Chapter 6

Parallel Multiplier Implementation

Ten different architectures for designing parallel multipliers are discussed in previous chapters. The delay analysis for different operand sizes of parallel multipliers has also been done on the basis of their topology using normalized gate delay. In the present chapter VHDL coding of parallel multipliers and their synthesis is discussed. The delay, area, power and cell counts obtained from synthesis are presented. A detailed analysis of all the multiplier synthesis results is done.

6.1 VHDL coding of multipliers

Multipliers of operand sizes 8, 16, 32, 54 and 64 bits are designed using all the ten architectures discussed in the previous chapter at gate level using VHDL coding. Each multiplier design is divided into three components. The first component is partial product generator, the second component is partial product accumulator and the third component is the final adder. These three components are instantiated in the main multiplier architecture.

For partial product generation, a uniform code with slight modifications as per the requirements is used for all architectures except the two using radix-64 encoding. This code uses radix-4 Booth encoding. This code is used to generate partial products for different operand sizes of multipliers, by changing the operand size through a generic statement. For radix-64 architecture, a different code is developed.

Coding for partial product accumulation was the most challenging. For coding of all architectures, delay minimization and regularity in code is considered as the main objective.

As the accumulation stages are complex, effort is made to have a common code for a specific architecture, so that it can be used for all operand sizes with only slight modifications as per the requirements.

In the final adder or RB to NB conversion stage CLA or CLEBC adders are used. Their performances are already discussed in chapter 4. For coding of higher length adders, bottom up approach is followed. Initially an 8-bit CLA is coded. This 8-bit CLA is instantiated in a 16-bit CLA. The 16-bit CLA is used as a component in the 32-bit CLA. The 32-bit CLA is again used as a component in the 64-bit CLA and the 64-bit CLA is used as a component in 128-bit CLA. Same methodology is also used for coding of all the CLEBCs.

Thus for all the ten multiplier architectures discussed in chapter 5, multiplier codes for bit sizes 8x8, 16x16, 32x32, 54x54 and 64x64 are written. The functionality of all these 50 multipliers is verified with sufficient random input patterns using ModelSim simulator.

6.2 Synthesis of multipliers

Given the functionally correct multiplier codes, the implementation objective is to synthesize multipliers at ASIC level using Magma EDA tool. But before that, the codes are tested for all dangling wires, unused variables, signals and all other warnings using FPGA Advantage 5.2. Codes are modified to remove such warnings. Such warnings can also be obtained using Magma tool but Magma tool cannot indicate the locations of the causes of warnings in the code. So we have used FPGA Advantage 5.2 for this purpose.

The synthesis-ready code is used by Magma tool for synthesis. Magma is an EDA tool suitable for highly complex integrated circuits while maximizing quality of results with respect to area, timing and power, and at the same time reducing overall design cycle time and cost. Magma provides a complete RTL-to-GDSII design flow that includes prototyping,

synthesis, place & route, and signal and power integrity chip design capabilities in a single executable, offering “The Fastest Path from RTL to Silicon”.

In the first step of synthesis, a standard cell library is imported, which is used for mapping to the gates present in the code. Then the VHDL codes for all the units of a multiplier are imported. Among the imported ones, the main entity is elaborated to check for errors in the code. In this stage, the code can be corrected for any errors or warnings. Once the code is elaborated successfully, a dataflow graph of the architecture is created, which can be viewed in schematic editor. Then a fix RTL command is used to map the data flow graph to technology independent generic primitive cells of Magma. In case of any unbounded cell in the RTL design to the library cell, they are bounded using run bind logical command. As the multiplier design is made for delay minimization, the design is constrained by time. The multiplier cells are added with timing information of the super cell models of library. A super cell model is a model abstraction. It contains a number of models of different sizes for a single entity. It has fixed delay but variable size, and it scales linearly with the load it drives. These super cells are later mapped back to actual standard cells with driving strength suitable for load in the physical optimization stage using fix cell command. A fully synthesized multiplier of operand size 8 implemented using WM42CLA architecture is shown in Appendix B. Once a model is synthesized, using check model command checks the correctness of the synthesized model.

