Content Addressable Memory (CAM) Implementation and Power Analysis on FPGA

by

Teng Hu

B.Eng., Southwest Jiaotong University, 2008

A Report Submitted in Partial Fulfillment

of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

University of Victoria

Supervisory Committee

Content Addressable Memory (CAM) Implementation and Power analysis on FPGA

by

Teng Hu

B.Eng., Southwest Jiaotong University, 2008

Supervisory Committee

Dr. Chris Papadopoulos, (Department of Electrical and Computer Engineering)

Co-Supervisor

Dr. Mihai Sima, (Department of Electrical and Computer Engineering)

Co-Supervisor

Supervisory Committee

Dr. Chris Papadopoulos, (Department of Electrical and Computer Engineering)

Co-Supervisor

Dr. Mihai Sima, (Department of Electrical and Computer Engineering)

Co-Supervisor

Abstract

Content Addressable Memory (CAM) has been widely used in network devices for fast lookup functions. CAM implementation based on Field Programmable Gate Array (FPGA) has become a popular solution due to its flexibility. With the increasing capacity of CAM, reducing power consumption has been the main challenge for implementation on FPGA. This report investigates and implements two low-power schemes, pipelining and precomputation for RAM-based CAM. The pipelining scheme divides RAM into several segments as a pipeline. Mismatched RAM blocks disable the subsequent search operation in the following segments and therefore power consumption is reduced. For precomputation scheme, extra information is extracted from CAM words and input keys before the search operation. It saves power by filtering out mismatched RAM blocks in the precomputation stage.

In this work, a complete power analysis of RAM-based CAM using Xilinx Vivado has been performed. The comparison of the power consumption between the conventional scheme and low-power schemes is illustrated. Under the same test case, the average dynamic power consumption of CAM with the pipelining scheme can be reduced by 86% compared to the conventional scheme. The precomputation scheme based on the pipelining scheme further optimizes the power consumption of CAM. It decreases the final result of power estimation by 36% compared to the pipelining scheme.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would first like to thank Dr. Chris Papadopoulos and Dr. Mihai Sima for their guidance and support throughout my graduate study. As my teachers and supervisors, they are always nice and patient with me. I truly appreciate all they have done throughout my study at the University of Victoria.

I would like to thank my parents for their love and encouragement. They are always proud of me. Without their endless spiritual support, I can't go further and complete my graduate study.

I would also like to thank my lab members, Po Zhang and Mi Tian for their assistance and support in the lab.

I am grateful to Mr. Lam and Mr. Ralph at Intel who gave me an offer several months before my graduation. I appreciate their patience.

I must thank all my friends who give me help and share everything with me.

Chapter 1

Introduction

Content-addressable memory (CAM) provides high-speed search functions. CAM implemented on hardware circuits is much faster than approaches based on software algorithms and it is widely used in search-intensive applications such as address lookup in network routers. The CAM implementation based on Field Programmable Gate Array (FPGA) has been a solution for applications that do not require very large memory capacity or very high performance, and there has been a growing concern about its power consumption as the memory capacity increases. This report will focus on power optimization to CAM based on FPGA.

1.1 Content-addressable memory (CAM)

A content-addressable memory (CAM) searches an entire table of stored data inside for the input key, and returns the address of the matched data [1]. It is a special type of memory providing look-up table mapping. It differs from a traditional memory that returns the data stored at the given address. The comparison is shown in Figure 1.1; it is apparent that the function provided by CAM is the opposite of the traditional memory like Random Access Memory (RAM). For a read operation in RAM, an address location

is given as an input and the target RAM cell is accessed by the control logic. The output is the content of that address. Unlike having a certain address, a search process is to be performed in CAM because the target data location is unknown. Through matching between input data and stored data, the output shows where the target data is stored. CAM is typically used to find matched contents in a database behaving as a search engine.

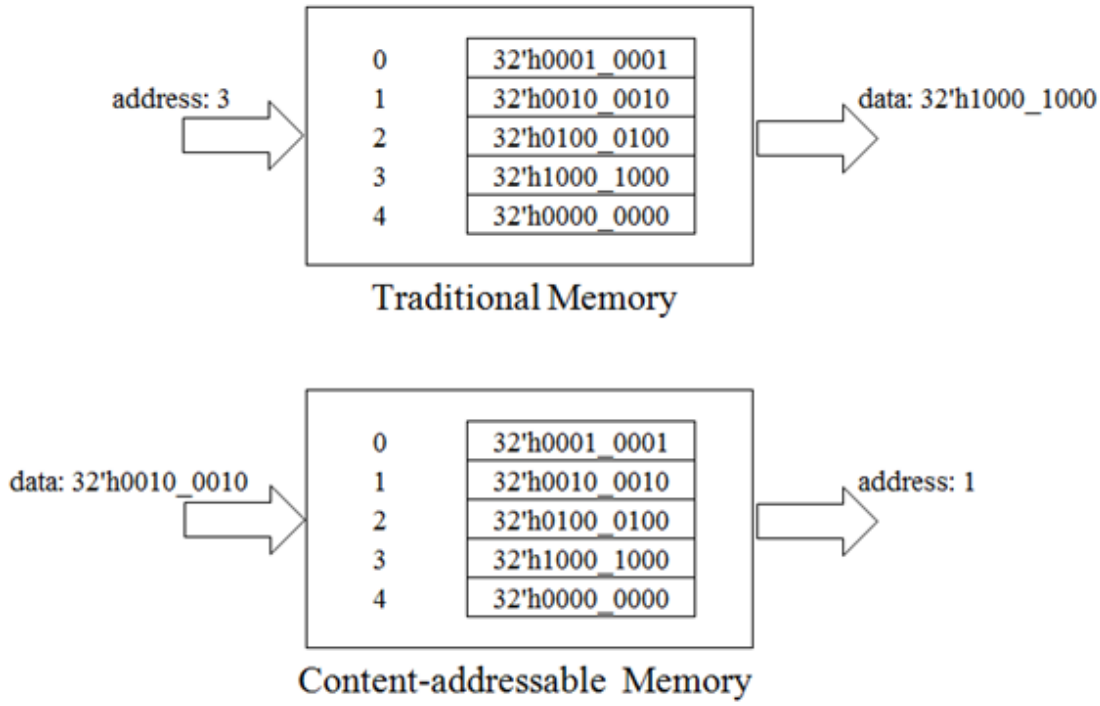


Figure 1.1 Comparison between CAM and Traditional Memory

Hence CAMs are especially popular in network routers since packet forwarding and packet classification requires fast lookup function [2]. In Figure 1.2, a routing table is

stored in a CAM inside the router. The destination address specified in the packet header is captured for routing table mapping. The router searches all reachable destinations listed and decides which port the packet will be forwarded to. It is critical that the search latency is reduced, otherwise the whole routing process will be delayed. CAM is also the first choice for applications requiring high-speed table look-up such as Hough transformation in digital signal processing [17].

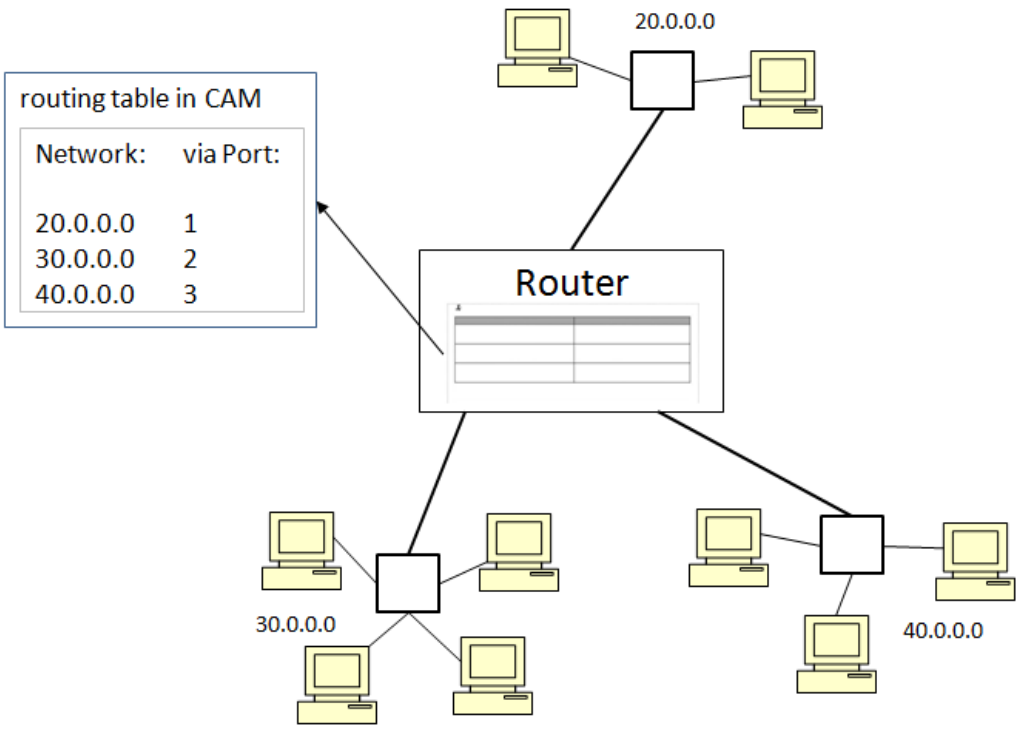


Figure 1.2 CAM application in router

1.2 Solutions for CAM

As the Internet has been widely used in daily life, the need for network devices is driven to increase. The exponentially growing amount of network traffic requires higher speed and better performance. Like the router shown in Figure 1.2, fast look-up table mapping is expected to provide solutions for the networking process including address lookups, intrusion detection, packet classification, address translation, etc [2].

The software-based solution provides flexibility and saves cost. The look-up table is written into assigned memory space in RAM or Flash. A network processor accesses to the memory and searches for target data with various algorithms such as linear search, binary search, and hashing search. However, the software solution is based on sequential access by instruction execution rather than direct hardware implementation, thus it is inherently slow.

CAM based on the custom circuit such as Application Specific Standard Product (ASSP) of CAM is a good choice for applications requiring high performance. This solution is able to implement fast table mapping in a single clock cycle by dedicated circuits for look-up function. Nevertheless, expensive ASSP brings impacts to system integration and raises power consumption and cost.

Modern FPGA chips have been developed rapidly, and provide a large number of diverse resources including memory, peripheral blocks, and embedded processor. This

flexible platform with reconfigurable logic units applies to various designs bringing benefits for system integration.

The abundant hardware resources in FPGA provide an option for CAM implementation. Optimized look-up table function based on programmable logic blocks working in parallel meets the requirement for fast table mapping. Furthermore, configurable Embedded Memory Blocks and look-up tables in FPGA improve resource utilization. The ASSP of CAM which is designed for general-purpose applications provides a large capacity and big data width usually but requires a large non-recurring engineering cost. It is a waste of power and cost for some applications requiring small size production batches. The FPGA design can be customized for specific applications making full use of hardware resources. In addition, there are broad FPGA applications in network devices. Thus CAM can be integrated into the system together with other function units such as networking processor for desired application.

1.3 Technical Challenges

Modern computer networks demand fast and high capacity CAMs. Meanwhile, the improvement of the two major design parameters leads to increased power consumption. The main topic of recent research in large capacity CAMs is to decrease power consumption without sacrificing performance [1].

Power consumption on FPGA has been a growing concern because of its notorious power hungry nature. FPGA consumes more power than Application Specific Integrated Circuits (ASICs) comparatively with the same logic design, because ASIC design provides full custom capability. Technology process, transistor architecture, wiring and layout, all design elements from bottom to top are available for full customization and optimization in terms of power, cost and performance. However FPGA is a prefabricated device containing programmable logic components. There are redundant resources in FPGA, as they are made for general purpose and re-usability. Place-and-route constraints lead to low logic utilization which brings extra power consumption on FPGA from unused logic, long signal paths and inefficient clock-trees [8].

CAM implementation on FPGA needs a large number of Logic Elements to hold memory values. The large memory array implemented in FPGA leads to complex interconnection and big path delay. In addition, the parallel access to memory and data matching leads extra power consumption. Furthermore, high-speed lookup function requires large logic circuits toggling at same cycle because of big data width. The huge amount of active resources result in high peak power consumption.

The power optimization for CAM based on FPGA is challenging because of above reasons. Two low-power schemes pipelining and precomputation are proposed to save power by reducing the scope of active resources in the search operation. The low-power schemes and the conventional scheme of CAM based on FPGA will be implemented and discussed in this report.

The thesis is organized as follows. In chapter 2, an overview of power analysis in FPGA is provided. Chapter 3 describes the implementation of the above three schemes. In chapter 4, a complete power analysis of CAM has been performed by comparing power consumption between different implementations. Chapter 5 gives conclusion and suggests future work.

Chapter 2

FPGA Power Analysis

FPGA becomes a popular choice for digital system implementation because it provides short time-to-market, flexibility, and low cost. Modern FPGA provides a larger capacity and abundant resources with leading edge process technologies for system integration. Currently, large System on Chip (SOC) designs on FPGA with millions of gates consume several watts of power [9].Thus power consumption is one of the most important concerns to design.

However, it takes much time and effort to get the actual power consumption through measurement after system integration and implementation. The designers must obtain reliable and accurate power estimation of FPGA at early design stage especially for low-power design. With power estimation result from desired tools, design tradeoffs are able to be taken into consideration at a high level of abstraction, so design effort and cost can be reduced [10].

2.1 Power analysis in Vivado

Vivado Design Suite is an EDA tool providing an integrated design environment for FPGA by Xilinx. It supports entire design flow from RTL design, synthesis to

implementation and simulation including power analysis. The Vivado power analysis feature performs power estimation through all stages of the flow: post-synthesis, post-placement, and post-routing [11].

2.1.1 Power estimation flow

The flow of power analysis is shown in Figure 2.1. After specification and RTL design completion, power estimation can be performed in Vivado. RTL design is mapped into actual resources such as logic units, RAM and IO in the target FPGA device through synthesis. However, the power estimation result after placement and routing is more accurate, since the exact logic and routing resources are available after implementation [11]. The back-annotated netlist after placement and routing contains all details of implementation including routing path and timing delay. The stage of Simulation in Figure 2.1 is optional. The power analysis without simulation stage is vectorless that doesn't require stimulus input. The assessment is based on average figures. The power analysis with simulation stage is vector based and the real data from simulation is used. Hence it provides the most accurate result.

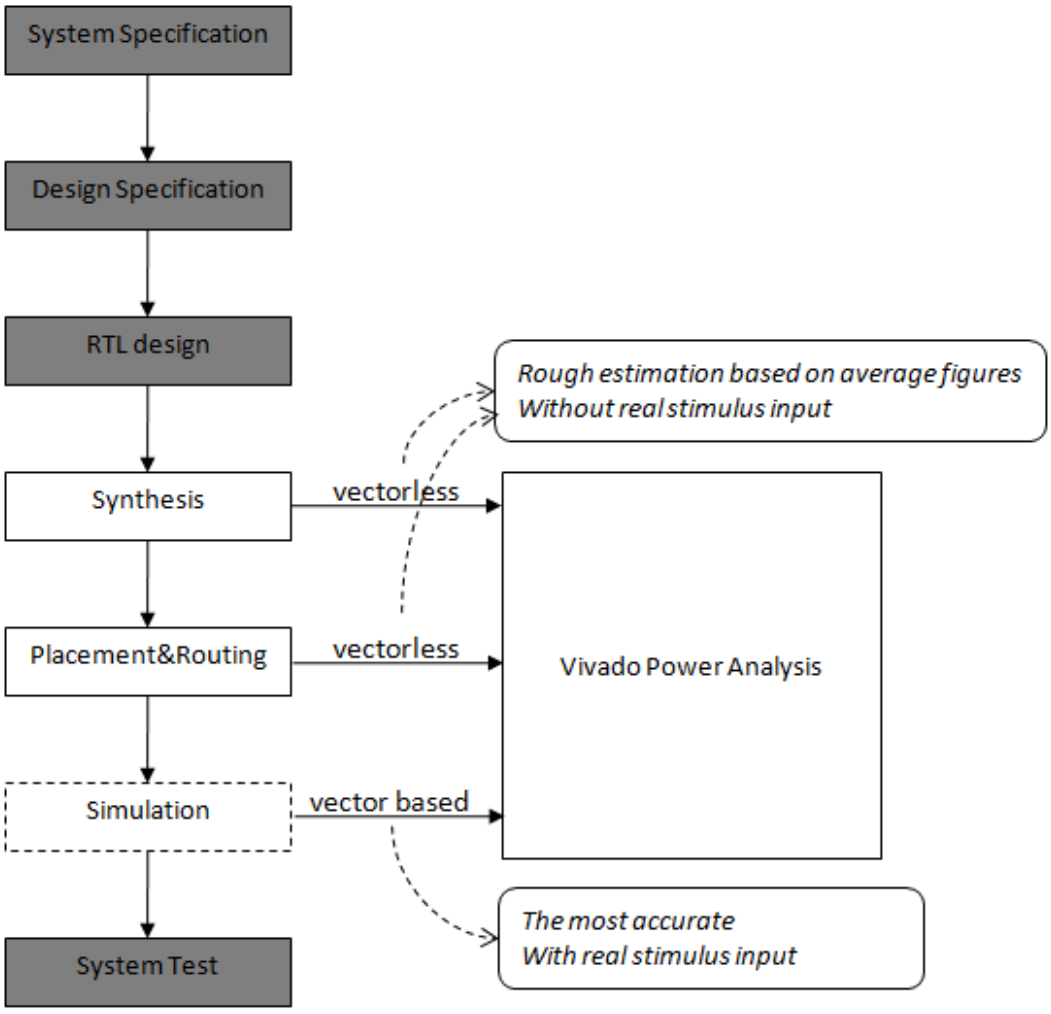


Figure 2.1 Power estimation flow

2.1.2 Vector based power estimation

The following equation is used to estimate the dynamic power of a gate or a connection line:

$$P = \alpha \cdot C_l \cdot V_{dd}^2 \cdot f \quad (1)$$

where α (referred to as switching activity) is the average number of 0→1 transitions in one clock cycle, C_l is the load capacitance, V_{dd} is the power supply voltage, and f is the clock frequency [3].

Equation (1) is fundamental power model for power estimation. Clock frequency and power supply voltage are predefined by system design specification. After placement and routing, the load capacitance of each node in netlist can be extracted from wire load model based on physical circuits.

Switching activity as a variable independent of design and physical device is obtained through simulation. It requires designers to perform simulations by giving test vectors so that target design behaves as expected. To get more accurate power estimation, a complete test scenario representing typical operation is needed to capture realistic design activity.

Switching Activity Interchange Format (SAIF) file is a type of file used for power analysis. It contains information of switching activity for each node in the netlist and can be generated by simulation in Vivado. From Figure 2.2, the duration of logic states and transition count for each node is described in SAIF file. For example, the information of switching activity for node "a1[0]" is described in the ninth line. The time duration at level 0 was 4680000 picoseconds and the transition of signal value occurred 431 times. Vivado applies switching activity for power analysis.

```
(VENDOR "Xilinx, Inc")
(PROGRAM_NAME "Vivado Simulator")
(VERSION "2015.3")
(DIVIDER /)
(TIMESCALE  1 ps)
(DURATION  10000000)
(INSTANCE  test
   (INSTANCE  uut
      (NET
         (a1\[0\] (T0 4680000) (T1 4315000) (TX 1005000) (TZ
0) (TB 0) (TC 431))
         (a1\[1\] (T0 4270000) (T1 4725000) (TX 1005000) (TZ
0) (TB 0) (TC 459))
         (a1\[2\] (T0 4530000) (T1 4465000) (TX 1005000) (TZ
0) (TB 0) (TC 456))
         (a1\[3\] (T0 4470000) (T1 4525000) (TX 1005000) (TZ
0) (TB 0) (TC 473))
         (a1\[4\] (T0 4735000) (T1 4260000) (TX 1005000) (TZ
0) (TB 0) (TC 474))
         (a1\[5\] (T0 4465000) (T1 4530000) (TX 1005000) (TZ
0) (TB 0) (TC 451))
         (a1\[6\] (T0 4545000) (T1 4450000) (TX 1005000) (TZ
0) (TB 0) (TC 462))
         (a1\[7\] (T0 4715000) (T1 4280000) (TX 1005000) (TZ
0) (TB 0) (TC 428))
         (a1\[8\] (T0 4445000) (T1 4550000) (TX 1005000) (TZ
0) (TB 0) (TC 445))
         (a1\[9\] (T0 4500000) (T1 4495000) (TX 1005000) (TZ
0) (TB 0) (TC 465))
```

Figure 2.2 SAIF file

At an early design stage, a complete test scenario may be unavailable. The switching activity parameters such as toggle rate and static probability can be set manually in vectorless power analysis for rough power estimation in Vivado.

2.1.3 Power estimation result

An example of power estimation result from Vivado is shown in Figure 2.3.

