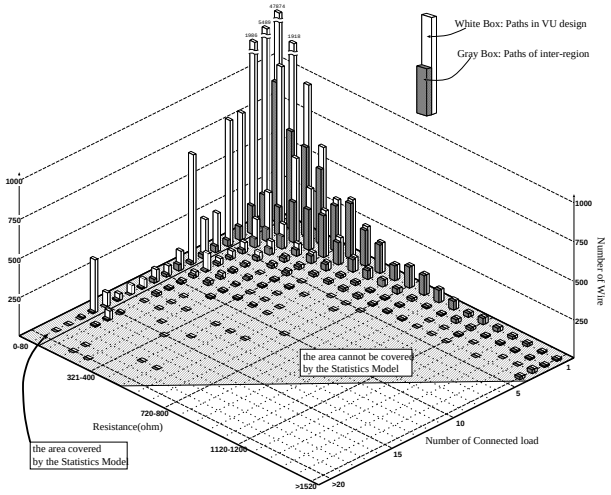

Fig. 3. a distribution of wire resistance in VU

count decreases, but still, a few wires have large resistance despite they have point-to-point connection.

In order to deal with this problem, wires are categorized into two groups in this scheme by dividing VU into several functional blocks and categorizing intra-block wires and inter-block wires. Each functional block has about 4,000 gates and logic synthesis is performed by the block basis. This the functional block is also defined as a module in RTL so that the inter-block wires can be identified even on RTL level (shown in Fig.4). Upon floor planning, one block is treated as a unit, so that conventional wire load model is applicable to intra-block wires. Wire load of the inter-block wirings is estimated from their length separately from the wire load model for the intra-block wires, and the wire load is annotated to timing analysis.

Moreover, by making use of this estimated wire loads not only for logic synthesis but also for pre-layout timing analysis, timing can be predicted in the early stage of the design flow with higher accuracy in short period of time.

The design flow details is discussed in the next section (see also Fig.5).

IV. Detailed Timing Design

A. RTL design and floor planning

The VU pipeline is is designed suitably for 3D floating point operations. For the high speed design, it is important to take advantage of the pipeline structure and to shorten wire length in partitioning the functional blocks.

The fetch and decode are allowed to consume only 1 cycle each in the VU pipeline. VU f–Bypass is quite complicated because it is not only bypassing operands but also broadcasting them This means that there is only one cycle assigned each to bypassing and broadcasting operands.

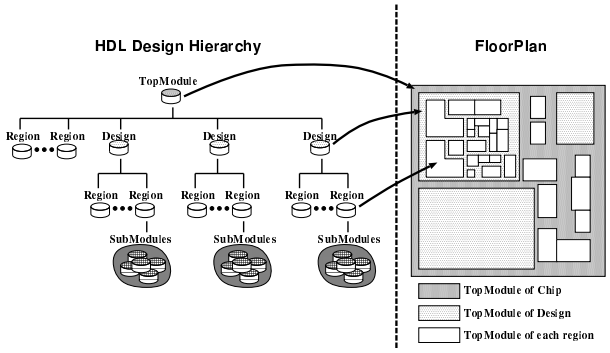

Fig. 4. HDL and Floor-Plan hierarchy correspondence

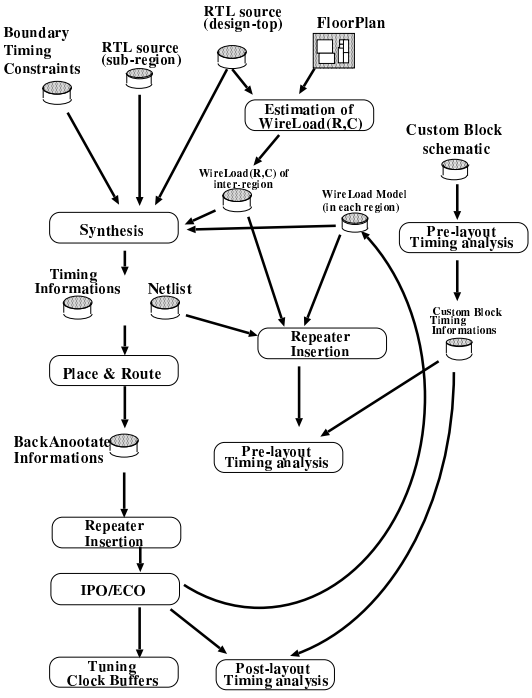

Fig. 5. Design Flow

In making full use of the pipeline style, instruction decode, bypass control, hazard detection and pipeline stall control must be completed in a cycle. Thus, all bypasses, a program counter unit(PCU) and some buffer trees of pipeline control are implemented in custom blocks to decrease the delay of data transfer, and the PCU and all integer data paths are grouped into one function block.