For all multiplier implementations only four metal layers are used. In a four-layer design layers 1 and 3 are for horizontal routing layer and 2, 4 are for vertical routing. Out of these four layers, usually layer 1 is used by standard cell geometry and is unusable for routing. Metal layer 2 is 20% obstructed by vias connecting layer 1 and layer 2. Layer 3 is

horizontal and fully available and layer 4 is vertical and fully available. For this reason there is 80% more vertical routing resource than horizontal resource. So the ratio of horizontals to vertical routing resource is 1/1.8 = 0.56. Because of these reasons we have chosen the aspect ratio of the core of all the multipliers as 0.56. The standard cell library available and used is tcb013 (130-nm silicon process technology library of Toshiba). The process technology details are summarized in Table 6.1.

Table 6.1 Process technology used in the synthesis of multipliers

Technology	0.013µm CMOS, 4-metal layer
Supply voltage	1.08V
METAL1 Routing/Width/pitch/spacing	Horizontal/0.15u/0.4u/0.17u
METAL2 Routing/Width/pitch/spacing	Vertical/0.19u/0.45u/0.2u
METAL3 Routing/Width/pitch/spacing	Horizontal/0.19u/0.4u/0.2u
METAL4 Routing/Width/pitch/spacing	Vertical/0.19u/0.45u/0.2u

6.3 Hardware and operating system used

The synthesis of complex and large multipliers using Magma requires huge computational resources. For this SUN Solaris 8 operating system with 1.28 GHz Ultra SUN SPARK IIIi computers available in OLAB are used. The size of RAM is 8GB. With this computation facility, a typical 64x64 WM32CLA multiplier takes 8 hours and 23 min for completing synthesis.

6.4 Criteria in Evaluating Multipliers

The most important criterion taken for comparison is delay. The other quantities i.e. area, power and cell count are also taken into consideration for comparison.

Delay

It is the worst-case delay for entire multiplication of a synthesized multiplier. This is obtained by using the command “report timing path”.

Power

It is the combination of *leakage*, *swcap* and *internal* power. *Leakage* power is the power dissipation through leakage current, when the signal is stable. *swcap* is the power dissipated to charge and discharge any capacitance of input load and wire. The *internal* power dissipation is because of the short circuit current that flows through the PMOS and NMOS parts, when both are conducting. This is obtained by using the command “report power analysis”.

Area

This is the core area of synthesized multiplier. This is obtained by using the command “report model”.

Cell count

This gives the total number of cells used in multiplier architecture. The cell count is obtained by using “report model” command.

6.5 Simulation results

The coding and synthesis of multipliers with operand sizes of 8, 16, 32, 54 and 64 bits using different architectures has been discussed in the previous sections. The performance of synthesized multipliers is measured in terms of the delay, area, power, and cell count and is listed in tables A.1 to A.10 in appendix-A.

6.6 Comparison of architectures

The theoretically calculated delays for different multiplier architectures and for different operand sizes are discussed in chapter 4. All the multipliers are also implemented and the delay, power and area for each multiplier are summarized after synthesis in Tables A1 to A10 in appendix A.

The theoretical delays and the post synthesis delays, areas and leakage powers for the 8x8 multiplier using different architectures are summarized in Table 6.2. The theoretical architectural delay in terms of T_{XOR} for 8x8 multiplier is maximum using WMRBCLA and is 20.84 T_{XOR}. This is considered as 1 and all other delays are normalized with respect to this and are noted. Similarly the synthesized multiplier delay (T), area (A), power (P), TAP (product of delay, area and power) and AT^2 (product of area and square of delay) are also normalized and noted in Table. 6.2. Now all these normalized values are plotted in Fig 6.1 for performance comparisons of different architectures. From the figure the theoretical and post route delay estimates indicate that the WM42CLEBC is the fastest architecture for an 8x8 multiplier. For all the other figures of merit, the WM42CLA is the best as it has the smallest A, P, TAP and AT^2.