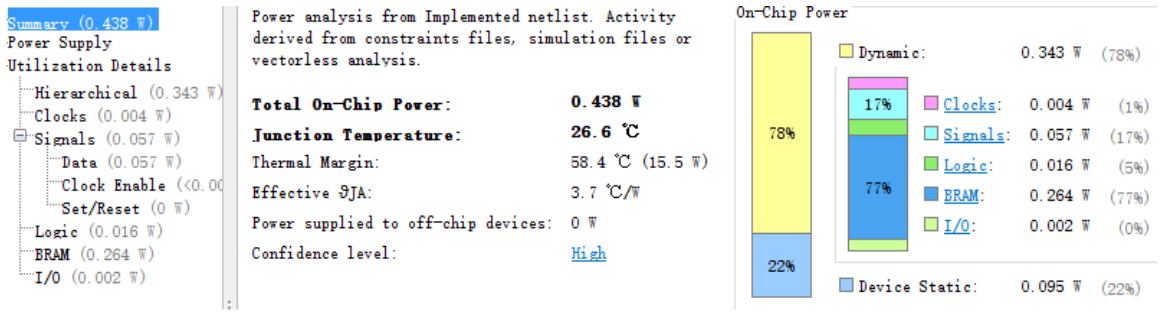


Figure 2.3 Example of power estimation result

"On-Chip Power" report is comprised of dynamic power and static power. Static power is the power consumption in the steady state resulting from intrinsic transistor leakage. It depends on device process, voltage, and temperature. Under same condition, static power is a constant value independent of logic design.

Dynamic power comes from internal switching activity of custom design which fluctuates in every clock cycle. If the voltage is a constant value, dynamic power is proportioned to the amount of active resource and clock frequency. Since static power is independent of logic design, the power optimization discussed later focuses on dynamic power only in this report.

Vivado provides power distribution on resource in the list of "Utilization Details" in Figure 2.4. The entry of "Hierarchical" breaks down power consumption through design

hierarchy. The power dissipation on each wire and logic unit is available under entries of “Signals” and “Logic”. Signal rate shown in Figure 2.4 denotes the number of transitions per second occur in the given time period.

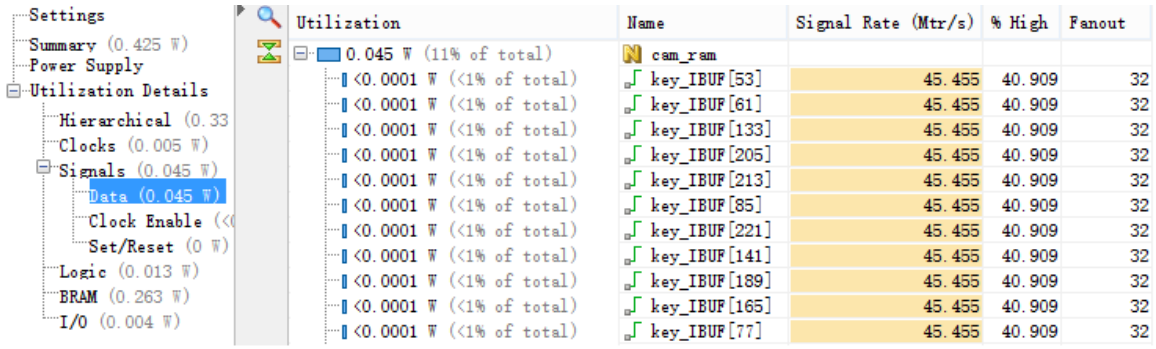
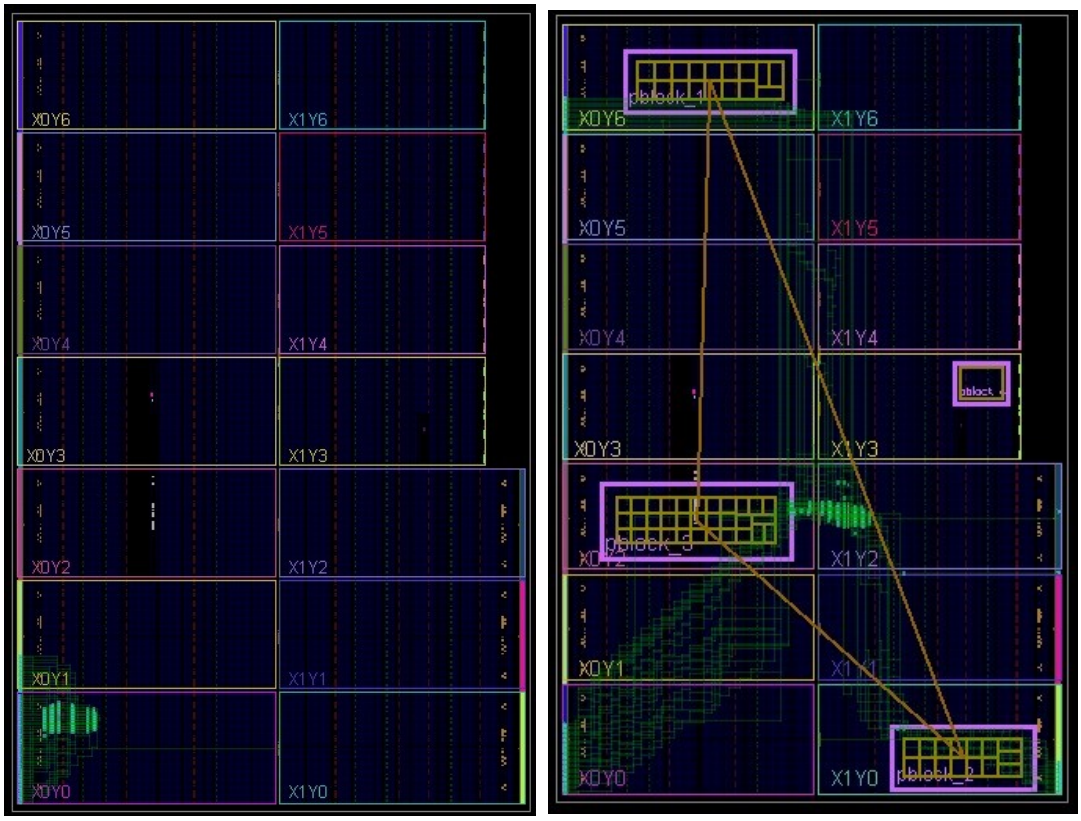


Figure 2.4 Power dissipation on resources

2.1.4 Power estimation for interconnection

Interconnection consumes most of the dynamic power of FPGA. In the literature [12], a thorough power analysis for Xilinx Virtex-II device showed that interconnection including clock network took up more than 70% of total dynamic power. The interconnection which takes up 90% of total FPGA area dominates in power consumption , because of the composition of the interconnect structures, including prefabricated wire segments of various lengths, with used and unused routing switches attached to each wire segment [10].The netlist after placement and routing provides physical information of interconnection like routing path. The example below shows the considerable impact of interconnection on power estimation in Vivado.

Figure 2.5 (a) is a layout of a multiplier after placement and routing. All LUTs and Registers are placed close to IO in the bottom left corner for better utilization and timing predictability. In this case, wire path is very short and its wire capacitance is comparatively small, thus the dynamic power is expected to fairly small.



(a) (b)

Figure 2.5 Layout of a multiplier. (a) Short wire routing. (b) Long wire routing.

The layout is changed with different constraints for placement and routing in Figure 2.5 (b). The IOs of the multiplier are assigned to different sides and registers are also

distributed to different areas. Hence wire routing goes across slices and wire length is significantly increased, so that the dynamic power is expected to be large. Table 2.1 shows power estimation of same wire with a different routing. The power consumption of long routing path is ten times larger than that of the short routing path. To further explore the impact of routing on power, some wires are gathered in Table 2.2 as samples for power analysis. Vivado doesn't provide direct information about the length of wires, but the list of all tiles which each wire goes across is available after placement and routing. The FPGA fabric is divided into a two-dimensional array of tiles, and they are conceptually laid out edge to edge [16]. A long wire goes across more tiles than a short wire. Hence the number of tiles can be used as a reference to the length of a wire. In Figure 2.6, the trend line shows that data almost matches the linear model. Samples of number 11 and 12 are normalized since their signal rate is half as other samples. Overall power estimation of interconnection with routing information brings the most accurate result in Vivado.

Table 2.1 Power estimation of wire p_1_in19_in

	Signal rate(Mtr/s)	% High	Fanout	Power (mW)
Short routing	100	50	15	0.07
Long routing	100	50	15	0.72

Table 2.2 samples of wires for power analysis of interconnection

Number	Power (mW)	Name	Signal Rate (Mtr/s)	Length (number of tiles)
1	0.081	a_IBUF[0]	100.00	125
2	0.470	a_reg_reg_n_0_[0]	100.00	447
3	0.079	b_IBUF[0]	100.00	107
4	0.215	b_reg_reg_n_0_[0]	100.00	208
5	0.164	p_0_in29_in	100.00	157
6	0.231	p_0_in199_in	100.00	212
7	0.665	p_1_in3_in	100.00	667
8	0.326	p_1_in7_in	100.00	284
9	0.535	p_1_in15_in	100.00	532
10	0.722	p_1_in19_in	100.00	751
11	0.097	r_OBUF[0]	50.00	240
12	0.075	r_OBUF[25]	50.00	215
13	0.024	row15[16].fadder/r_reg_reg[2]	100.00	25
14	0.014	row15[16].fadder/r_reg_reg[13]_2	100.00	9

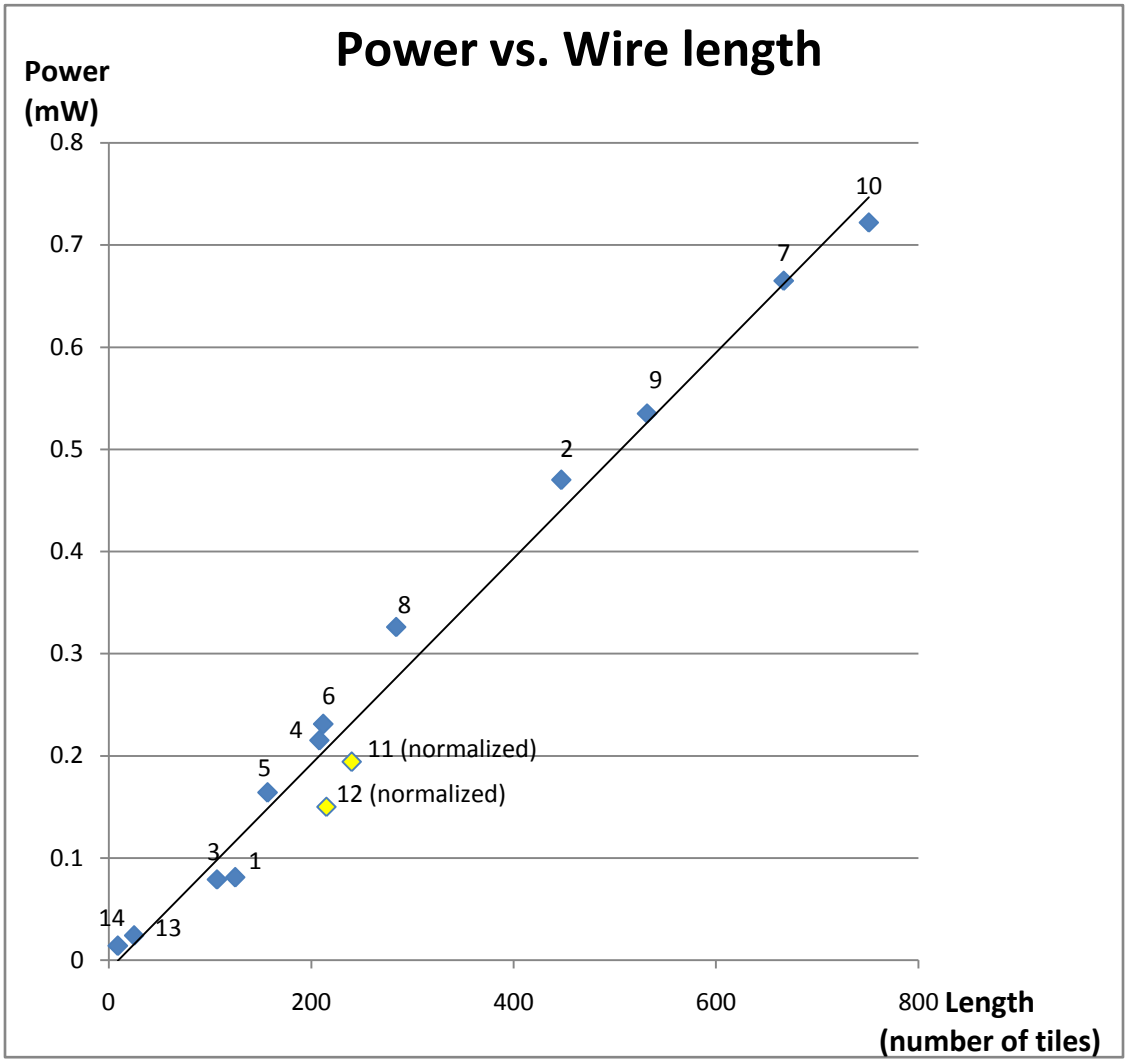


Figure 2.6 Power analysis on wire length

2.1.5 Instantaneous power estimation

Vector-based power analysis in Viavdo reports average power estimation in a user defined time period. However, instantaneous power estimation may be required for power optimization and power profiling. it can be obtained through vector-based power

analysis if the time period of switching activity capture is shortened to a single clock cycle.

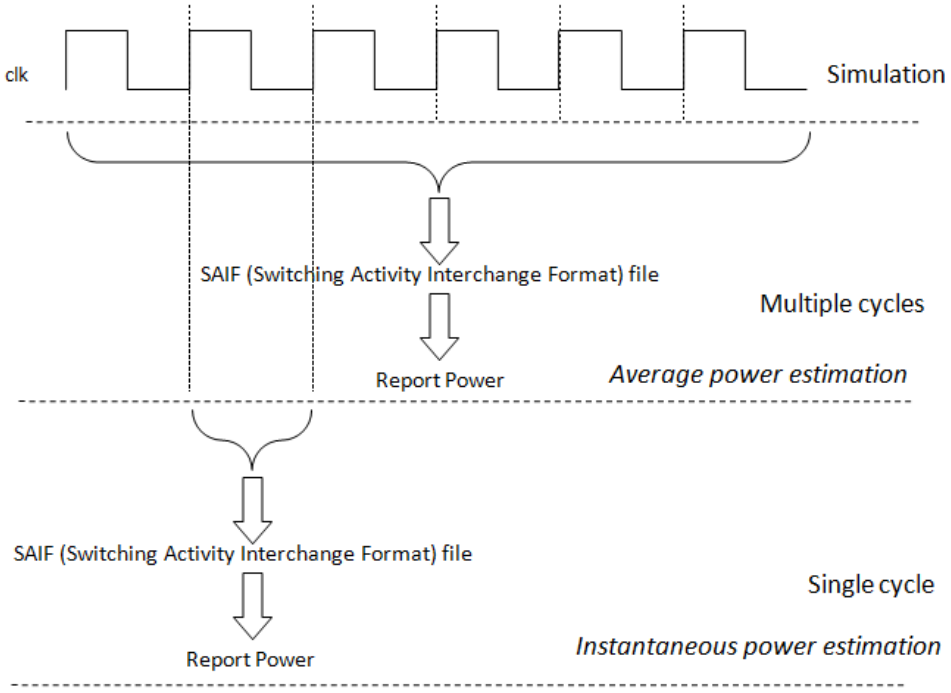


Figure 2.7 The approach to instantaneous power estimation

The approach to instantaneous power estimation is shown in Figure 2.7. The time period when SAIF file is generated can be specified in Vivado. The estimation result based on SAIF file from a single cycle is referred to as instantaneous power estimation. Figure 2.8 shows an example of power profile. The target design contains 64 32-bit counters and it is implemented and configured in Xilinx's Kintex-7 XC7K325T for power estimation.

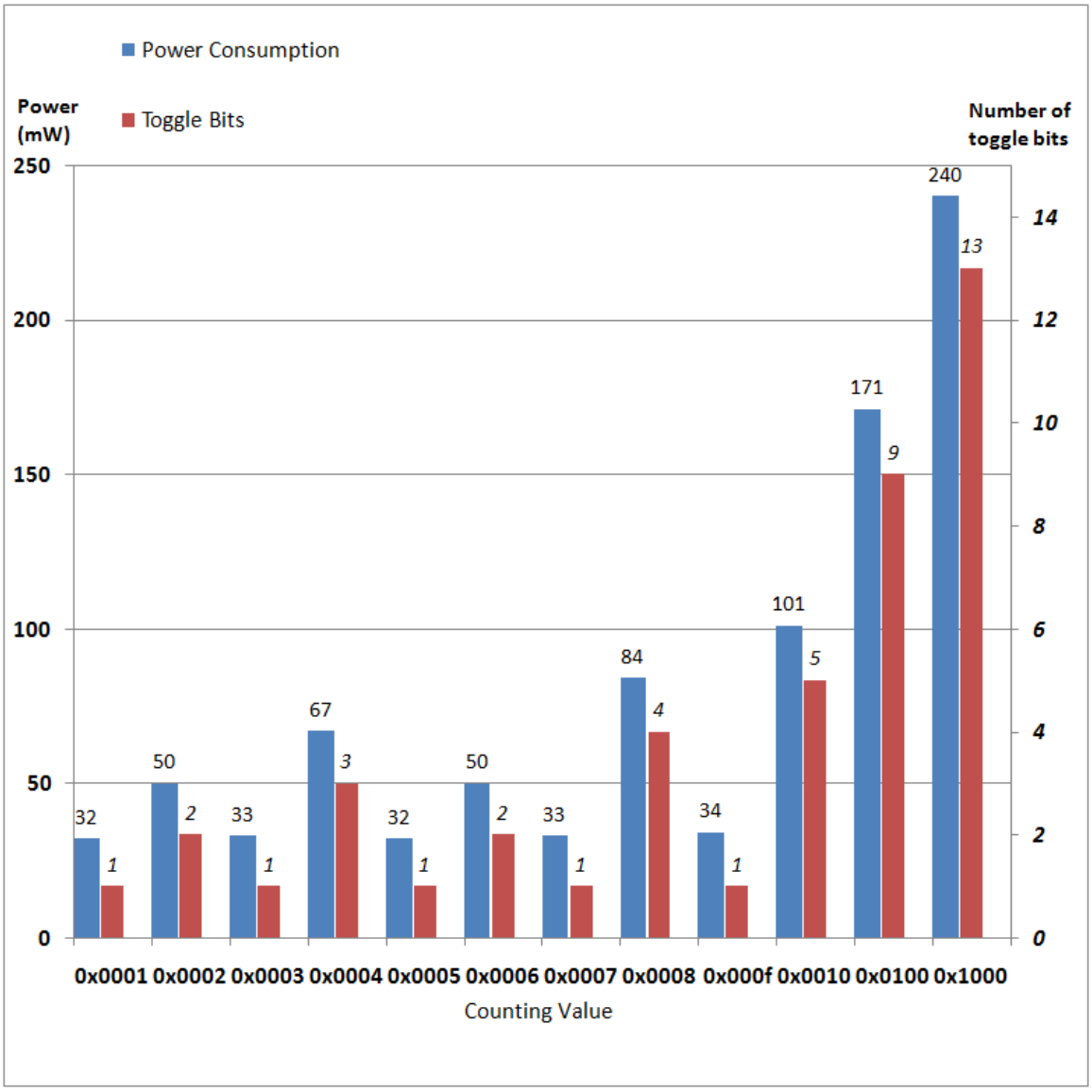


Figure 2.8 Instantaneous power analysis- 32-bit counter

The 64 counters keep counting synchronously and their counting value updates every cycle. The number of toggle bits varies from 1 to 32 depending on counting value. The more bits toggling at same cycle consume more power comparatively. SAIF files are

generated for each cycle and the power estimation results are illustrated in Figure 2.8. At the first cycle, only one bit toggles which leads to lowest power consumption. At the cycle when counting value equals to 0x1000, there are 13 bits toggling together. The power consumption rises to more than 200 mW. Overall the power consumption of counter is in proportion to the number of toggling bits and the instantaneous power estimation in Vivado provides a reliable and accurate result.

Chapter 3

CAM Implementation Strategies on FPGA

As a case study, a 256-bit wide RAM-based CAM with 32 entries is to be implemented on FPGA. Besides conventional scheme, the two optimized schemes pipelining and precomputation for reducing CAM power consumption at the architectural level will be reviewed and implemented.