To achieve shorter wire length, the functional blocks are floor-planned as shown in Fig.6. The well-arranged floor plan eliminates lengthy interconnects traveling from one end of VU to the other and reduces the wire load by increasing the percentage of the wires connecting neighboring blocks.

B. Wire load estimation

In accordance with the way the synthesis tool and timing analysis tool handle the delay, wire load estimation are made in two different ways:

- For logic synthesis
  The most important point is to map adequate size of drivers at its interface to other blocks. This does not require high accuracy and therefore only the capacitance is handled as the wire parasitic. The capacitance estimation is made by measuring the distance from the interface to the most distant ends of the functional blocks connected to the intra-block wires. The estimation tools are coded in perl language. The inputs of tools are RTL codes and floor-plan geometry information, and the output is a wire capacitance list. It takes about 5 minutes to estimate all wire loadings in VU.

- For timing analysis
  Unlike the one for logic synthesis, both resistance and capacitance must be estimated from wire shape and locus. Detailed estimation of the wire shape and location, however, requires real place-and-route operation and consumes too much time. In this scheme, wire load is estimated in the following manner(shown in Fig.7):

  1. locate the center of gravity of the wiring from block position and block connection information assuming that all the pins are placed at the center of the block.
  2. draw either a horizontal or vertical line passing the center of gravity.
  3. draw perpendicular line from the center of each block until it reaches the above line.
  4. calculate resistance and capacitance of each segment of the lines.

  In this way, we can estimate not only the wire capacitance but also the wire resistance. Thus we can increase the accuracy of estimation.

  The wire shape estimation described above is well-researched([4],[5],et al.) and proved to be effective for the small and high speed chip designs. But we do not apply the calculation of wire delay in VU design

directly because wire delay information is not necessary for our repeater insertion. Instead, more optimal delay can be calculated by the static path analysis after repeater insertion(see the following subsection).

The estimation tool is coded in C language. The inputs of the tool are RTL codes and floor-plan geometry information, and the output is a file included consisting of the wire information of resistance, capacitance, and shape for static timing analysis. It only takes about 2 minutes to estimate all wire loadings in VU.

The reason for using the center of gravity for locating buffer is that the actual start point and end point are not known until logic placement. This becomes a problem for big function block like memory macros. Therefore, real pin location is used for these large blocks by extracting the pin position from the layout.

C. Synthesis and Place–and–Route

Each functional block is synthesized individually, and the netlist of the whole VU is built up by combining the synthesized blocks. To reduce complexity of netlist processing by tools, the flattened netlist of the whole VU is employed.

In our design flow, repeater buffers to improve slew late are automatically inserted into inter–region nets by Repertory[6] after the first Place–and–Route(P&R). Repertory inserts the buffers with the wire shape and load information taken into consideration.

D. Static timing analysis

The static timing analysis is applied to the whole VU after the synthesis and also after in place optimization (IPO) process.

During the timing analysis after synthesis(pre–layout), the estimated wire shape and load (see Section4.2) is annotated. Moreover, repeater insertion is executed according to the estimated wire load for the timing analysis in order to increase the accuracy of analysis.

In the analysis after IPO(post–layout), the wire load is back–annotated from the final P&R result.

In our design, custom blocks are laid out as structured elements through the design flow. The timing information of custom blocks is given in the form of a gray model which has all latch–to–latch timing arcs.

V. Evaluation of design flow

The distribution of timing error paths in VU is shown in Fig.8. The x-axis indicates slack values and the y-axis the number of the error paths.

The data are obtained from the four kinds of path analysis explained below.