Using the same method the normalized performance in terms of different figures of merit of all the architectures are plotted for operand sizes of 16, 32, 54 and 64 bits in Fig. 6.2 to Fig. 6.5. The plot for 16x16 multiplier shows that the WM42CLEBC has the second best theoretical delay and smallest post route delay. The AMCLA architecture has the smallest area. In terms of power, ATP and AT^2 the WM42CLEBC architecture outperforms the others.

The next plot corresponding to 32x32 multiplier in Fig. 6.3 shows that the WM42CLEBC is practically as well as theoretically the fastest. The AMCLA architecture is

most compact and AMCLEBC has the smallest value of P and TAP. Finally AT^2 is smallest for the WM42CLEBC.

The comparison plot corresponding to 54x54 multiplier in Fig. 6.4 shows that, the WM32CLA has the smallest post route delay. For area and power the AMCLA out performs others. WM32CLA has the smallest AT^2. In terms of ATP also it is very close to the smallest value.

The performance plot of 64-bit multipliers is shown in Fig. 6.5. It shows that the WM42CLEBC is the best for delay, TAP and AT^2. For area and power AMCLA is the best.

Table 6.2 Summary of performance of a 8x8 multiplier using different architectures

Multiplier architectures / Performance measuring parameters	AMCLA	AMCLEBC	WM32CLA	WM32CLEBC	WM42CLA	WM42CLEBC	WMRBCLA	WMRBCLEBC	Radix64CLA	Radix64CLEBC
Theoretical dealy in terms of Txor	17.68	11.98	18.68	12.98	18.68	12.98	20.84	15.14	20.58	14.88
Normalized theoretical dealy	0.848369	0.574856	0.896353	0.622841	0.896353	0.622841	1	0.726488	0.987524	0.7140115
Post route delay in psec (T)	2659	2570	2835	2656	2593	2312	3376	3367	3446	3558
Normalized post route delay(T)	0.74733	0.722316	0.796796	0.746487	0.72878	0.649803	0.948848	0.946318	0.968522	1
Area in mm^2 (A)	0.015	0.018	0.009	0.009	0.008	0.012	0.014	0.064	0.040	0.036
Normalized A	0.234375	0.28125	0.140625	0.140625	0.125	0.1875	0.21875	1	0.625	0.5625
Cell count	770	833	558	538	580	686	771	931	1541	1204
Power in micro watt (P)	149.4	209.9	105	123.5	112.4	150.1	174.2	221.2	340.3	263.1
Normalized P	0.439024	0.616809	0.308551	0.362915	0.330297	0.441081	0.511901	0.650015	1	0.7731413
ATP	5958.819	9709.974	2679.075	2952.144	2331.626	4164.374	8233.389	47665.95	46906.95	33699.95
Normalized ATP	0.125015	0.203713	0.056206	0.061935	0.048917	0.087368	0.172734	1.00002	0.984096	0.707017
AT^2	106054.2	118888.2	72335.03	63489.02	53789.19	64144.13	159563.3	725548.1	474996.6	455737.1
Normalized AT^2	0.146171	0.16386	0.099697	0.087505	0.074136	0.088408	0.219921	1	0.654673	0.628128