3.1 Background

3.1.1 Basic Approaches

There are three basic approaches to CAM implementation based on FPGA. The first approach adopts registers as basic CAM cell. The value of CAM is stored in registers and a group of registers can be configured as a CAM array. The second approach is based on embedded RAM blocks, which can be configured and implemented with desired width and depth. The values of CAM array are written into RAM blocks and read out for comparison every search cycle. The last implementation approach uses Look-Up Table (LUT) for implementation instead of sequential logic circuits. It applies to where the value of CAM is known and fixed. Since CAM content is constant, it is optimized and combined into comparison circuit. A LUT embedded in FPGA is configured as dedicated

circuits with CAM value of 5'b01110 shown in Figure 3.1. However, it cannot be updated or modified after implementation, thus such a technique is not appropriate for implementing adaptive CAMs.

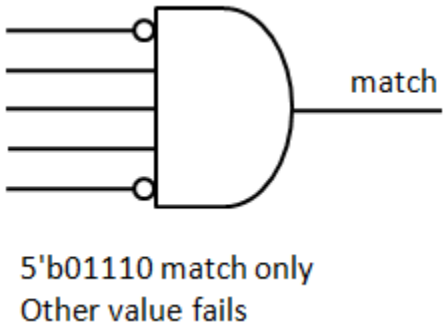


Figure 3.1 CAM cell based on LUT

3.1.2 Comparison between basic approaches

The work in [2] discussed and elaborated the three approaches above. Through comparison based on performance and utilization figure, it gave a conclusion that the RAM based design can provide much larger number of entries than the register and LUT based designs without sacrificing speed [2]. Furthermore, registers and LUTs are distributed across whole chip as basic logic resource. As the capacity of CAM increases, the large number of registers or LUTs configured will lead to complex interconnection as well as placement and routing. The wire delay will be considerable and much more power will be consumed by interconnection. Block RAM (BRAM) is a configurable memory module embedded on FGPA. It is provided as a physical IP core which is pre-

designed and optimized in terms of power and area. Each BRAM stores up to 36K bits of data and it greatly reduces the complexity of placement and routing for designs with a large volume of stored data. Hence placement and routing brings less impact to RAM based design comparatively. The CAM implementation in this report will be based on BRAMs.

3.1.3 Target device for implementation

Xilinx Artix-7 series FPGA is one of best choices for cost-sensitive solutions because it provides advanced functionality for high-performance applications with just half of power consumption of the previous generation [13]. The width of the input key is fixed as 256 bits. With several pins for control such as clock, read and write, the required number of total available IOs is around 270. For conventional scheme, every two CAM words are stored in each RAM (256 bits x 2), so it requires 16 BRAMs for 32 entries. For pipelining scheme, RAM is divided into four segments, thus 64 BRAMs are required. Taking resource requirement into consideration, XA7A75T is selected as target device which provides 285 available IOs and 105 BRAMs [14] [15].

3.2 Conventional Scheme

3.2.1 Architecture of conventional scheme

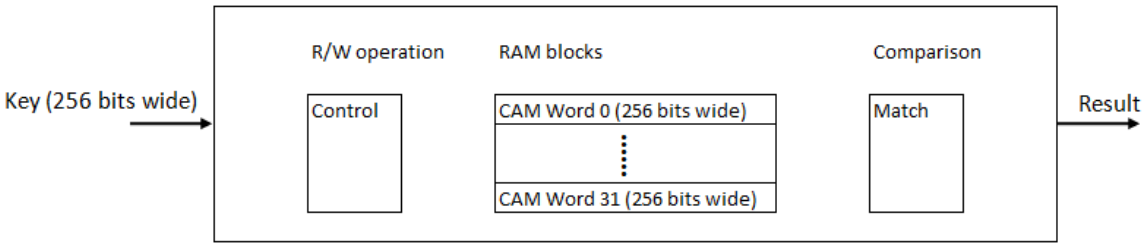


Figure 3.2 CAM structure of conventional scheme

The BRAM in Xilinx FPGA can be configured as an integral RAM block with desired width and depth, as shown in Figure 3.2. But it will take at least 32 cycles for one search operation, because only one CAM word in this RAM block can be read at one cycle. The operation latency is proportioned to the number of entries. Hence it configures multiple RAM blocks as shown in Figure 3.3, so data from different RAM can be accessed at same cycle.

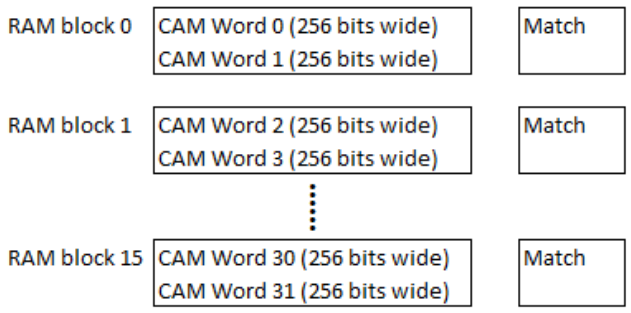


Figure 3.3 RAM blocks in conventional scheme

There are total 16 RAM blocks configured in this design. They are 256-bit wide in depth of two. So every two CAM words are stored in each block which is shown in Figure 3.4.

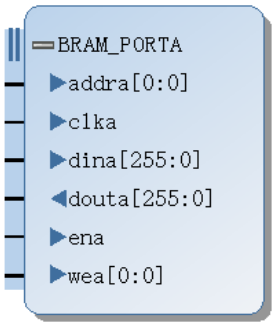


Figure 3.4 RAM block (256 bits x 2) diagram

3.2.2 Timing of the conventional scheme

It takes two cycles to read all CAM words and output a matching signal at the third cycle, as shown in Figure 3.5. This CAM module reads new data and outputs matching signal every cycle, as if operates as a 2-stage pipeline.

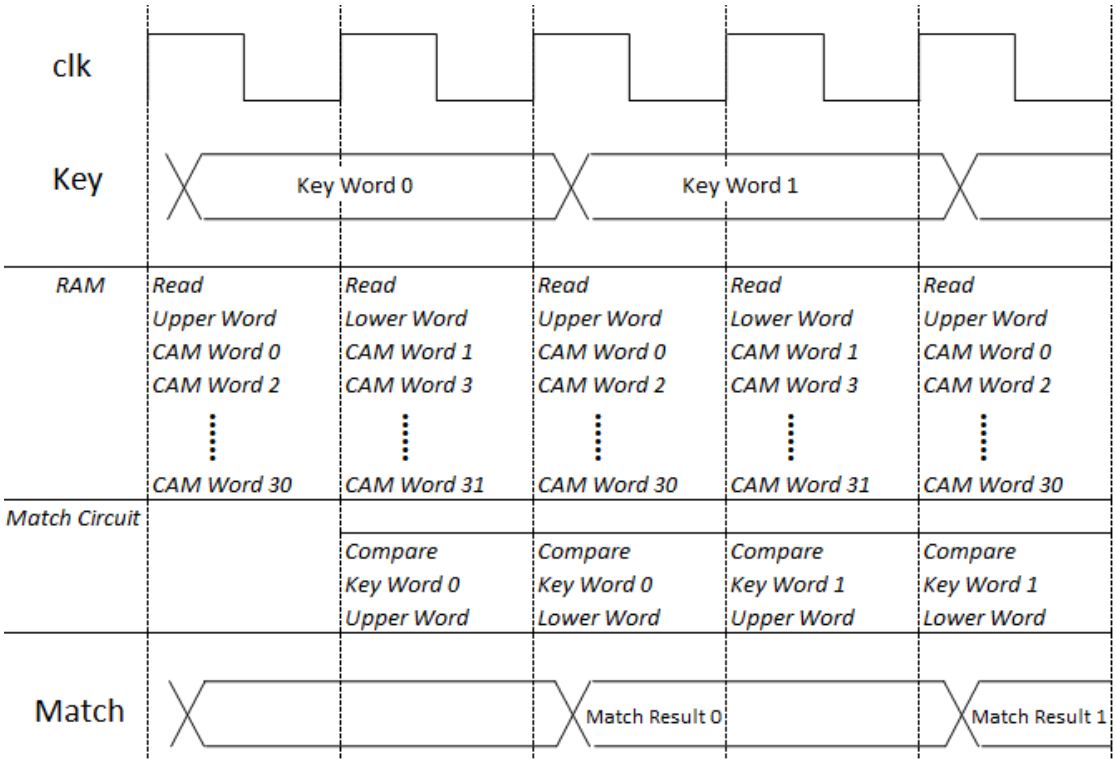


Figure 3.5 Timing diagram of conventional scheme

3.2.3 Resource utilization of conventional scheme

The target frequency for synthesis and implementation is set to 100MHz. In Figure 3.6, it is apparent that the RAM blocks take up more than half of BRAM resource in the entire device. Most IO resource is used to download the 256-bit wide input key. Since other logic is very small, only a tiny number of FF and LUT are utilized.

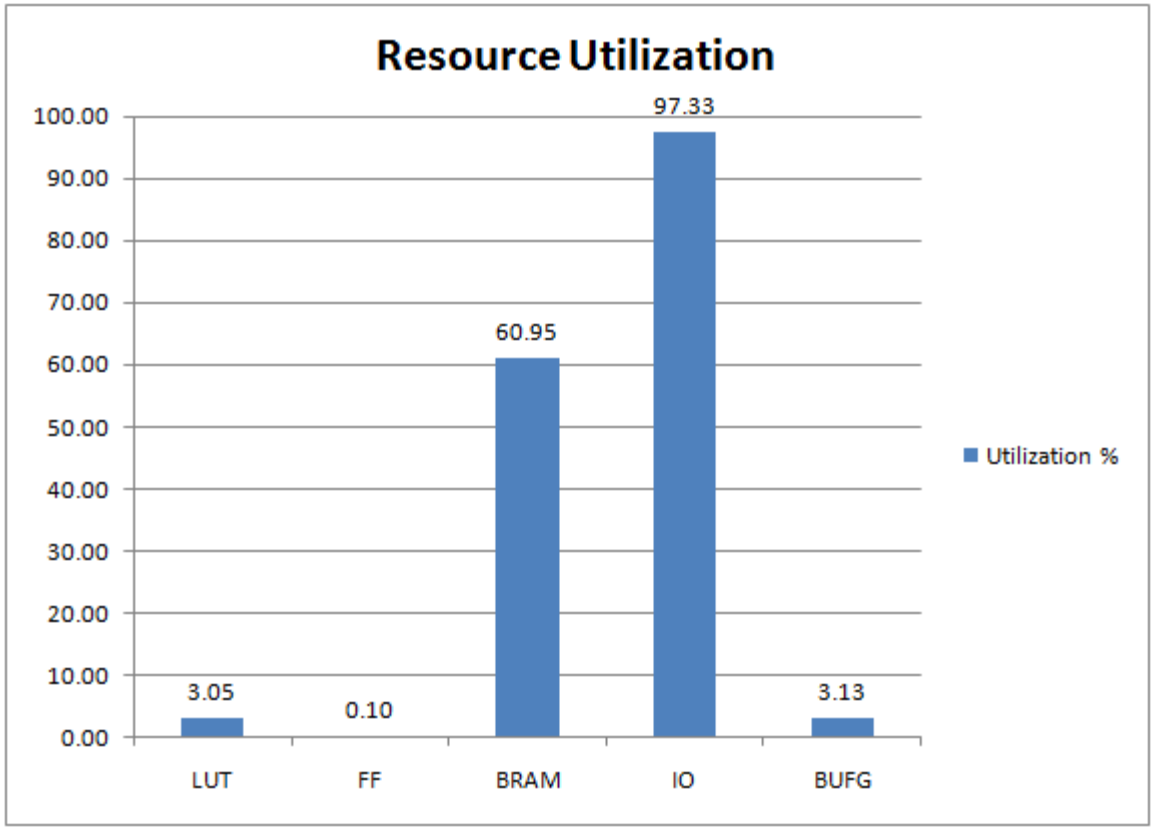


Figure 3.6 Resource utilization of conventional scheme

3.3 Low-Power Scheme

In order to compare power estimation between the conventional scheme and low-power schemes, a CAM with the same width and depth will be implemented on the same target device.

There are some low-power schemes proposed at transistor level for ASIC design. But these schemes for reducing power consumption do not apply to CAM design based on FPGA, because FPGA doesn't support circuit design at transistor level. The low-power

solutions through optimization on system architecture will be discussed and implemented.

3.3.1 Pipelining scheme

In [4], the architecture of pipelined match-lines was presented. The CAM word is divided into several segments with their own match circuits. The words that match a segment enable the search operation in their subsequent segments while the words that mismatch a segment disable their subsequent segments and therefore it saves power from reduced active resources [5]. Figure 3.7 outlines this process. It is discussed based on NOR-based CAM cell and match-lines using CMOS transistors in [5]. Nevertheless, this method can be adapted to use BRAMs in FPGA.

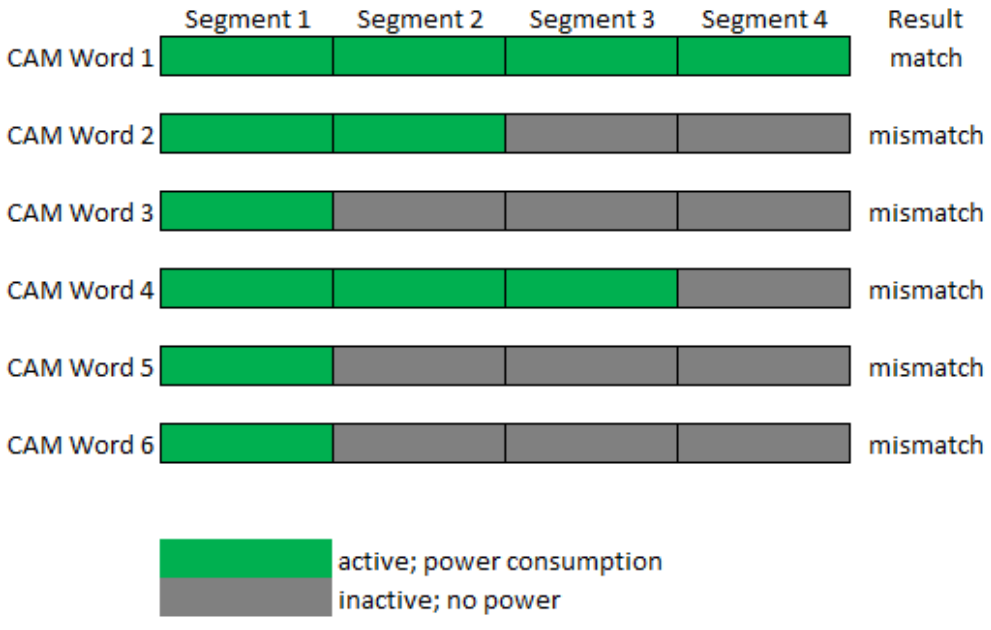


Figure 3.7 An example of active resource in pipelining scheme

Architecture of pipelining scheme

In our design, 256-bit wide CAM word is divided into four segments, with each a word length of 64 bits, as shown in Figure 3.8. Hence the whole CAM array with 32 entries is modified as shown Figure 3.9.

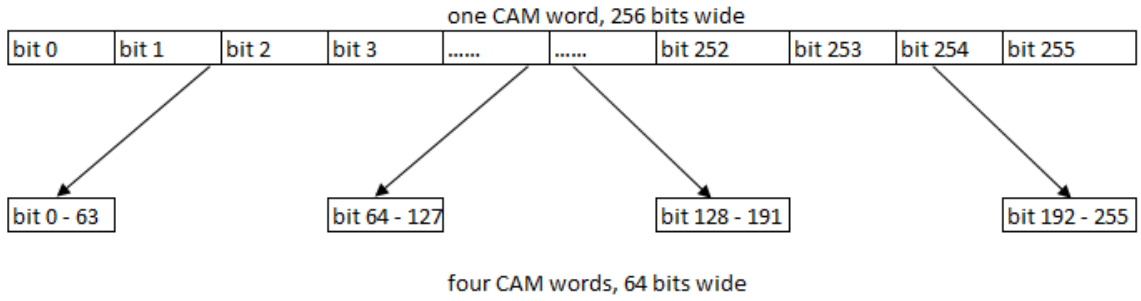


Figure 3.8 A CAM word divided into segments in pipelining scheme

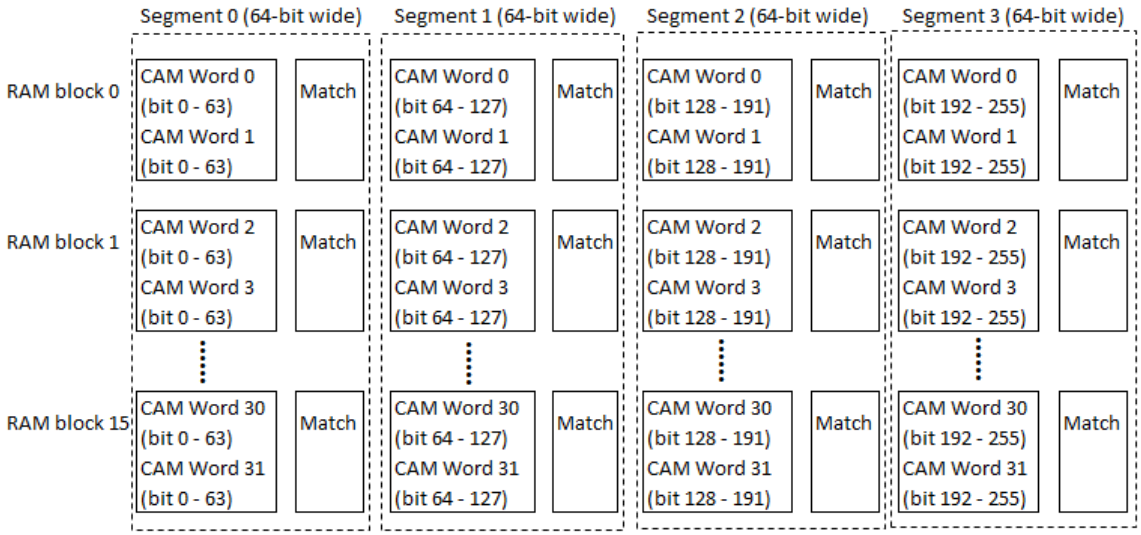


Figure 3.9 RAM blocks in pipelining scheme

Each segment contains 16 RAM blocks and respective matching circuits. It works independently as a smaller individual CAM array. In the first cycle of the search operation, the first segment is active for data lookup and output result. The mismatched CAM word will disable the corresponding CAM word in the second segment before the second cycle of the search operation starts. So data look-up will not be automatically triggered for the rest of RAM blocks. The matching signal output from the previous segment is used to activate input signals connected to the next segment. Figure 3.10 describes the activation logic between segments.

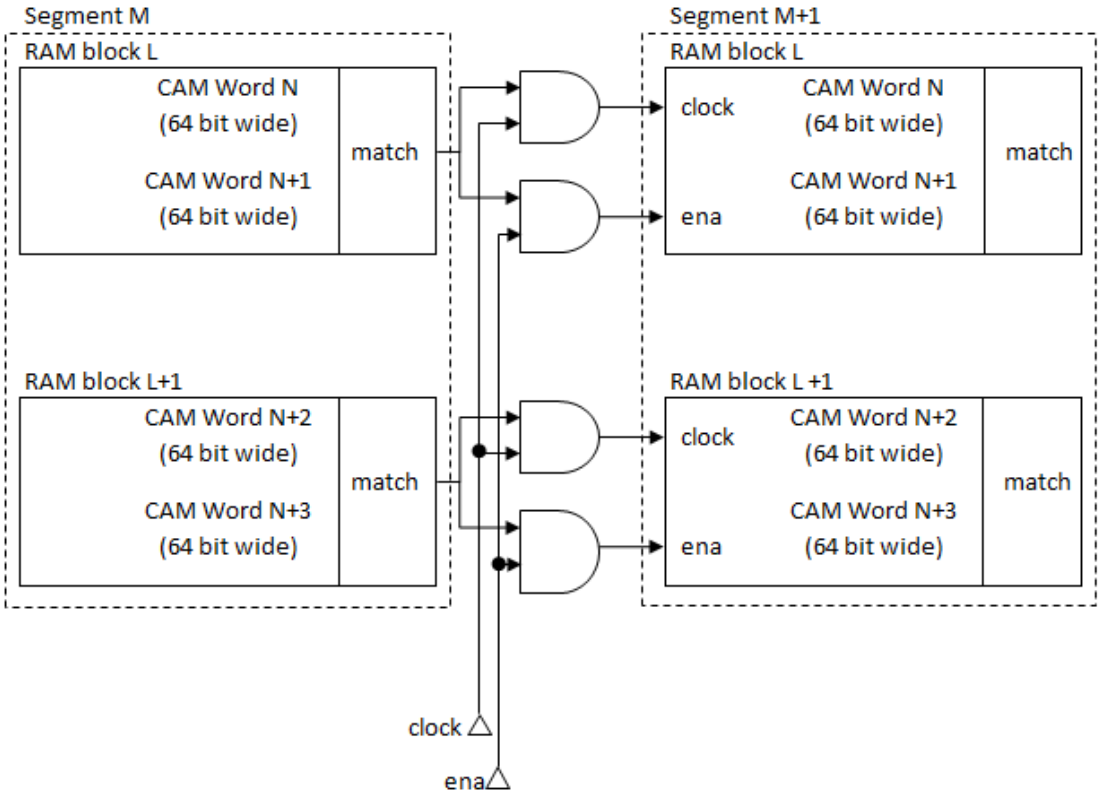


Figure 3.10 Connection between segments in pipelining scheme

Timing of pipelining scheme

In the pipelining scheme, a complete search operation is divided into four cycles, as shown in Figure 3.11. Each segment starts to work only after the previous segment finish data look-up. If outputs from the previous segment show a mismatch, the corresponding RAM blocks in next segment will keep inactive through the remaining cycles since their input signals are disabled.