1. *Pre–layout path analysis with the wire load model*
   The wire load is calculated from the wire load model

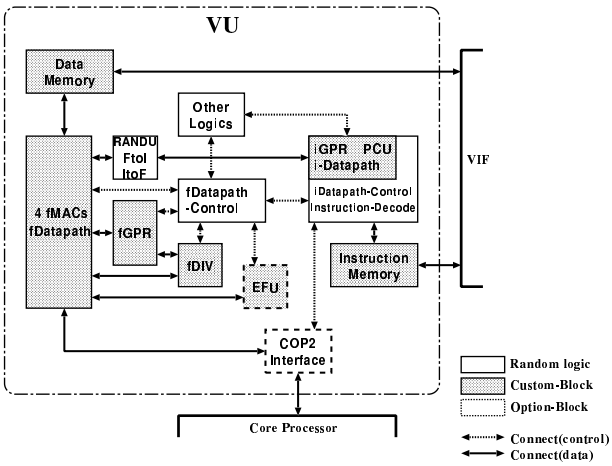

Fig. 6. Interconnects of VU function blocks

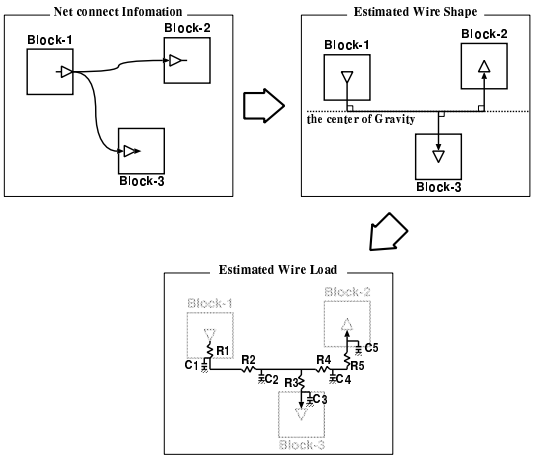

Fig. 7. Wire shape estimation for static timing analysis

2. *Pre–layout path analysis with capacitance annotation only*
   The wire load inside a region is calculated from the wire load model, while a inter-region area, the wire load is annotated only capacitance form.

3. *Pre–layout annotated R,C and repeater*
   The wire load in a region is annotated from wire load model. The wire load is annotated in the forms of both capacitance and resistance outside of the region. In addition, repeaters are inserted for the inter-region wires.

4. *Post-layout*
   The wire load is annotated entirely from the final P&R result.

As Fig.8 shows, the three pre-layout methods tend to have the same characteristic of path distribution against the slack values. However, in the region where the slack is worse than -0.1ns, analysis with R/C and repeaters exactly matches with that of post-layout. This extracts the major critical paths without proceeding to post-layout design flow.

In the region where the slack is better than -0.1ns on the other hand, the difference against the post-layout plots becomes bigger as the other two of the pre-layout results do. This means that the error gets bigger even with R/C, as the slack of the paths gets smaller. However, this implies that most of the violation paths have already been almost at the target speed. In this sense, the pre-layout analysis with R/C is effective in that , it is almost as accurate as post-layout for major timing violations, especially in the early phase of design.

The improvement of the timing error is shown in Fig.9.

Firstly, the number of the slack paths in the post-layout analysis is greater than that of the pre-layout one(Fig.9 (A)). The difference is caused by an inaccurate estimation of wire load for the -layout path analysis resulting from using only wire capacitance in the timing analysis. By applying the wire-shape estimation and repeater insertion in the next iteration, it can be reduced to (B) in the figure.

Next, the worst slack becomes worse in the post-layout path analysis compared with that of the pre-layout analysis(Fig.9 (C)). It is caused by an not-optimized region size and inner-region critical paths. After repartitioning regions to 4K–10K gates, the error is reduced to (D).

The time required for each design flow stage is shown in Table I. Pre–layout timing analysis consumes only one sixth time of post–layout timing analysis. This design flow has been iterated over 20 times until the operation frequency of 300MHz is achieved, and during a half of the iterations, only the pre–layout timing analysis has been tried and the further stages including P&R have been skipped. Thus, the total time required is reduced down to half by our design flow.