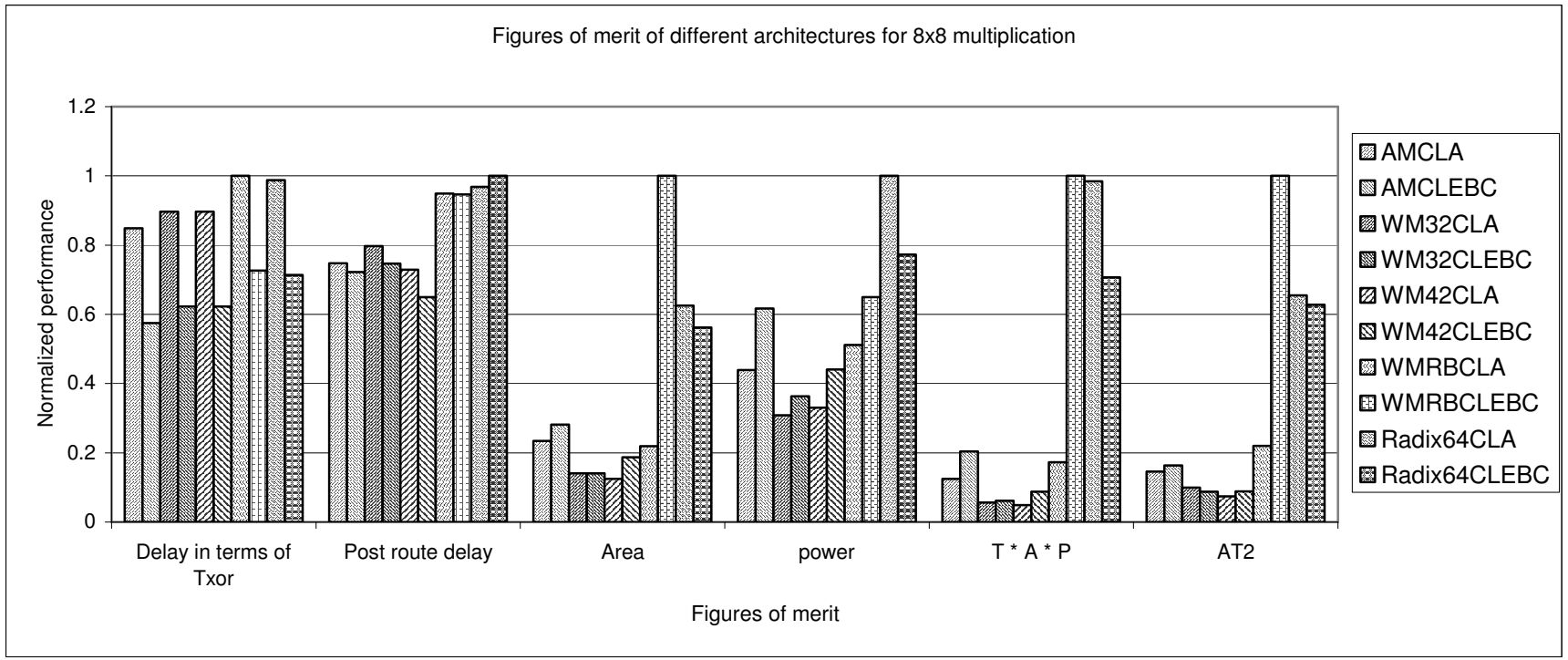

Fig. 6.1 Figures of merit of different architectures for 8x8 multiplication