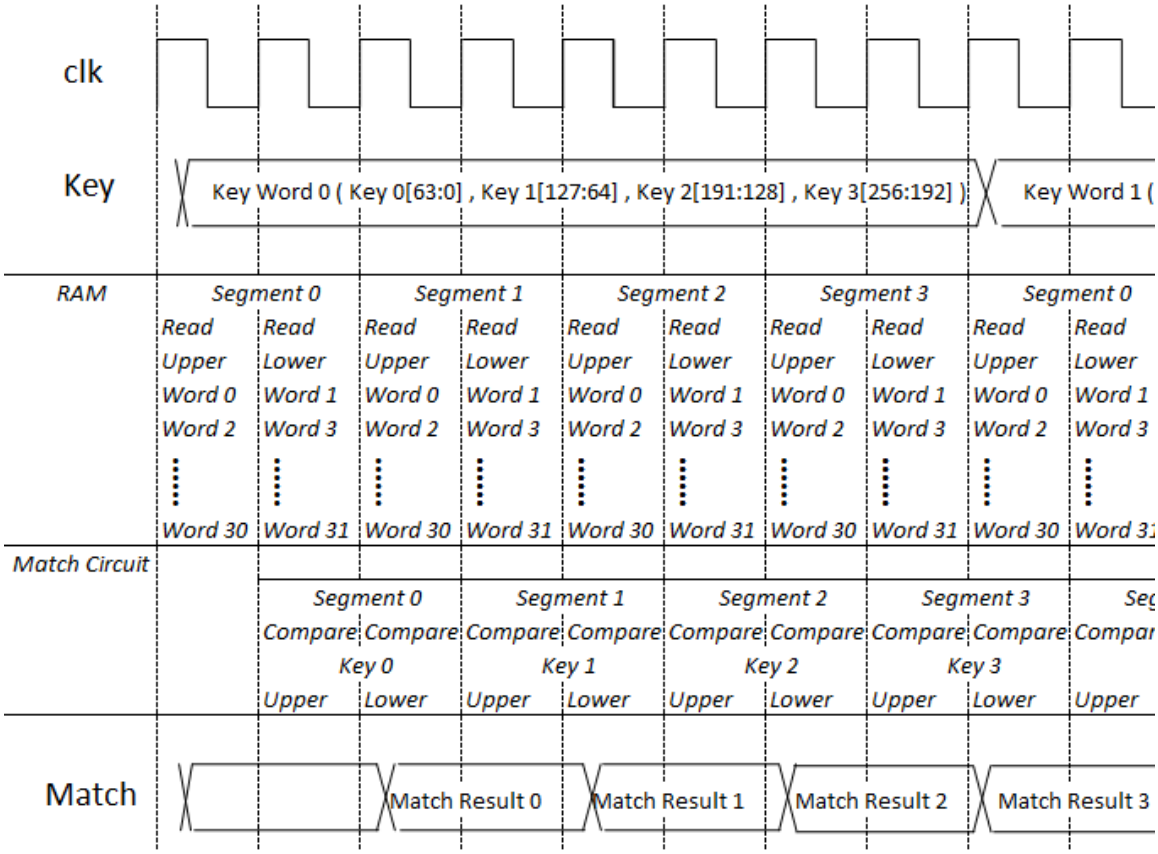


Figure 3.11 Timing diagram of pipelining scheme

Resource utilization of pipelining scheme

The target frequency for synthesis and implementation is set to 100MHz. Besides, all other strategies and constraints which were set in synthesis, placement and routing are same as that in the conventional scheme. From Figure 3.12, it is apparent that the resource utilization is almost same as the result in the conventional scheme. The pipelining scheme requires larger logic circuits such as duplicated matching circuits because of the multiple CAM segments employed. Hence there is a small increase in resource utilization of LUT and FF. But they still take up a tiny number of resources comparing to the whole design.

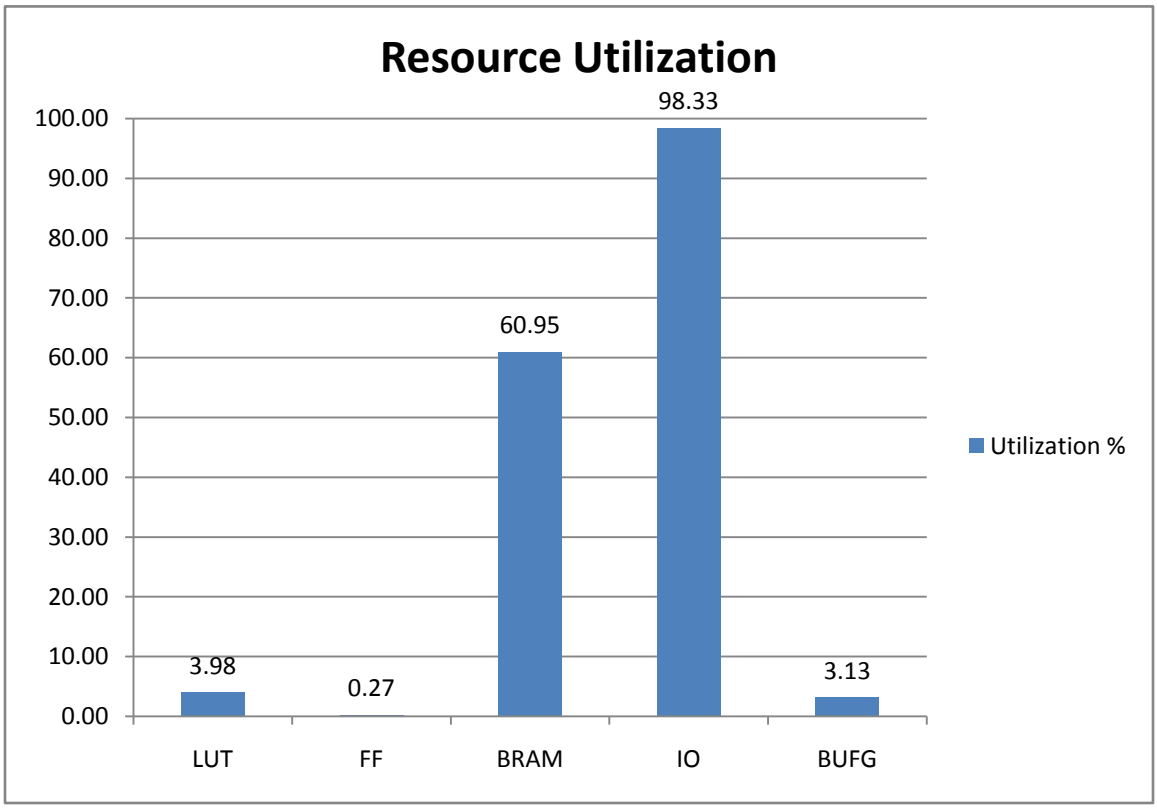


Figure 3.12 Resource utilization of pipelining scheme

3.3.2 Precomputation scheme

In conventional CAM scheme, data stored in every CAM cell is read and compared with the input key in parallel during each search cycle. As the CAM capacity grows, this operation leads to a large number of circuits active at the same time, which translates into a large power consumption. One of best solutions at architectural level is to reduce most of the comparison operations; such a strategy will save power during comparison [7].

In the pipelining implementation, it saves power by reducing the number of search operations through divided CAM segments. The majority of RAM blocks remain inactive due to mismatch from the search operation in previous segments. However, in the first search operation, every RAM block has to be active for data access and comparison. Besides for those input keys whose value is similar to data stored in CAM (e.g. only one bit value is different), some RAM blocks are probably active through all stages because mismatch occurs at the last segment.

In [6] [7], a solution based on precomputation was proposed and discussed. The basic method is to extract unique attributes of data by precomputation before the propoer search operation. Figure 3.13 shows the structure of precomputation logic. It includes extraction logic, extra memory for extracted information and comparison circuit. Every word stored in CAM goes through precomputation process while being written into CAM and its extracted information is stored in extra information memory such as RAM or registers. At the beginning of search operation, the input key is sent to precomputation

logic first. The attributes of the input key are extracted by the information extractor, and then the precomputation comparison circuits compare the attributes of the input key with the attributes of CAM words stored in the extra information memory in parallel [7]. The CAM words which mismatch with input key in precomputation comparison are disabled throughout the rest of operation. Hence the RAM blocks where mismatched CAM words are stored remain inactive without any data access and active resources for search operation are reduced.

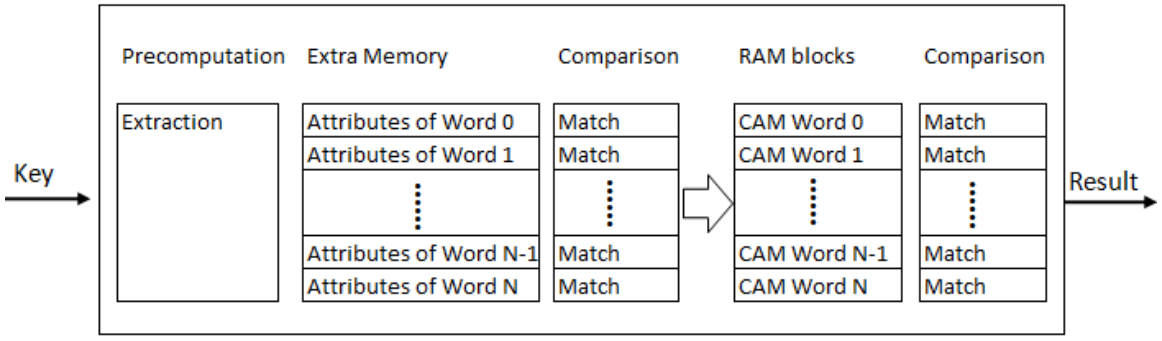


Figure 3.13 CAM structure of precomputation scheme

The precomputation process screens all CAM words for mismatched data at the beginning, without reading CAM cells which leads to major power consumption. Unlike the pipelining scheme, the precomputation scheme is independent of CAM array. It doesn't need to change the structure of CAM array and applies to CAMs with different implementation methodology.

The efficiency of precomputation varies depending on specific attributes to be extracted. The more complex and accurate information extraction filters out more mismatched data before the search operation. But meanwhile, it demands larger circuits and complicated design which bring impacts on power consumption and performance inversely. Thus the design of precomputation needs a concise and efficient information extraction. Some functions are used to realize the information extraction in the proposed CAM architecture, such as ones count function, parity function, and remainder function [7].

In this report, the precomputation based on ones count function will be implemented. Firstly its design is small comparing to other choices such as remainder function which requires a divider. Moreover, it is able to filter out mismatched data efficiently with a compact memory space for extracted information. For example, there are N+1 types of information extracted (the number of ones count varies from 0 to N) from an N-bit data. The width of extracted information is up to $(\log_2 N)$ +1. Taking an 8-bit wide CAM as an example, Figure 3.14 depicts the precomputation process.

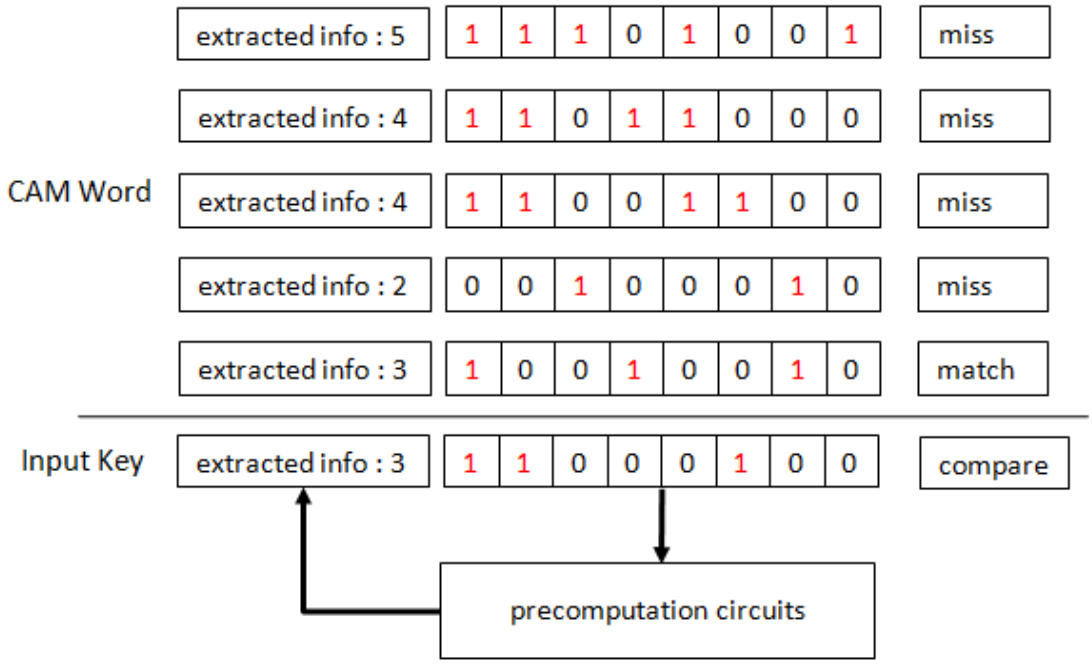


Figure 3.14 An example of precomputation process

Architecture of precomputation scheme

To reduce impacts on performance, the precomputation logic is comprised of combinational circuits only without any influence on original timing sequence. It is divided into 16 binary adders with 16 inputs in parallel shown in Figure 3.15.

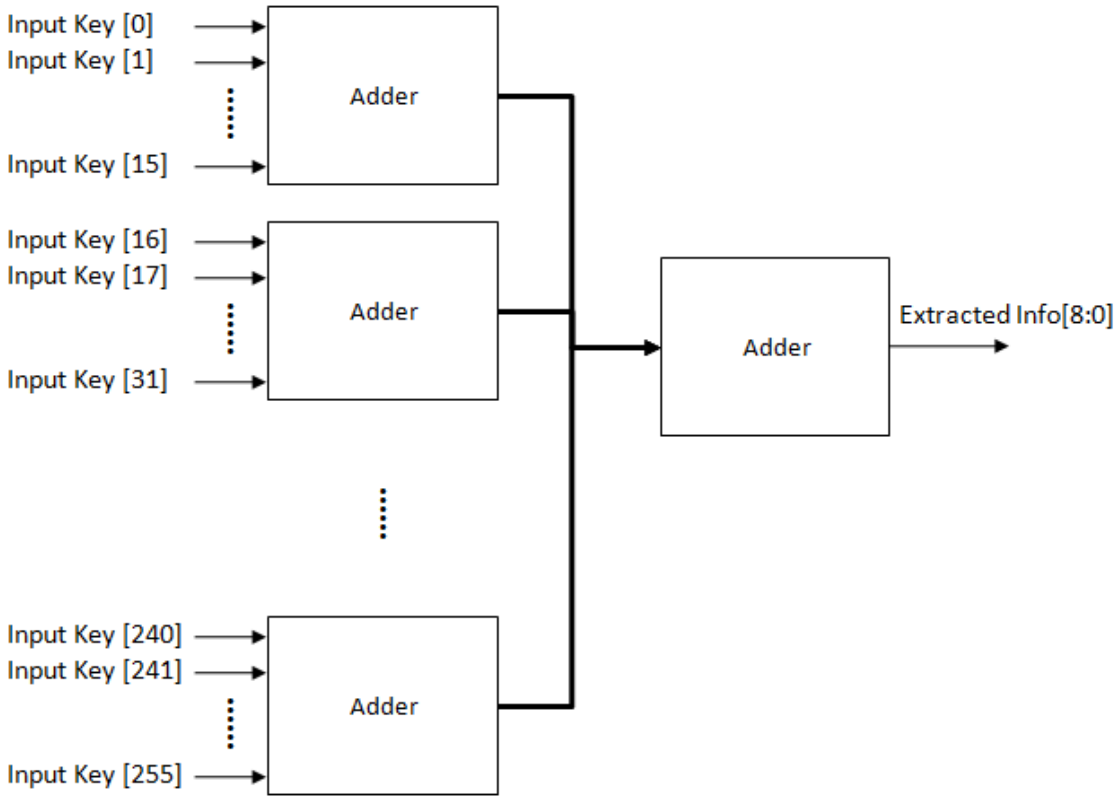


Figure 3.15 Precomputation logic based on adders

Since the precomputation scheme is independent of the structure of CAM array, the implementation is combined with the pipelining scheme in order to reduce power consumption further. The matching signals output from precomputation logic are used to activate the first segment of CAM array as shown in Figure 3.16. If a mismatch occurs in the precomputation process, the corresponding inputs to RAM blocks in the first segment are disabled, and corresponding RAM blocks in following segments remain inactive throughout the rest of search operation.

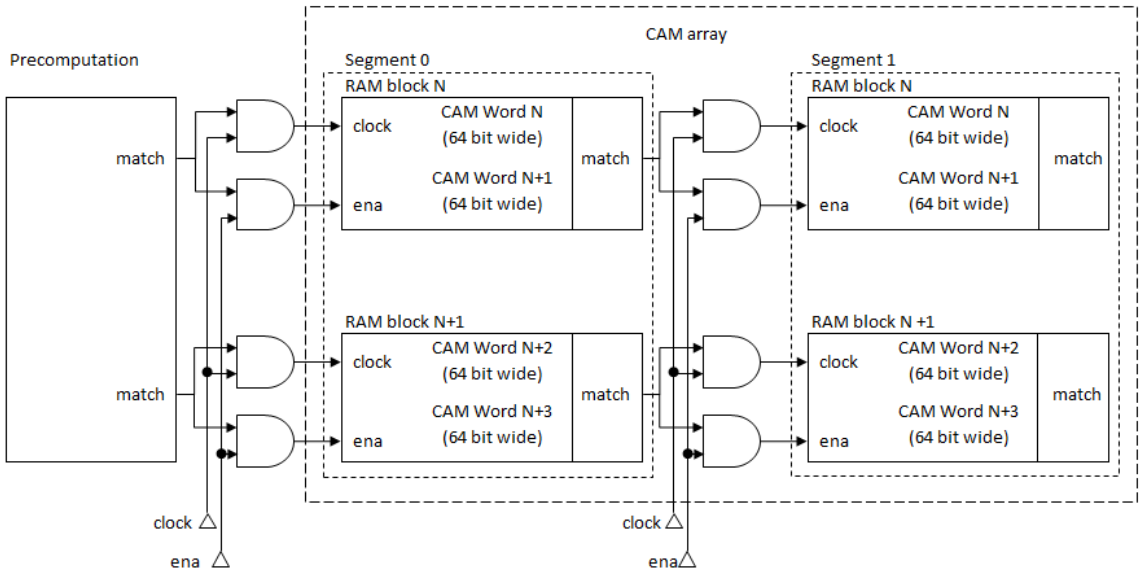


Figure 3.16 Connection of precomputation combined with pipelining scheme

Timing of precomputation scheme

Figure 3.17 shows the timing sequence of precomputation combined with the pipelining scheme. It is same as timing sequence of pipelining scheme except for the first cycle of precomputation. Theoretically, the precomputation process is able to be merged in the first cycle of the search operation in the pipelining scheme since the implementation of precomputation is based on combinational logic. But for better timing slack in placement and routing, one cycle is inserted for precomputation.