VI. Conclusions

Our new design method successfully reduce the wire load by partitioning functional blocks into appropriate sizes with

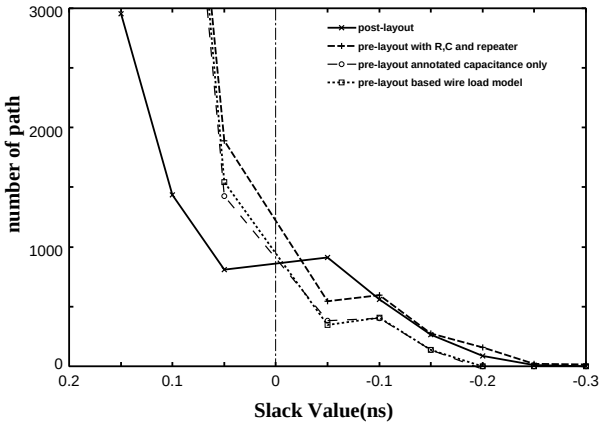

Fig. 8. the distribution of timing error paths in VU

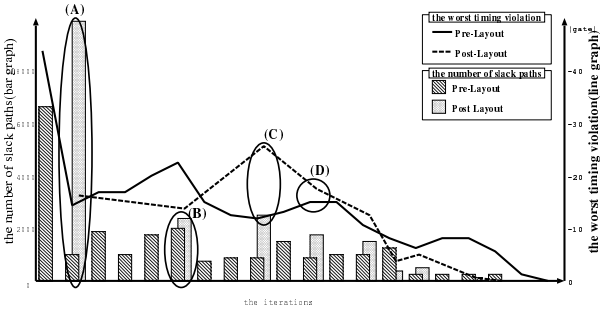

Fig. 9. the improvement of timing error paths in VU

TABLE I
THE TIME REQUIRED FOR EACH DESIGN FLOW STAGE

stage in design flow	pre–layout	post–layout
wire load estimation for synthesis	0.2 hours	0.2 hours
synthesis and scan path generation	5 hours	5 hours
estimation of wire load for timing analysis	0.05 hours	—
repeater insertion for timing analysis	0.15 hours	—-
pre–layout timing analysis	10 hours	—
the 1st Place–and–Route	—	10 hours
repeater insertion and Detail–Route	—	2 hours
In–Place Optimization	—	54 hours
the final Place–and–Route	—	4 hours
post–layout analysis	—	10 hours
Total	15.4 hours	85.2 hours

VU architecture taken into consideration. RTL hierarchy is matched exactly with the physical regions of the floor plan and it makes feasible to categorize the wire load into inter–region group and intra–region one. The pre-layout timing analysis gives highly accurate feedback to our timing design by estimating wire shape and annotating both resistance and capacitance.

The operating frequency of 300MHz has been achieved as a consequence of our improved design method.

VII. Acknowledgment

We thank Kouji Hashizume, Harutaka Goto, Shigeyuki Hayakawa, Yukihiro Urakawa, Junji Mori, and the members of System ULSI Lab. for their technical advice. We thank Masato Nagamatsu and Masashi Sasahara for their useful comments to improve our presentation.

References

[1] K.Kutaragi, et al., "A Microprocessor with a 128b CPU, 10 Floating-Point MACs, 4 Floating-Point Dividers, and an MPEG2 Decoder," *IEEE ISSCC Digest of Technical Papers*, Vol.42, pp.256-257, 1999.

[2] F.Michael Raam, et al., "A High Bandwidth Superscalar Microprocessor for Multimedia Applications," *IEEE ISSCC Digest of Technical Papers*, Vol.42, pp.258-259, 1999.

[3] T.Hattori, et al., "Design Methodology of a 200MHz superscalar microprocessor SH–4," *the 35th Design Automation Conference*, pp.246-249, 1998.

[4] A.Srinivasan and K.Chaudhary, "RITUAL:A Performance Driven Placement Algorithm for small ICs," *ICCAD-91*, pp.48-51, 1991.

[5] C.Ramachandran, F.J.Kurdahi, D.D.Gajski, A.C.-H.Wu, and V.Chaiyukul, "Accurate Layout Area and Delay Modeling for System Level Design," *ICCAD-92*, pp.355-361, 1992.

[6] N.Kojima, Y.Parameswar, C.Klingner, and Y.Ohtagro, "Repeater Insertion Method and its application to the 300MHz 128-bit 2-way Superscalar Microprocessor," *ASP-DAC2000*, in press.