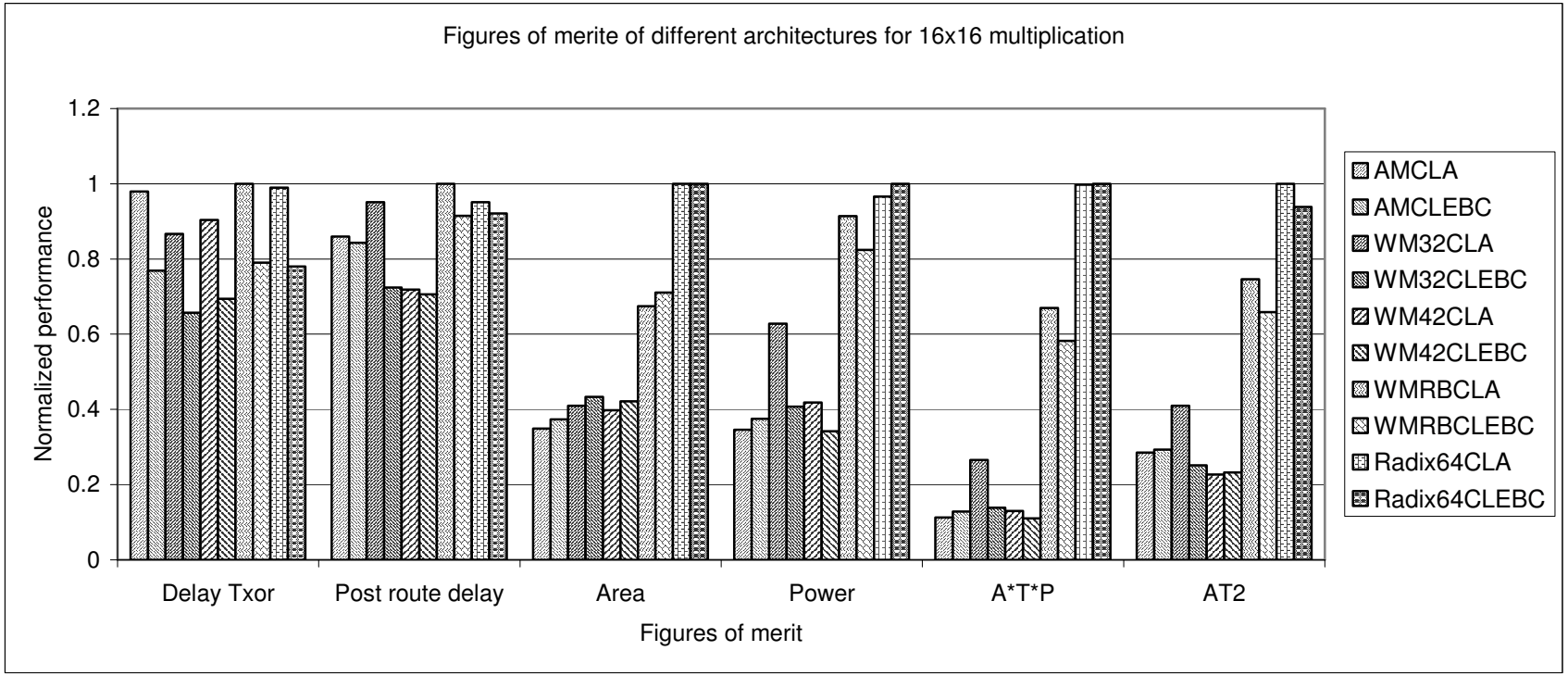

Fig. 6.2 Figures of merit of different architectures for 16x16 multiplication.