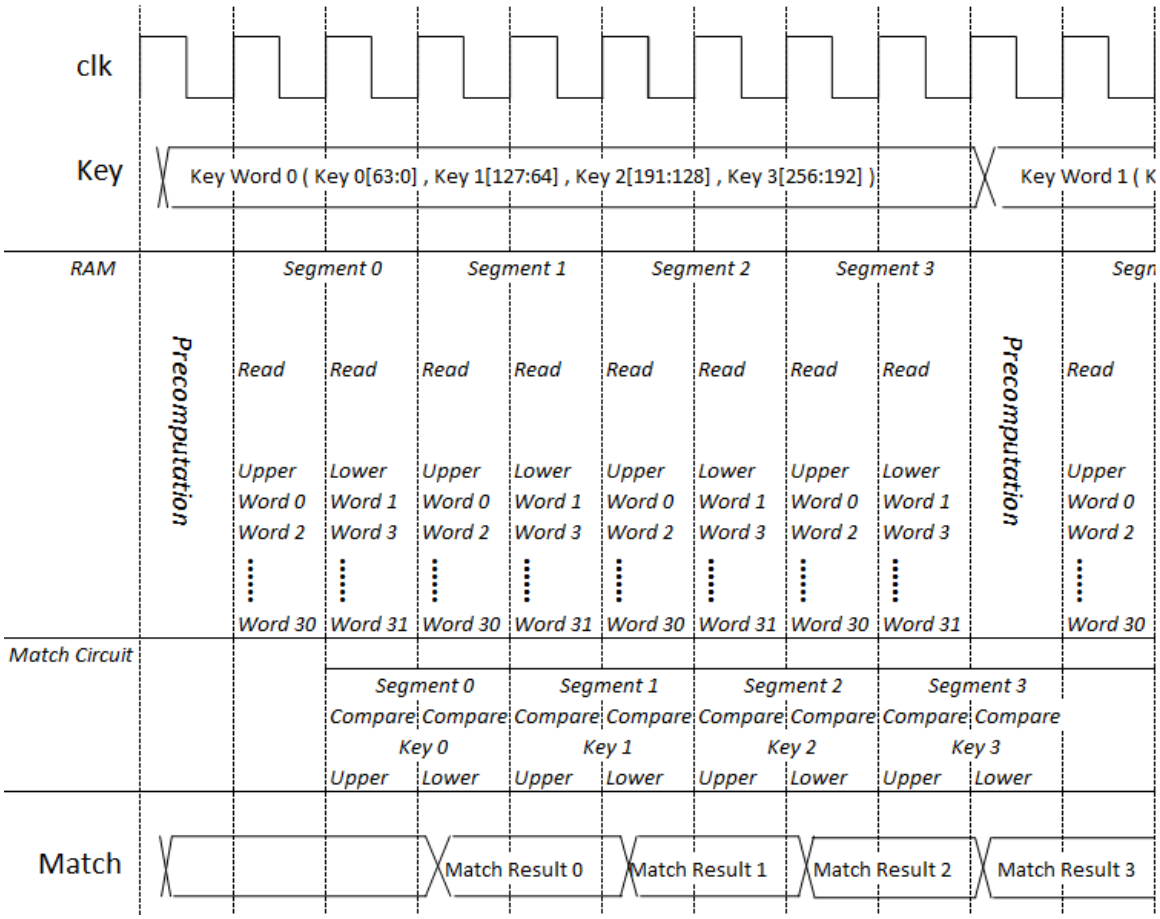


Figure 3.17 Timing diagram of precomputation combined with pipelining scheme

Resource utilization of precomputation scheme

The target frequency for synthesis and implementation is set to 100MHz. Besides, all other strategies and constraints which were set in synthesis, placement and routing are the same as that in the conventional scheme. From Figure 3.18, the resource utilization is almost same as the pipelining scheme. There is a slight increase in LUT and FF resources

due to adders in precomputation logic and extra memory space for extracted information from CAM words.

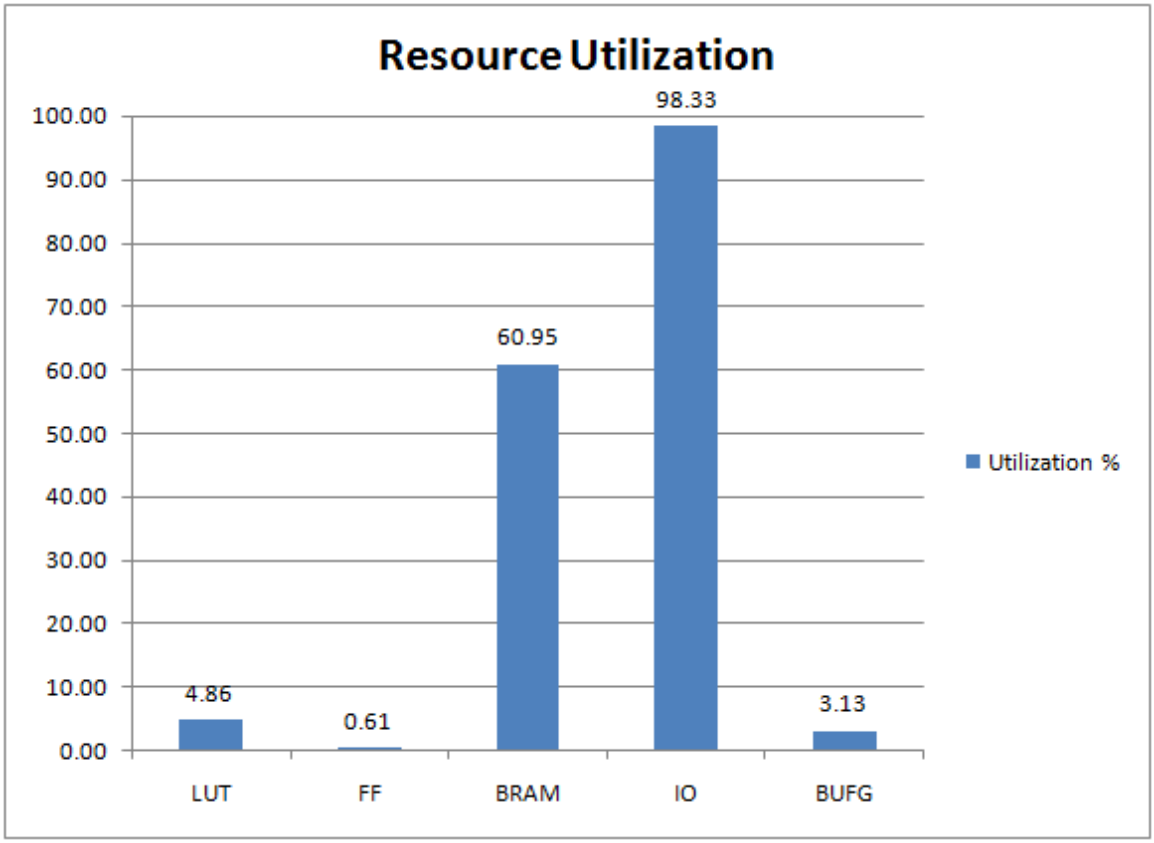


Figure 3.18 Resource utilization of precomputation combined with pipelining scheme

Chapter 4

Power Analysis of CAM on FPGA

To obtain accurate power analysis in Vivado, it requires two important input data elements. The first input is the complete netlist of CAM design generated after placement and routing stage. The second input is a SAIF file which is generated from the simulation. In order for accurate comparison in power consumption, same test scenario applies to all schemes to generate SAIF. The power estimation results are collected based on cycles and resources rather than a simple average number. So the power analysis can further illustrate how and where power is saved.

4.1 Test scenario

As discussed in Chapter 2, the most accurate power analysis in Vivado is vector-based power analysis with the netlist after placement and routing. Vector-based power analysis requires SAIF files which capture realistic switching activity of design. Thus a test scenario which represents typical operation is created for simulation. Figure 4.1 below shows the test flow of simulation.

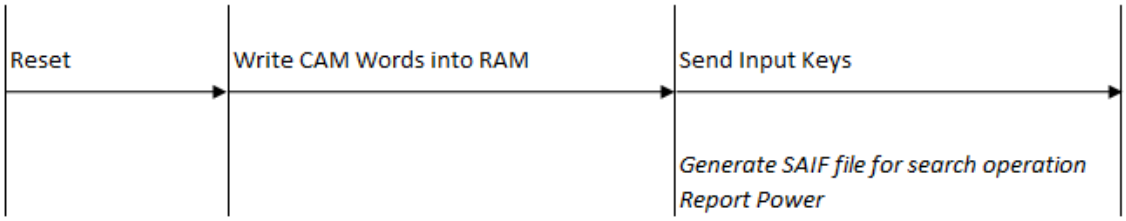


Figure 4.1 Test flow for power analysis

There are 32 words written into CAM before the search operation commences since it has 32 entries. The words stored in CAM are listed in Table 4.1. The first two columns from the left list the index numbers of RAM blocks and stored CAM words. Each row shows the value of a 256-bit CAM word in the form of four 64-bit segments.

Table 4.1 Value of CAM Words

RAM	CAM Words	[63:0]	[127:64]	[191:128]	[255:192]
RAM Block 0	CAM Word 0	64'h0101_0101_0101_0101	64'h0101_0101_0101_0101	64'h0101_0101_0101_0101	64'h0101_0101_0101_0101
	CAM Word 1	64'h0202_0202_0202_0202	64'h0202_0202_0202_0202	64'h0202_0202_0202_0202	64'h0202_0202_0202_0202
RAM Block 1	CAM Word 2	64'h0303_0303_0303_0303	64'h0303_0303_0303_0303	64'h0303_0303_0303_0303	64'h0303_0303_0303_0303
	CAM Word 3	64'h0404_0404_0404_0404	64'h0404_0404_0404_0404	64'h0404_0404_0404_0404	64'h0404_0404_0404_0404
RAM Block 2	CAM Word 4	64'h0505_0505_0505_0505	64'h0505_0505_0505_0505	64'h0505_0505_0505_0505	64'h0505_0505_0505_0505
	CAM Word 5	64'h0606_0606_0606_0606	64'h0606_0606_0606_0606	64'h0606_0606_0606_0606	64'h0606_0606_0606_0606
RAM Block 3	CAM Word 6	64'h0707_0707_0707_0707	64'h0707_0707_0707_0707	64'h0707_0707_0707_0707	64'h0707_0707_0707_0707
	CAM Word 7	64'h0808_0808_0808_0808	64'h0808_0808_0808_0808	64'h0808_0808_0808_0808	64'h0808_0808_0808_0808
RAM Block 4	CAM Word 8	64'h0909_0909_0909_0909	64'h0909_0909_0909_0909	64'h0909_0909_0909_0909	64'h0909_0909_0909_0909
	CAM Word 9	64'h0A0A_0A0A_0A0A_0A0A	64'h0A0A_0A0A_0A0A_0A0A	64'h0A0A_0A0A_0A0A_0A0A	64'h0A0A_0A0A_0A0A_0A0A
RAM Block 5	CAM Word 10	64'h0B0B_0B0B_0B0B_0B0B	64'h0B0B_0B0B_0B0B_0B0B	64'h0B0B_0B0B_0B0B_0B0B	64'h0B0B_0B0B_0B0B_0B0B
	CAM Word 11	64'h0C0C_0C0C_0C0C_0C0C	64'h0C0C_0C0C_0C0C_0C0C	64'h0C0C_0C0C_0C0C_0C0C	64'h0C0C_0C0C_0C0C_0C0C
RAM Block 6	CAM Word 12	64'h0D0D_0D0D_0D0D_0D0D	64'h0D0D_0D0D_0D0D_0D0D	64'h0D0D_0D0D_0D0D_0D0D	64'h0D0D_0D0D_0D0D_0D0D
	CAM Word 13	64'h0E0E_0E0E_0E0E_0E0E	64'h0E0E_0E0E_0E0E_0E0E	64'h0E0E_0E0E_0E0E_0E0E	64'h0E0E_0E0E_0E0E_0E0E
RAM Block 7	CAM Word 14	64'h0F0F_0F0F_0F0F_0F0F	64'h0F0F_0F0F_0F0F_0F0F	64'h0F0F_0F0F_0F0F_0F0F	64'h0F0F_0F0F_0F0F_0F0F
	CAM Word 15	64'h1010_1010_1010_1010	64'h1010_1010_1010_1010	64'h1010_1010_1010_1010	64'h1010_1010_1010_1010
RAM Block 8	CAM Word 16	64'h1111_1111_1111_1111	64'h1111_1111_1111_1111	64'h1111_1111_1111_1111	64'h1111_1111_1111_1111
	CAM Word 17	64'h1212_1212_1212_1212	64'h1212_1212_1212_1212	64'h1212_1212_1212_1212	64'h1212_1212_1212_1212
RAM Block 9	CAM Word 18	64'h1313_1313_1313_1313	64'h1313_1313_1313_1313	64'h1313_1313_1313_1313	64'h1313_1313_1313_1313
	CAM Word 19	64'h1414_1414_1414_1414	64'h1414_1414_1414_1414	64'h1414_1414_1414_1414	64'h1414_1414_1414_1414
RAM Block 10	CAM Word 20	64'h1515_1515_1515_1515	64'h1515_1515_1515_1515	64'h1515_1515_1515_1515	64'h1515_1515_1515_1515
	CAM Word 21	64'h1616_1616_1616_1616	64'h1616_1616_1616_1616	64'h1616_1616_1616_1616	64'h1616_1616_1616_1616
RAM Block 11	CAM Word 22	64'h1717_1717_1717_1717	64'h1717_1717_1717_1717	64'h1717_1717_1717_1717	64'h1717_1717_1717_1717
	CAM Word 23	64'h1818_1818_1818_1818	64'h1818_1818_1818_1818	64'h1818_1818_1818_1818	64'h1818_1818_1818_1818
RAM Block 12	CAM Word 24	64'h1919_1919_1919_1919	64'h1919_1919_1919_1919	64'h1919_1919_1919_1919	64'h1919_1919_1919_1919
	CAM Word 25	64'h1A1A_1A1A_1A1A_1A1A	64'h1A1A_1A1A_1A1A_1A1A	64'h1A1A_1A1A_1A1A_1A1A	64'h1A1A_1A1A_1A1A_1A1A
RAM Block 13	CAM Word 26	64'h1B1B_1B1B_1B1B_1B1B	64'h1B1B_1B1B_1B1B_1B1B	64'h1B1B_1B1B_1B1B_1B1B	64'h1B1B_1B1B_1B1B_1B1B
	CAM Word 27	64'h1C1C_1C1C_1C1C_1C1C	64'h1C1C_1C1C_1C1C_1C1C	64'h1C1C_1C1C_1C1C_1C1C	64'h1C1C_1C1C_1C1C_1C1C
RAM Block 14	CAM Word 28	64'h1D1D_1D1D_1D1D_1D1D	64'h1D1D_1D1D_1D1D_1D1D	64'h1D1D_1D1D_1D1D_1D1D	64'h1D1D_1D1D_1D1D_1D1D
	CAM Word 29	64'h1E1E_1E1E_1E1E_1E1E	64'h1E1E_1E1E_1E1E_1E1E	64'h1E1E_1E1E_1E1E_1E1E	64'h1E1E_1E1E_1E1E_1E1E
RAM Block 15	CAM Word 30	64'h1F1F_1F1F_1F1F_1F1F	64'h1F1F_1F1F_1F1F_1F1F	64'h1F1F_1F1F_1F1F_1F1F	64'h1F1F_1F1F_1F1F_1F1F
	CAM Word 31	64'h2020_2020_2020_2020	64'h2020_2020_2020_2020	64'h2020_2020_2020_2020	64'h2020_2020_2020_2020

Five input keys are sent to CAM for the search operation. Their values and corresponding results are shown in Table 4.2. The first two input keys mismatch all CAM words listed in Table 4.1. With the rest of three input keys, CAM is expected to perform complete search operations and output respective matching results. The same test scenario will apply to power analysis for each scheme.

Table 4.2 Value of input keys

Input Key	Segment	Value	Result
0	[63:0]	64'h0000_0000_0000_0000	Mismatch
	[127:64]	64'h0000_0000_0000_0000	
	[191:128]	64'h0000_0000_0000_0000	
	[255:192]	64'h0000_0000_0000_00FF	
1	[63:0]	64'h2121_2121_2121_2121	Mismatch
	[127:64]	64'h2121_2121_2121_2121	
	[191:128]	64'h2121_2121_2121_2121	
	[255:192]	64'h2121_2121_2121_2121	
2	[63:0]	64'h0303_0303_0303_0303	CAM Word 2
	[127:64]	64'h0303_0303_0303_0303	
	[191:128]	64'h0303_0303_0303_0303	
	[255:192]	64'h0303_0303_0303_0303	
3	[63:0]	64'h1D1D_1D1D_1D1D_1D1D	CAM Word 28
	[127:64]	64'h1D1D_1D1D_1D1D_1D1D	
	[191:128]	64'h1D1D_1D1D_1D1D_1D1D	
	[255:192]	64'h1D1D_1D1D_1D1D_1D1D	
4	[63:0]	64'h2020_2020_2020_2020	CAM Word 31
	[127:64]	64'h2020_2020_2020_2020	
	[191:128]	64'h2020_2020_2020_2020	
	[255:192]	64'h2020_2020_2020_2020	

4.2 Power Estimation in Conventional Scheme

It takes 11 cycles from the first input key sent into CAM to the last result output from CAM. SAIF files are generated for each cycle. Power estimation figures through cycles are shown in Figure 4.2. The power consumption of CAM varies from 264 mW to 415 mW through whole search operation. RAM blocks consume the majority of total CAM power in every cycle. The average power consumption of RAM blocks is 246 mW accounting for 78% of the total result. Because each RAM block is accessed for data fetching in every cycle, no matter whether the stored CAM words match input key or not. This point is illustrated in Figure 4.3. Each RAM block keeps active in every cycle and consumes power between 15 mW to 19 mW. For each search operation, there is only one matched CAM word. So the power consumed by the search operation with other 31 mismatched CAM words is wasted. To optimize power consumption, the number of active RAM blocks need to be reduced.