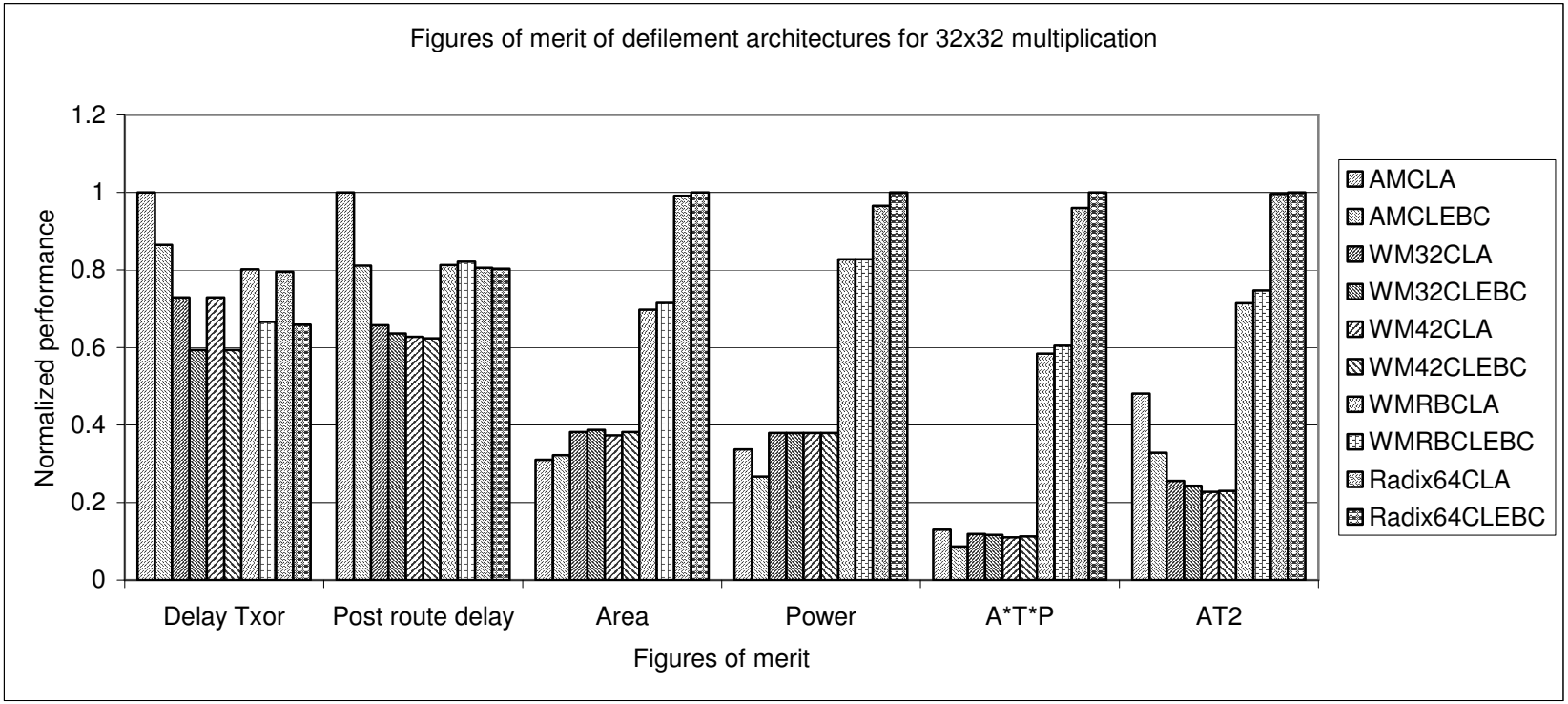

Fig. 6.3 Figures of merit of different architectures for 32x32 multiplication.