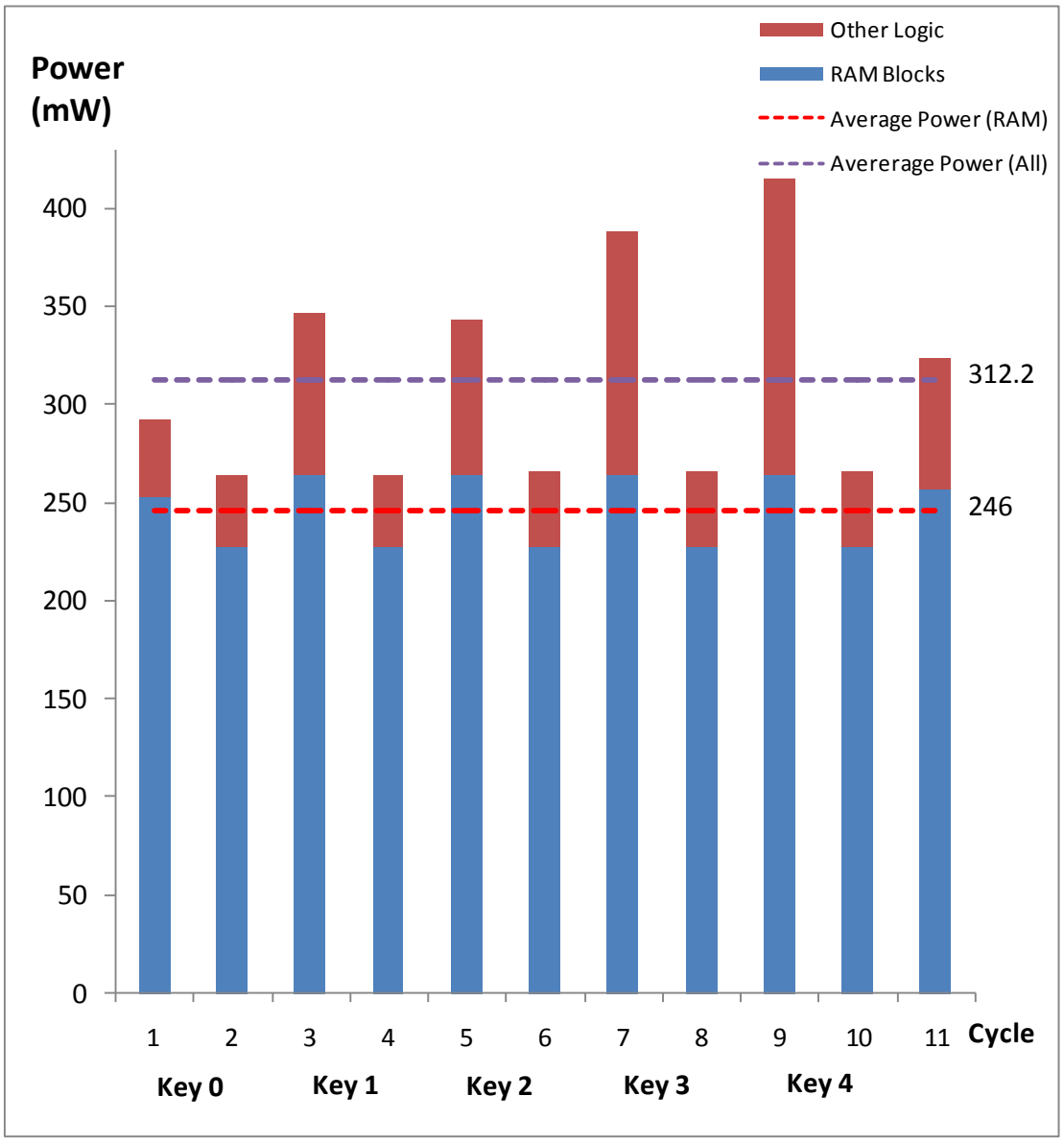


Figure 4.2 Power estimation through cycles in conventional scheme

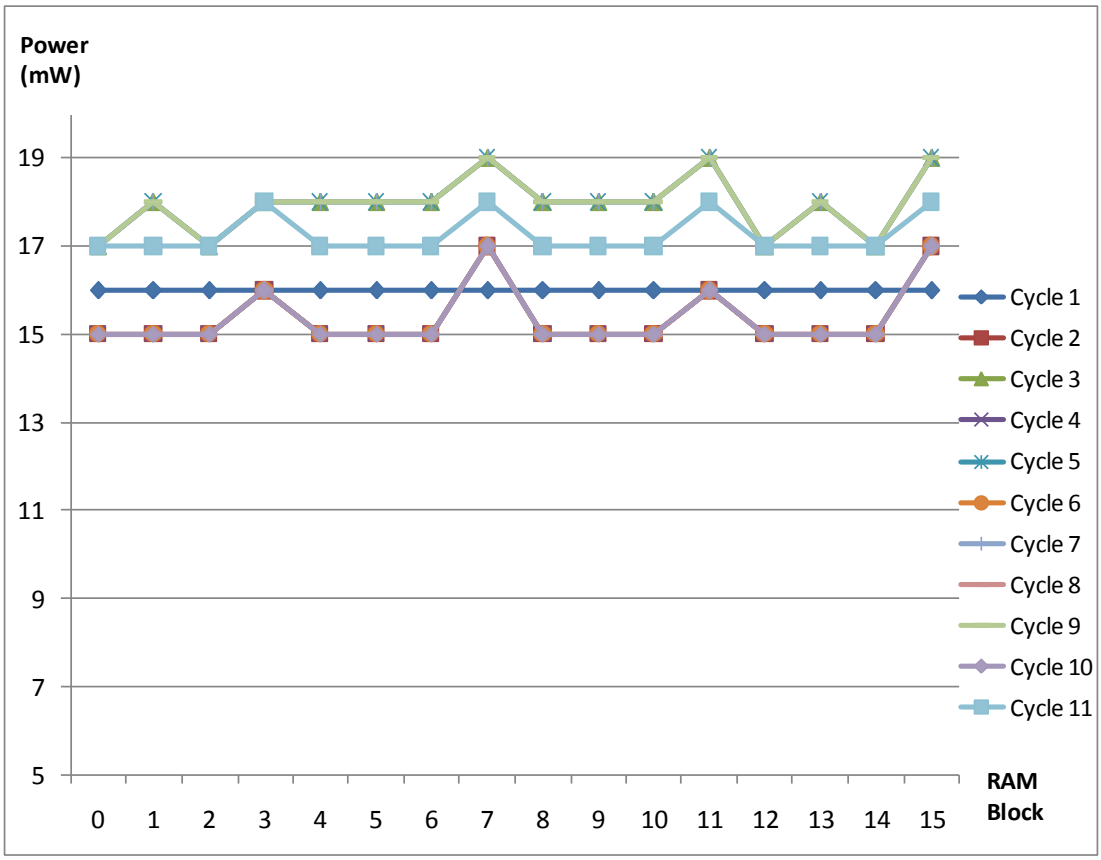


Figure 4.3 Power estimation through RAM blocks in conventional scheme

4.3 Power Estimation in Pipelining Scheme

The RAM blocks are divided into four segments in the pipelining scheme. Each segment takes 2 cycles for the search operation behaving like an individual CAM in the conventional scheme. Hence it takes 40 cycles to go through whole search operation in the same test scenario, and 8 cycles for each input key. The power estimation through cycles is shown in Figure 4.4. The power consumption of RAM blocks is greatly

decreased comparing to the conventional scheme. The proportion of power consumed by RAM blocks is reduced from 78% to 40%. The average total power decreases to 42.5 mW while it is up to 312.2 mW in the conventional scheme. The sharp decline in power consumption is due to reduced active RAM blocks in the search operation. Most of the power consumed by RAM blocks focuses on the first segment because all of the RAM blocks in the first segment are accessed. Then the RAM blocks of mismatched CAM words are disabled by the previous search result, thus there is a huge drop in power consumption after the search operation in the first segment.

The power estimation through segments is shown from Figure 4.5 to Figure 4.9. The power consumption distribution is depicted through the search operation of each input key. For key 0 and key 1, they do not match any CAM words. After the search operation in the first segment, the rest of segments are disabled by mismatch signal from the previous operation. Hence they remain inactive and consume very tiny power through all cycles. For key 2, key 3 and key 4, the first segment is active in first search operation but only the matched RAM blocks in the rest of segments are enabled and active for data access. For example, in Figure 4.9, there are three rises in power consumption from segment 1, segment 2 and segment 3 respectively. Because key 4 matches the 31st CAM word, and RAM block 15 in Table 4.1 is enabled and active through the whole period. Most of the mismatched RAM blocks are not activated which saves much power comparing to the conventional scheme.

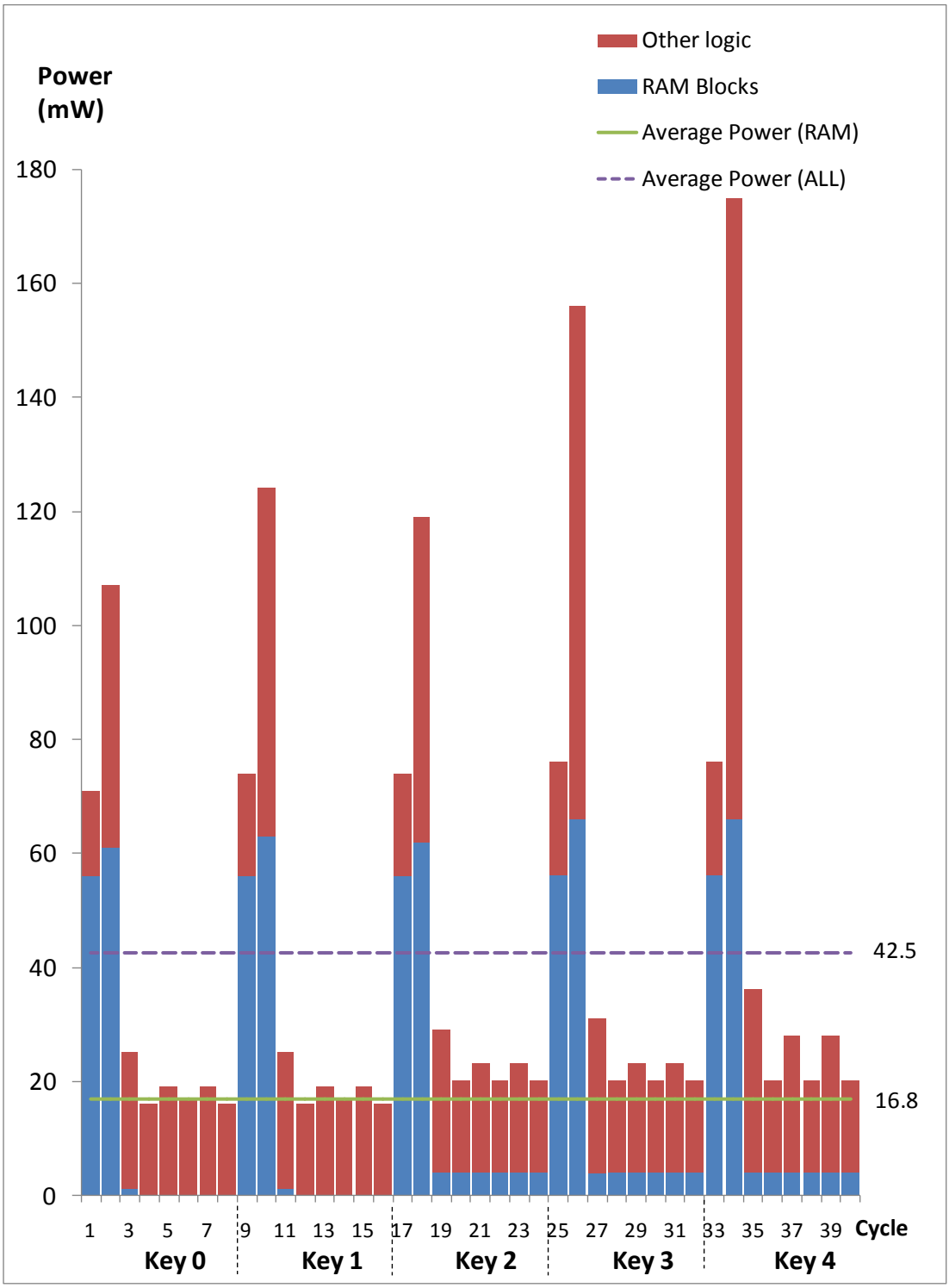


Figure 4.4 Power estimation through cycles in pipelining scheme

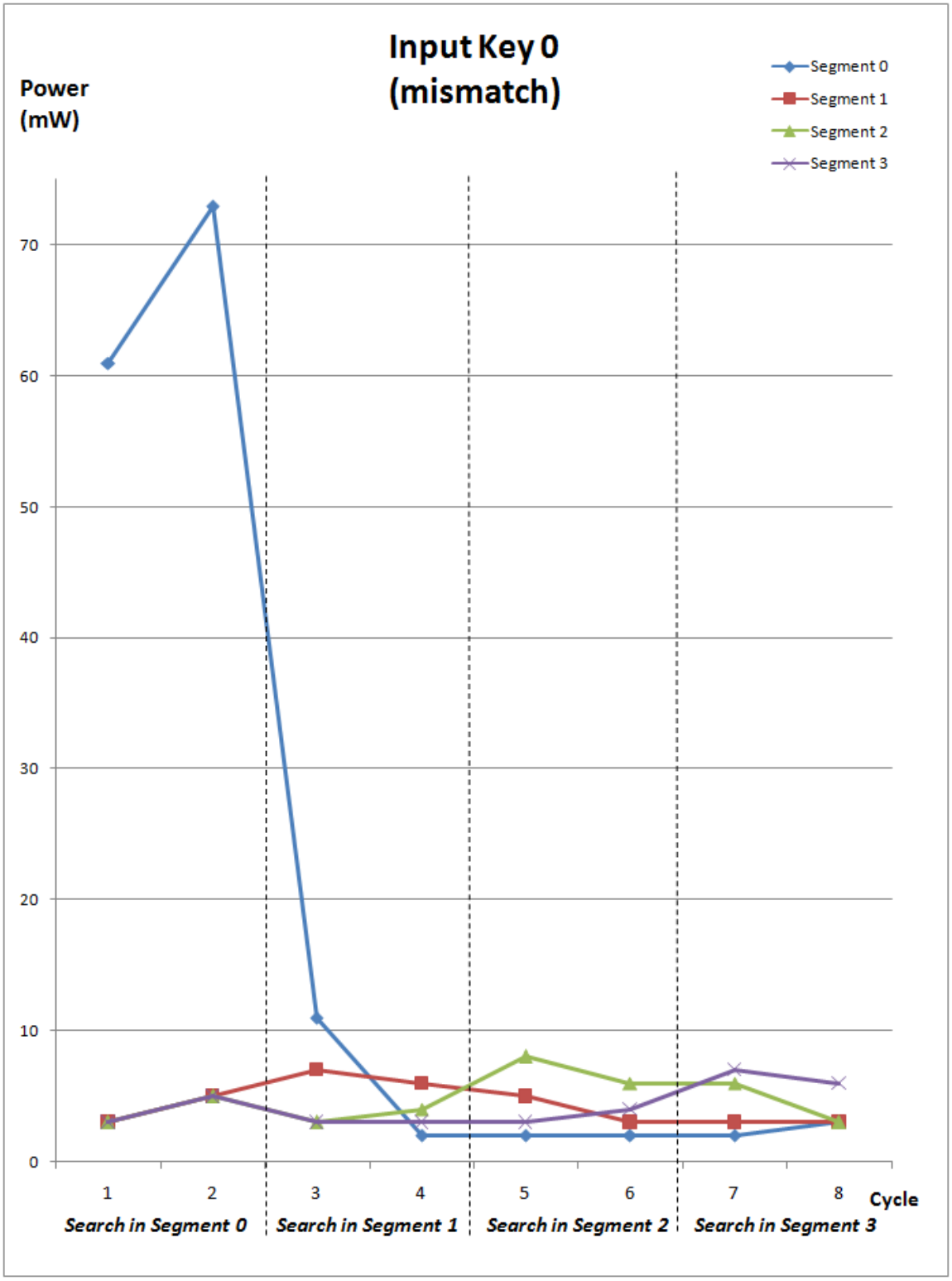


Figure 4.5 Power estimation in the period of input key 0 in pipelining scheme

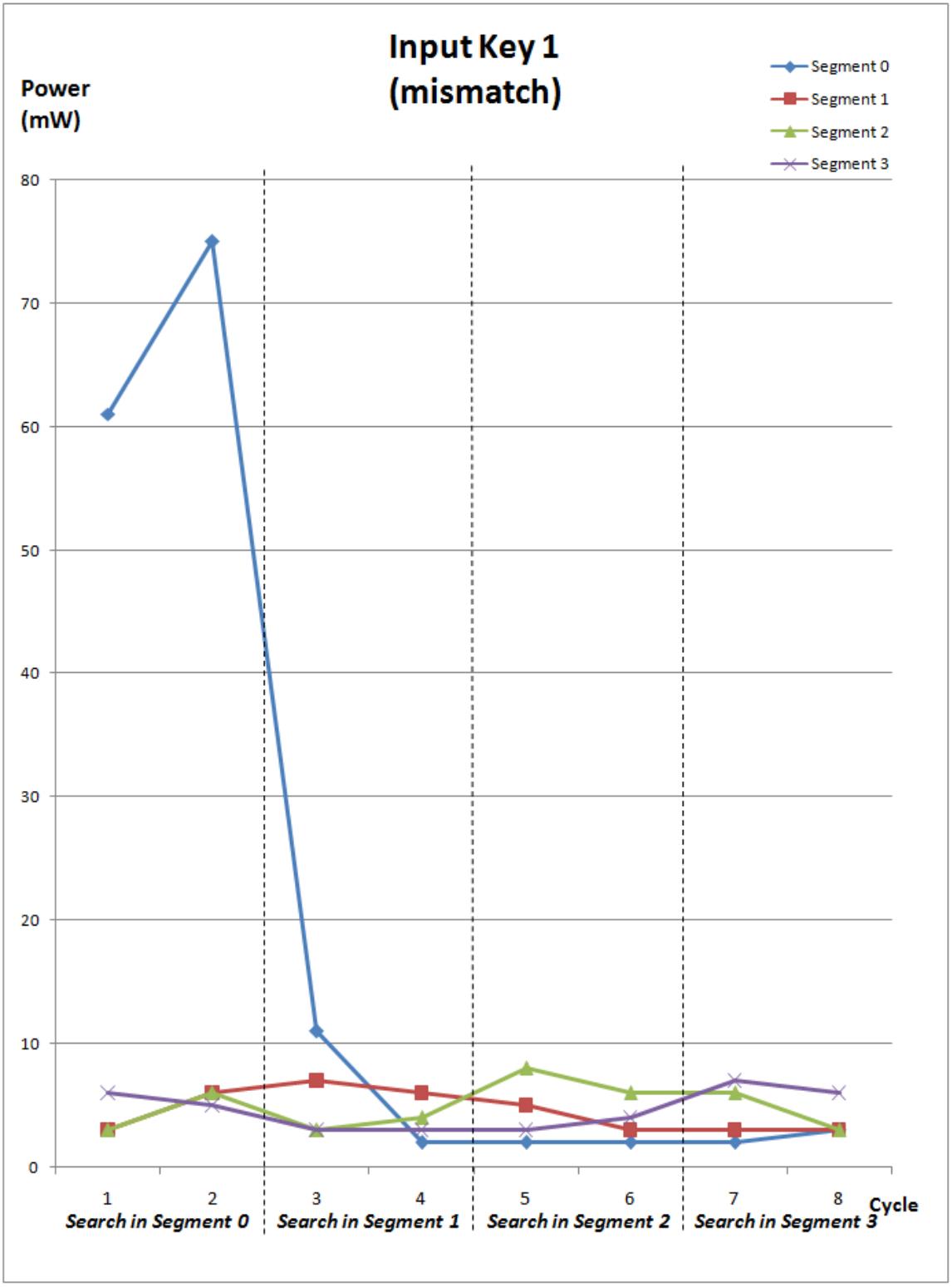


Figure 4.6 Power estimation in the period of input key 1 in pipelining scheme

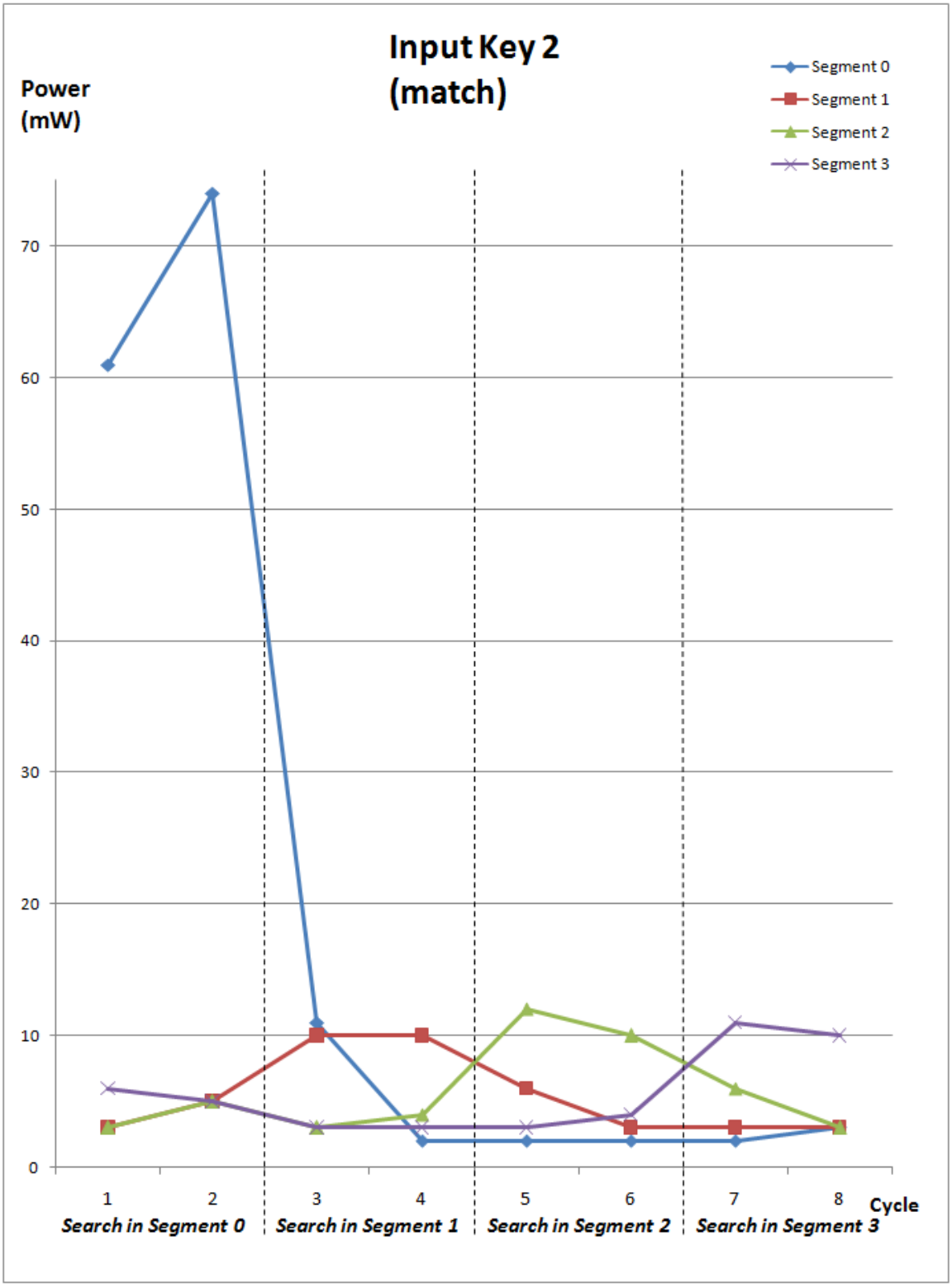


Figure 4.7 Power estimation in the period of input key 2 in pipelining scheme

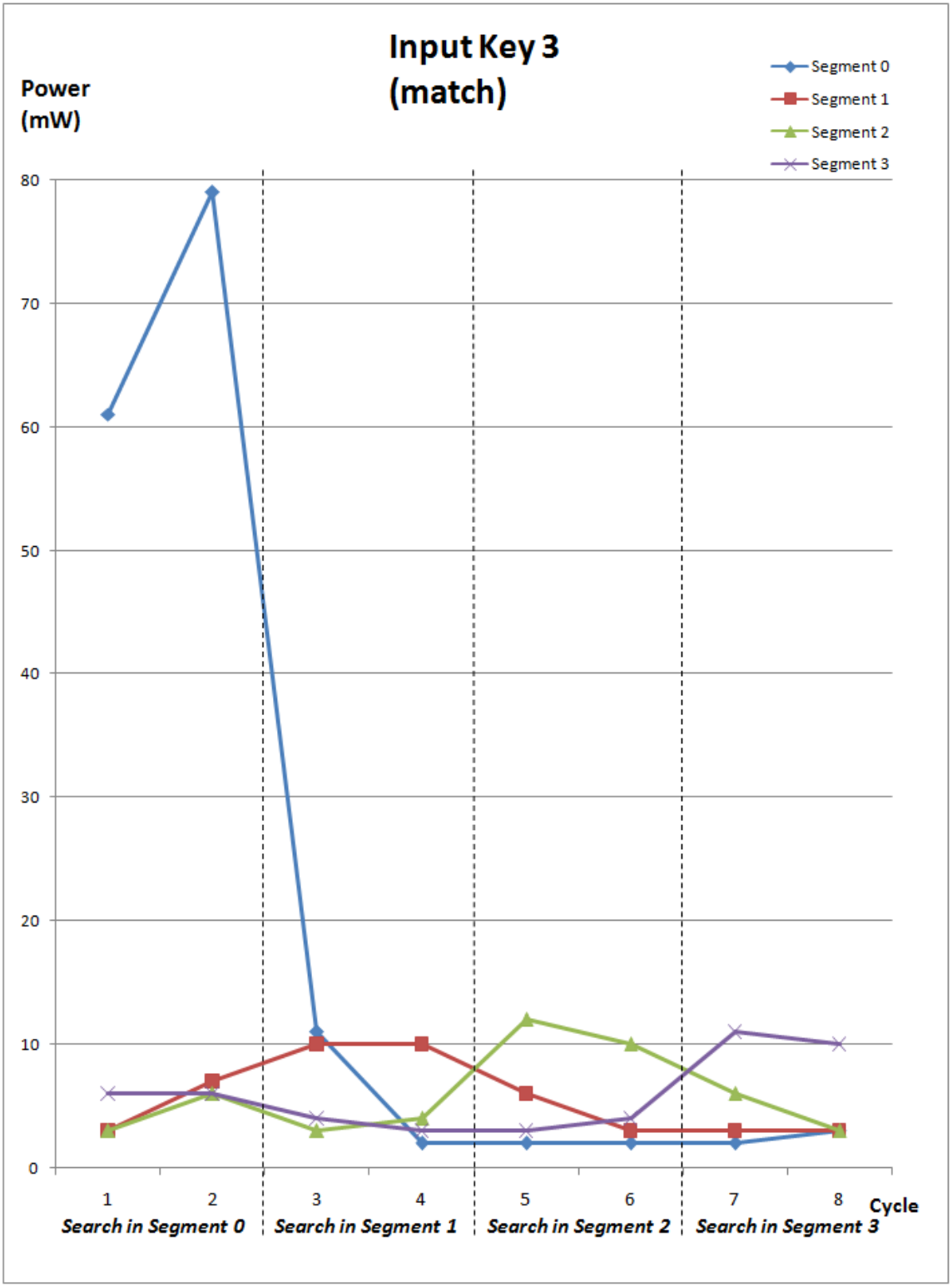


Figure 4.8 Power estimation in the period of input key 3 in pipelining scheme

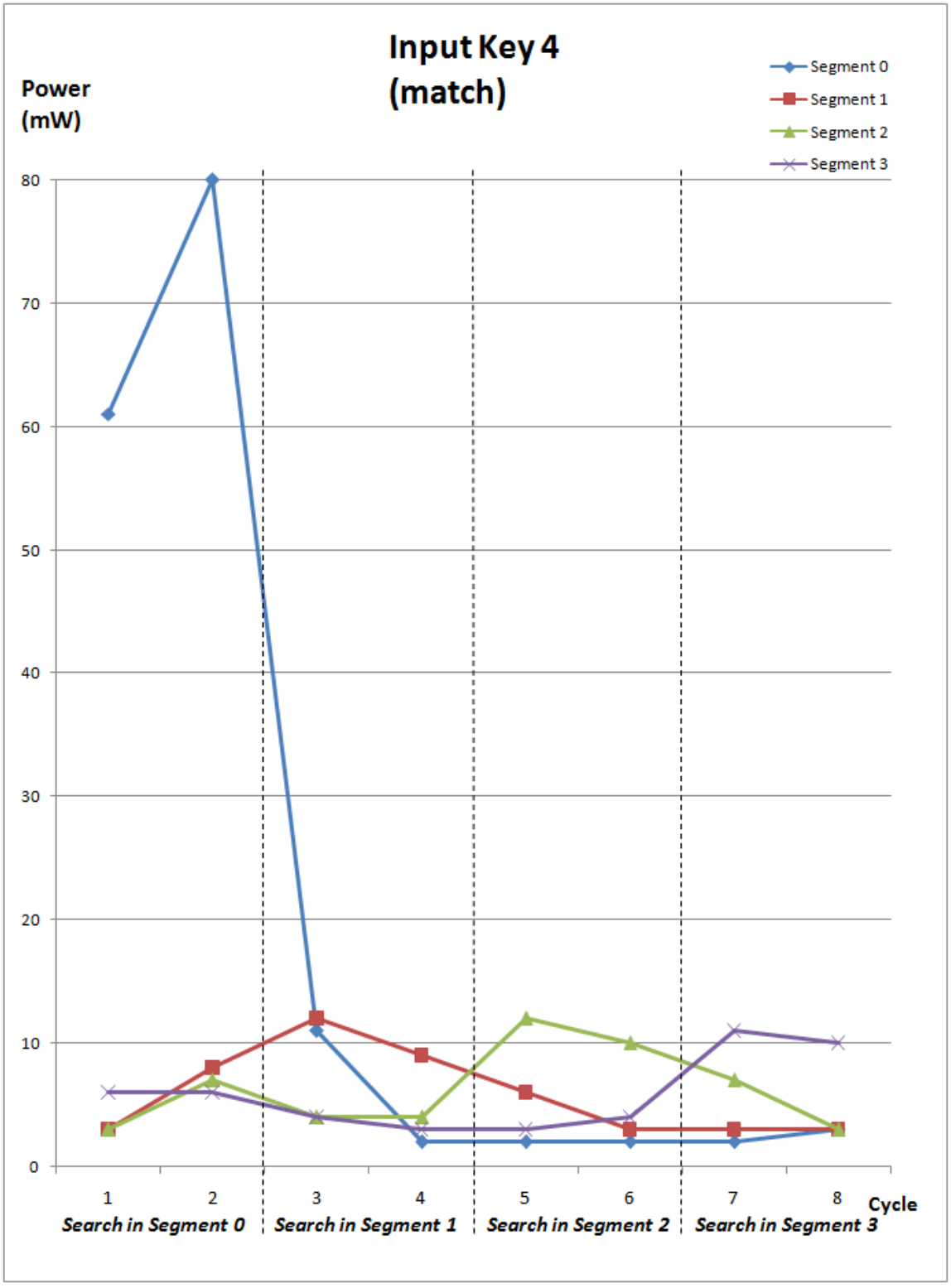


Figure 4.9 Power estimation in the period of input key 3 in pipelining scheme

4.4 Power Estimation in Precomputation Scheme

The implementation of precomputation is combined with the pipelining scheme. It takes one more cycle for precomputation process before every search operation. Therefore the whole search operation in same test scenario lasts 45 cycles and 9 cycles for each input key. From the estimation result in Figure 4.10, the average power consumption of RAM blocks is further reduced from 16.8 mW to 5.24 mW. The average total power decreases 36% comparing to the pipelining scheme. The decline of power consumption results from reduced active RAM blocks in the first segment due to precomputation process.

Comparing the power consumption distribution between the pipelining scheme and the precomputation scheme, the power estimation through segments shown from Figure 4.11 to Figure 4.15 further illustrates how power consumption is reduced. In the pipelining scheme, all RAM blocks in the first segment are active and accessed for data fetch. The precomputation process filters out mismatched CAM words by extracted information at the beginning, so only RAM blocks with matched CAM words are enabled and the active resources are further reduced in the first segment and it saves extra power. For example, in Figure 4.11, the power consumption in the first segment is down to 4 mW from 61 mW, because key 0 doesn't match any CAM word in precomputation process. Thus none RAM block is enabled and accessed while all RAM blocks must keep active for data fetch in the pipelining scheme. From Figure 4.11 to Figure 4.15, the power consumption in the first segment is also reduced comparing to the pipelining scheme due to the precomputation process.

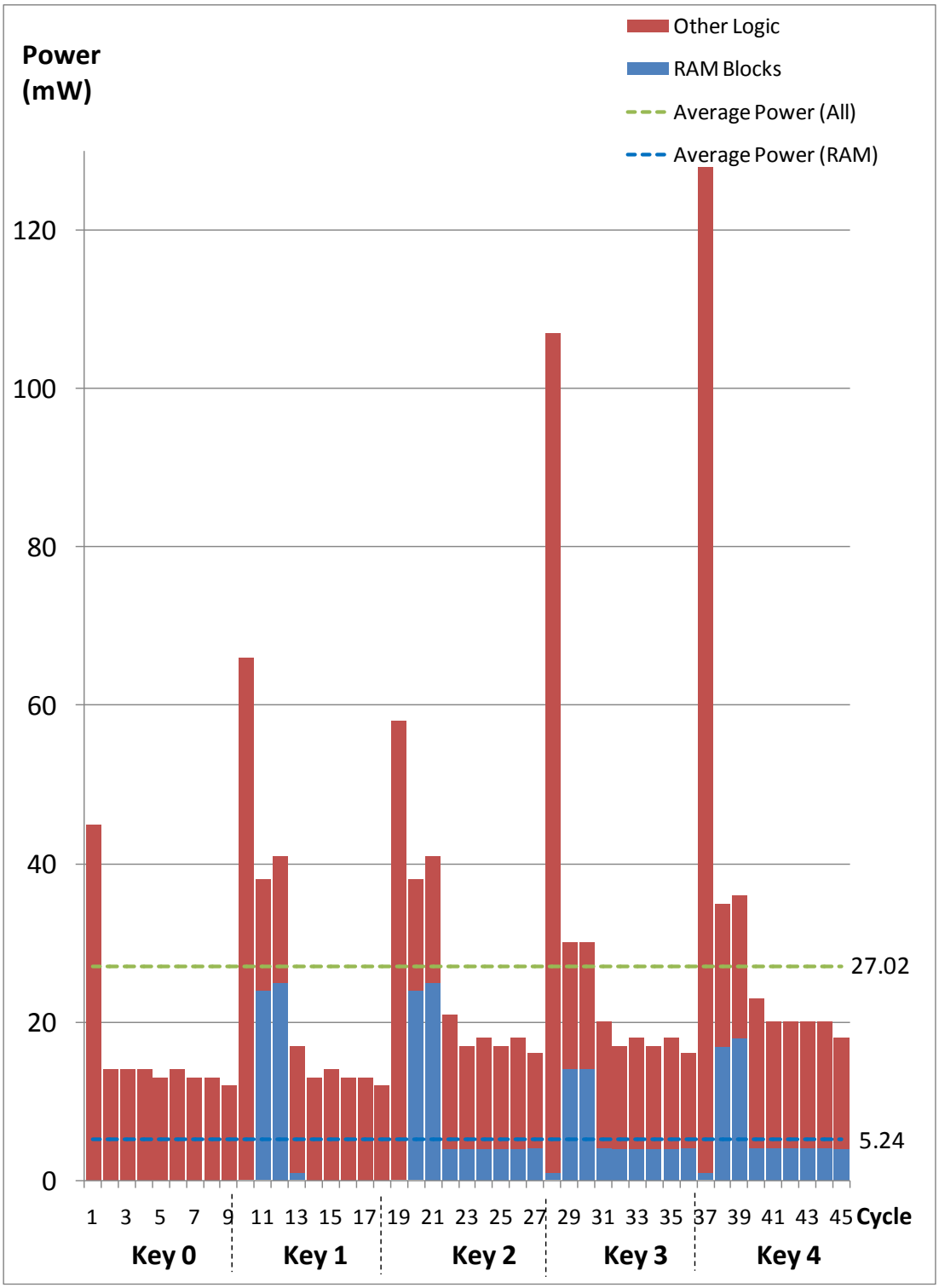


Figure 4.10 Power estimation through cycles in precomputation scheme

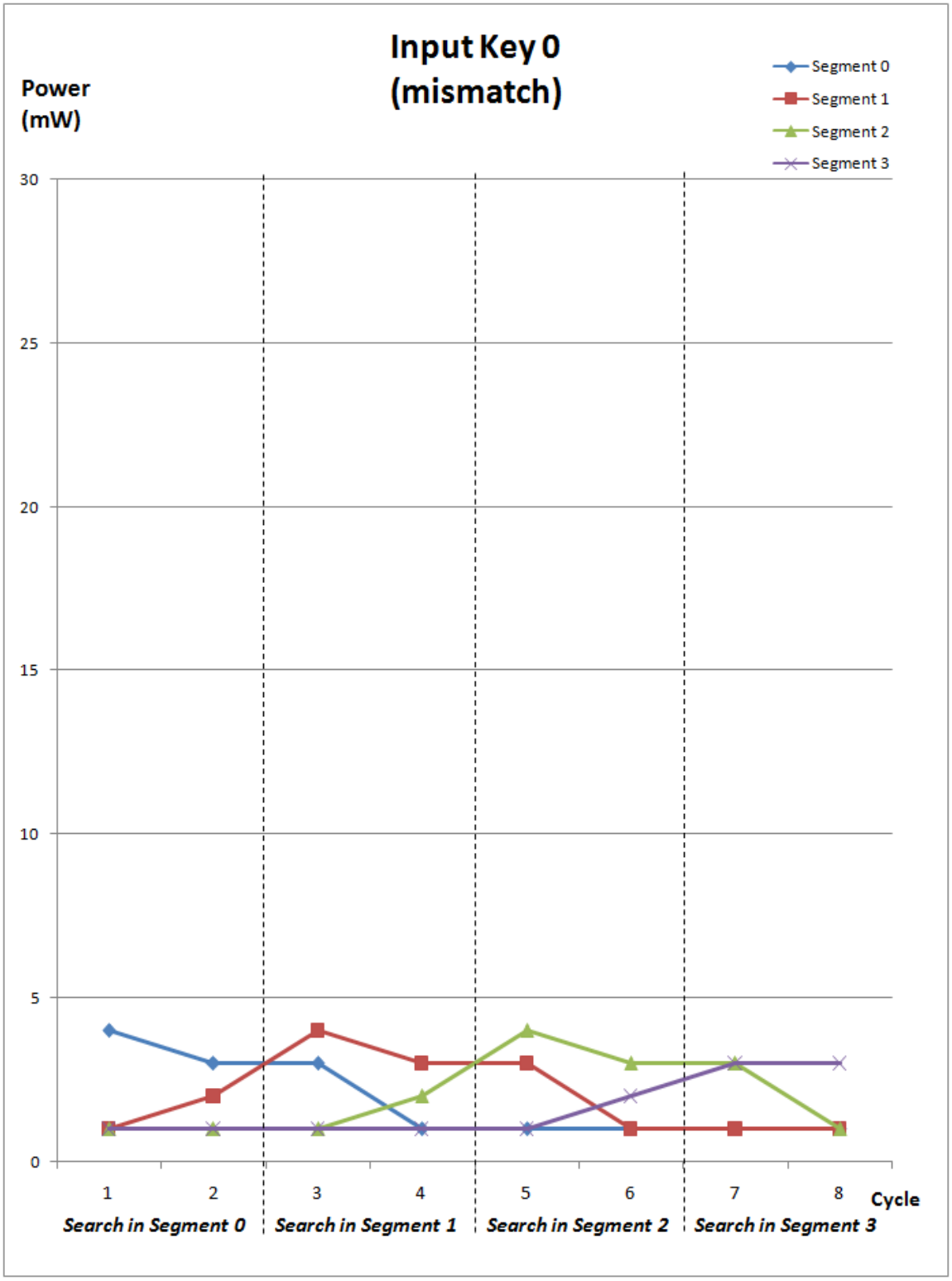


Figure 4.11 Power estimation in the period of input key 0 in precomputation scheme

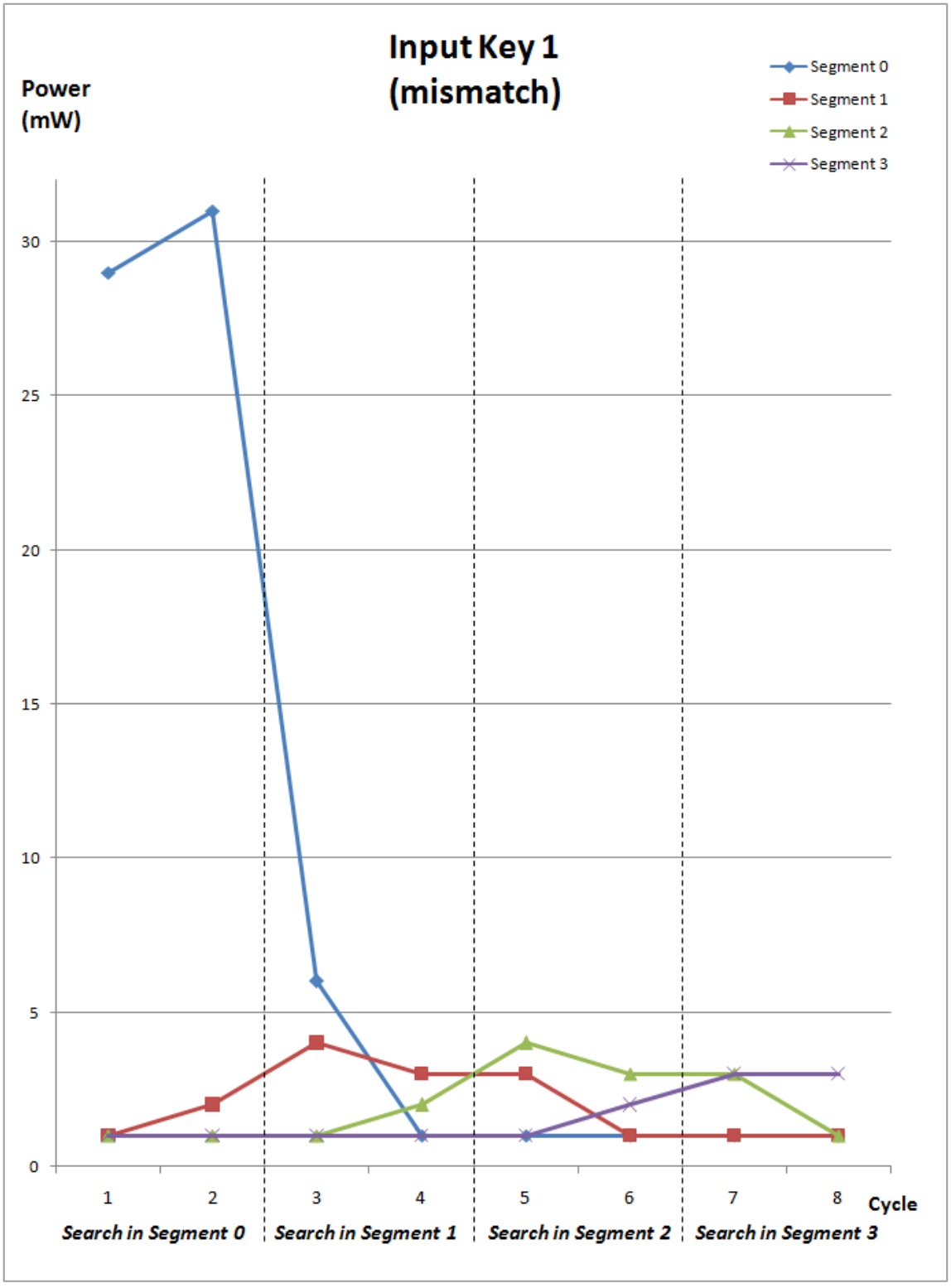


Figure 4.12 Power estimation in the period of input key 1 in precomputation scheme