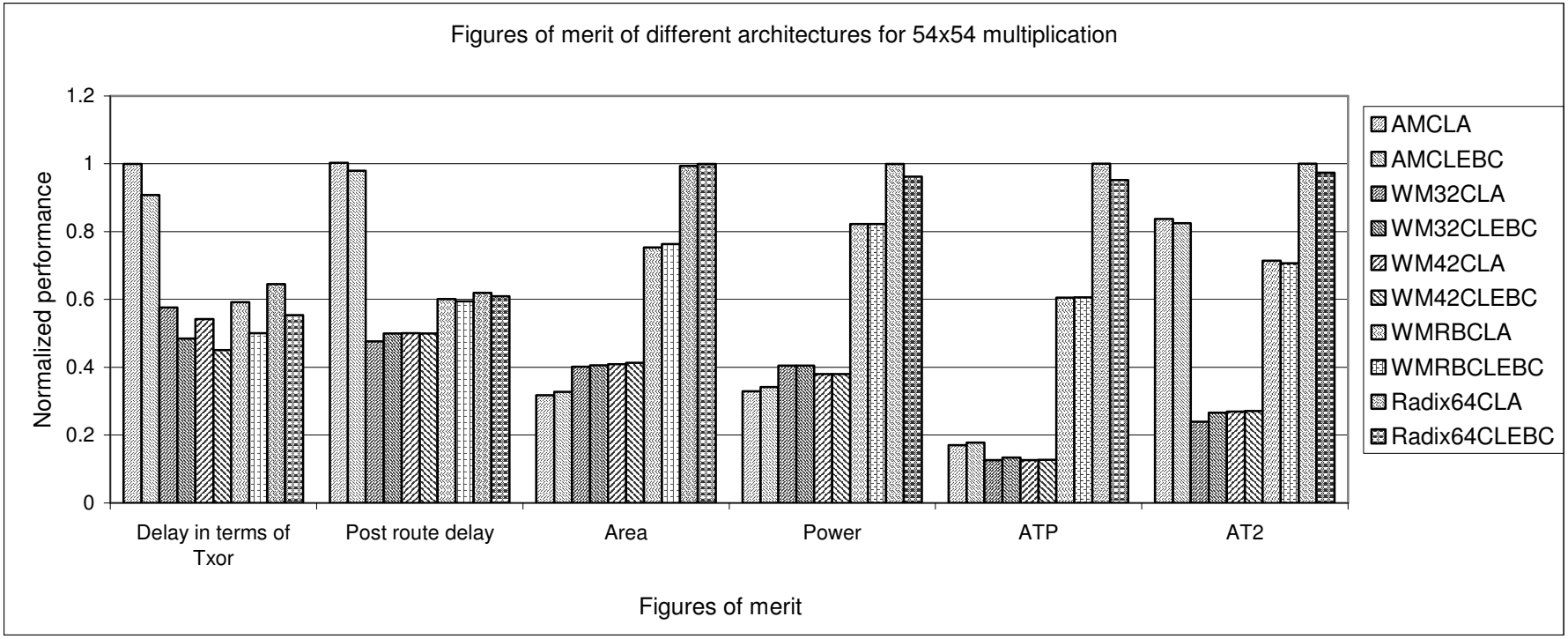

Fig. 6.4 Figures of merit of different architectures for 54x54 multiplication.

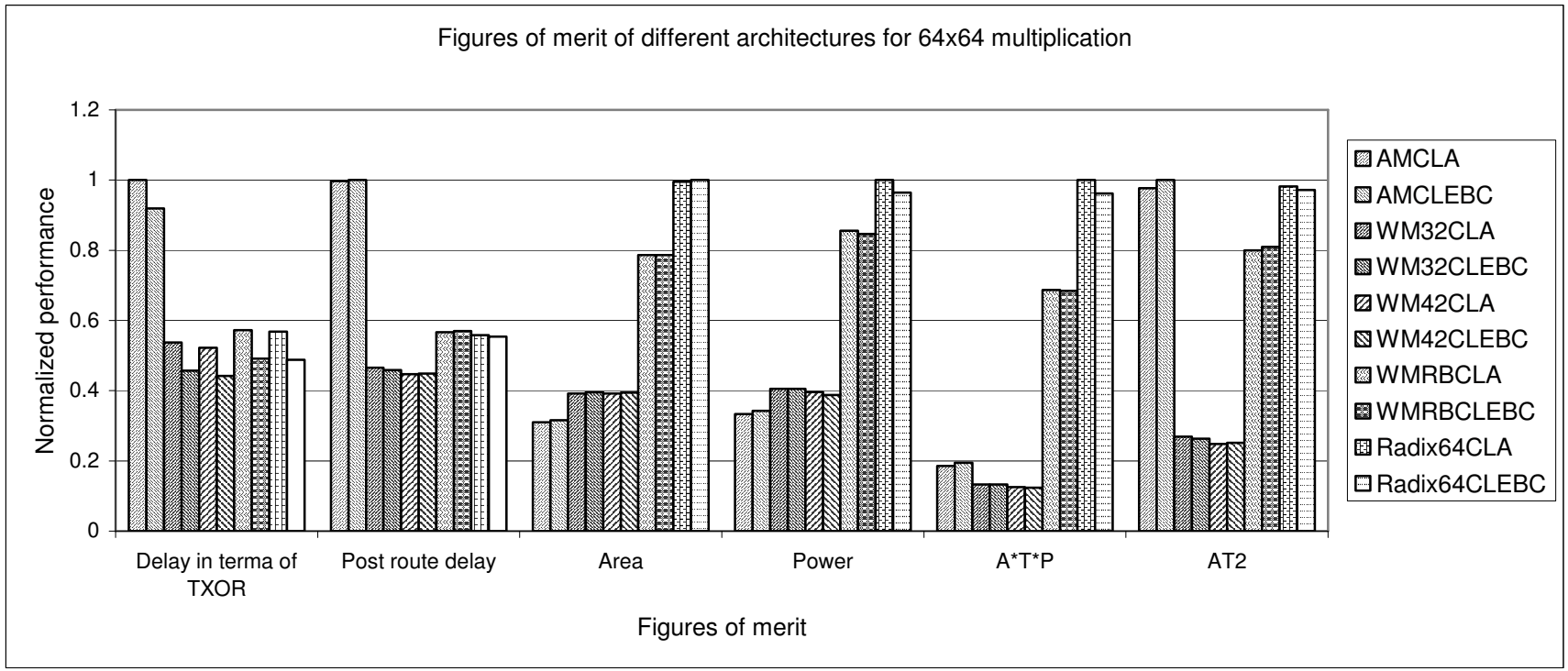

Fig. 6.5 Figures of merit of different architectures for 64x64 multiplication.