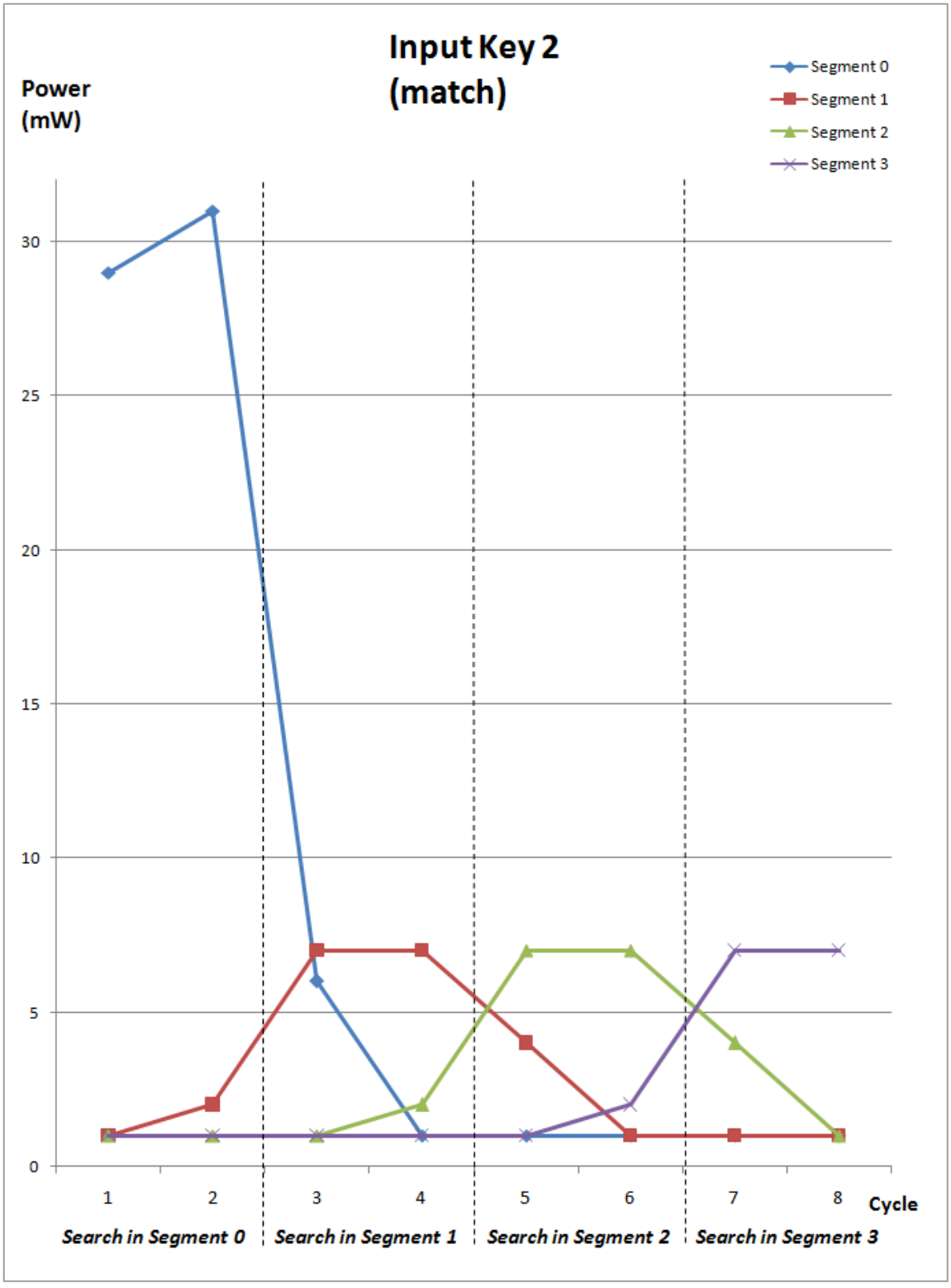


Figure 4.13 Power estimation in the period of input key 2 in precomputation scheme

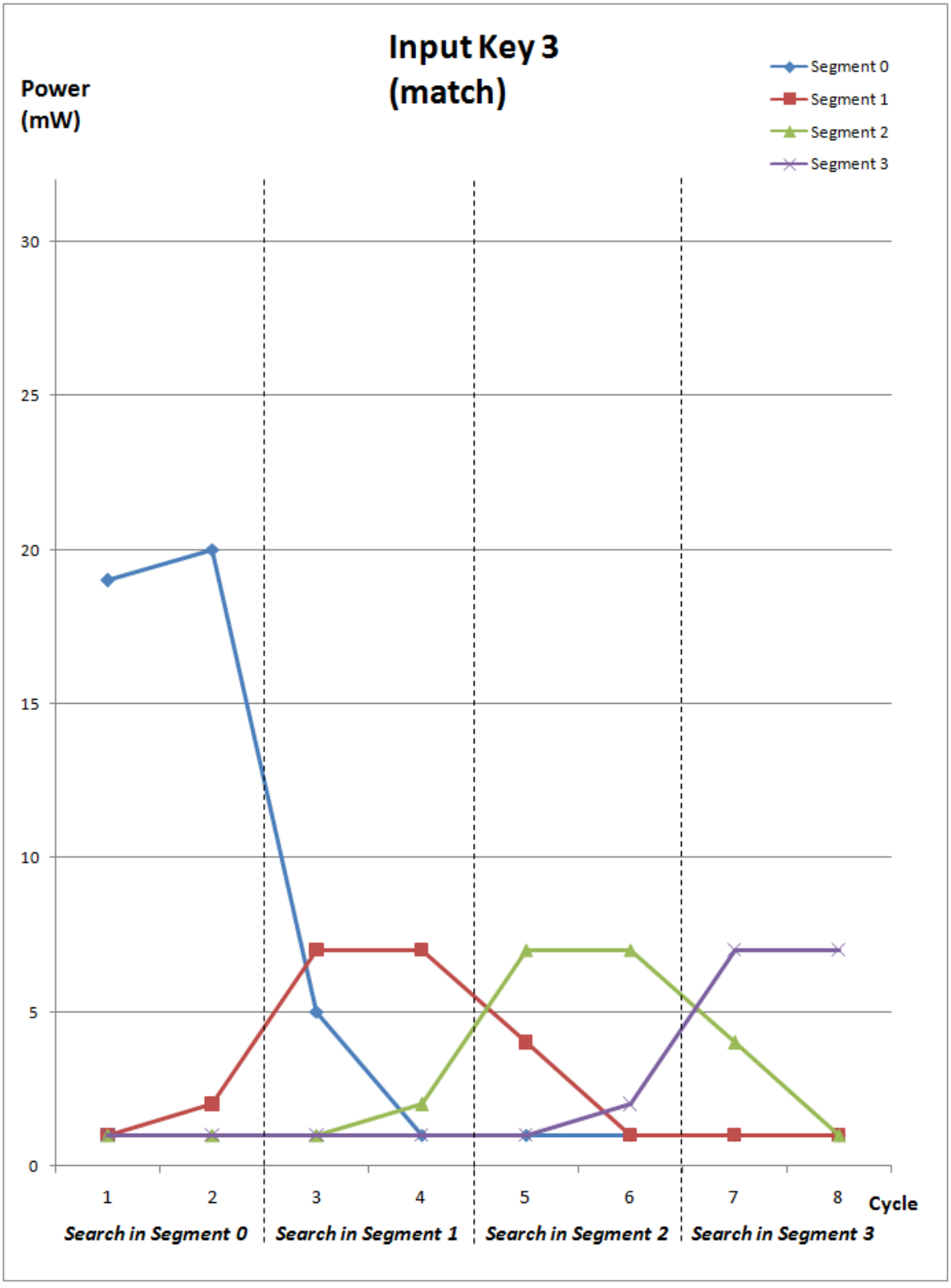


Figure 4.14 Power estimation in the period of input key 3 in precomputation scheme

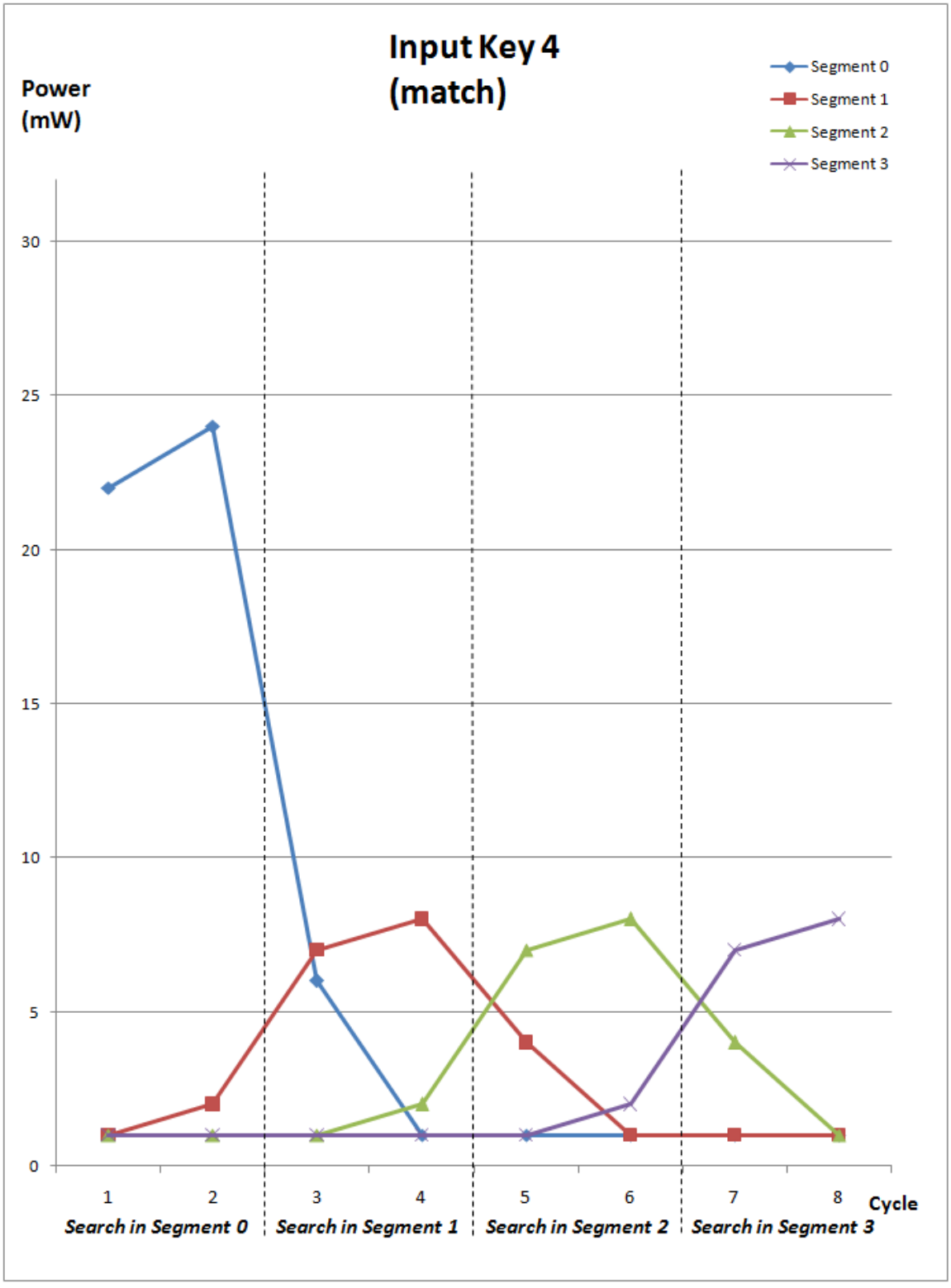


Figure 4.15 Power estimation in the period of input key 4 in precomputation scheme

4.5 Summary

The average power consumption comparison between schemes is shown in Figure 4.16. There is a huge reduction in the pipelining scheme which decreases average power consumption by 86% comparing to the conventional scheme. The precomputation scheme based on the pipelining scheme further reduces it by 36%. Overall the power optimization by low power schemes cuts 91% of average power consumption of conventional scheme. From Figure 4.16, the sharp decline in power consumption mostly comes from reduced RAM power consumption.

Figure 4.17 compares the active resource between schemes and it further illustrates how power consumption is optimized. For the conventional scheme, all RAM blocks keep active for data fetch and the search operation on mismatched CAM words wastes most of the power consumption. In the pipelining scheme, search operations abort if a mismatch occurs in previous segments and it saves power. However, all RAM blocks in the first segment must keep active for the first search operation. The precomputation scheme filters out mismatched CAM words by extracted information at the beginning and power consumption in the first segment is further optimized and Table 4.3 shows the comparison of the power consumption in the first segment between the precomputation scheme and the pipelining scheme.

The static power consumption of above three schemes is around 94 mW. For the conventional scheme, the dynamic power consumption takes up the majority of the total power consumption. For the pipelining scheme, the dynamic power consumption is

greatly reduced and the static power consumes most of the total power. Even though the dynamic power consumption is further reduced by 36% in precomputation scheme, the total power consumption is decreased by merely 11% while the search latency is increased by 12.5%. This tradeoff has to be taken into consideration between power and performance.

For the pipelining scheme and the precomputation scheme, power consumption varies depending on the value of input keys and CAM words. They are unable to filter out all mismatched data completely at an early stage. Figure 4.18 shows an example in the precomputation scheme. The extracted information of input key 4 is 32 (number of ones count) and there are six CAM words share same extracted information. Therefore the five RAM blocks where the six matched CAM words are located in segment 0 are all enabled. Although there is only one CAM word matching input key finally, eleven out of sixteen RAM blocks are disabled which still saves extra power. For some extreme cases, like very similar values (minor difference in LSB) stored in CAM, mismatched CAM words cannot be filtered out and keep active until the end of the whole operation. However taking the typical application of CAM into consideration such as the routing table in routers, the value of CAM words stands for address range which has a big width and rare similarity. In most of the cases, the precomputation based on the pipelining scheme is able to filter out mismatched values and save power efficiently.

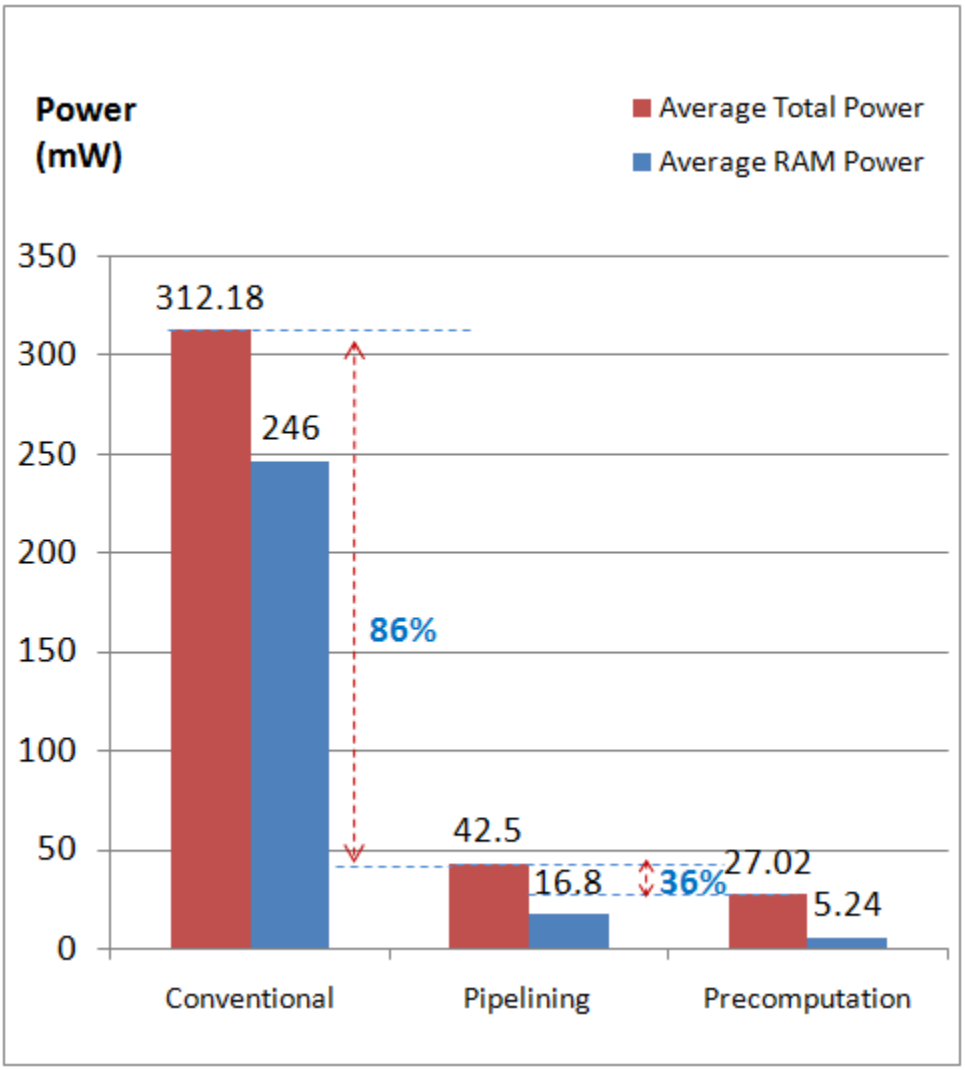


Figure 4.16 Average total dynamic power comparison between schemes

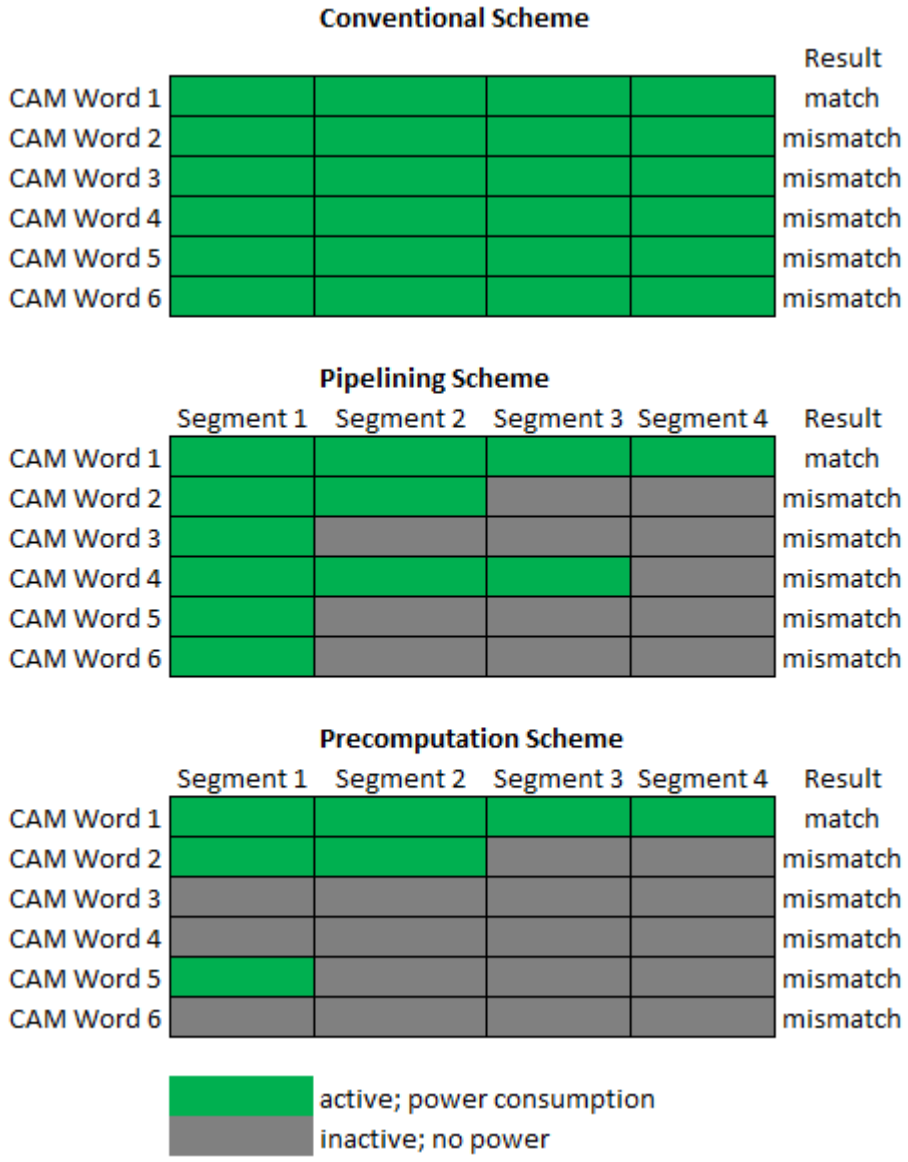


Figure 4.17 Active resource comparison between schemes

Table 4.3 Comparison of power consumption in the first segment between pipelining and precomputation

First Segment	Number of active RAM blocks		Power consumption of RAM		Power decrease
	Pipelining	Precomputation	Pipelining	Precomputation	
Key 0	15	0	58.5 mW	0 mW	100%
Key 1	15	7	59.5 mW	24.5 mW	58.82%
Key 2	15	7	59 mW	24.5 mW	58.47%
Key 3	15	4	61 mW	14 mW	77.05%
Key 4	15	5	61 mW	17.5 mW	71.31%

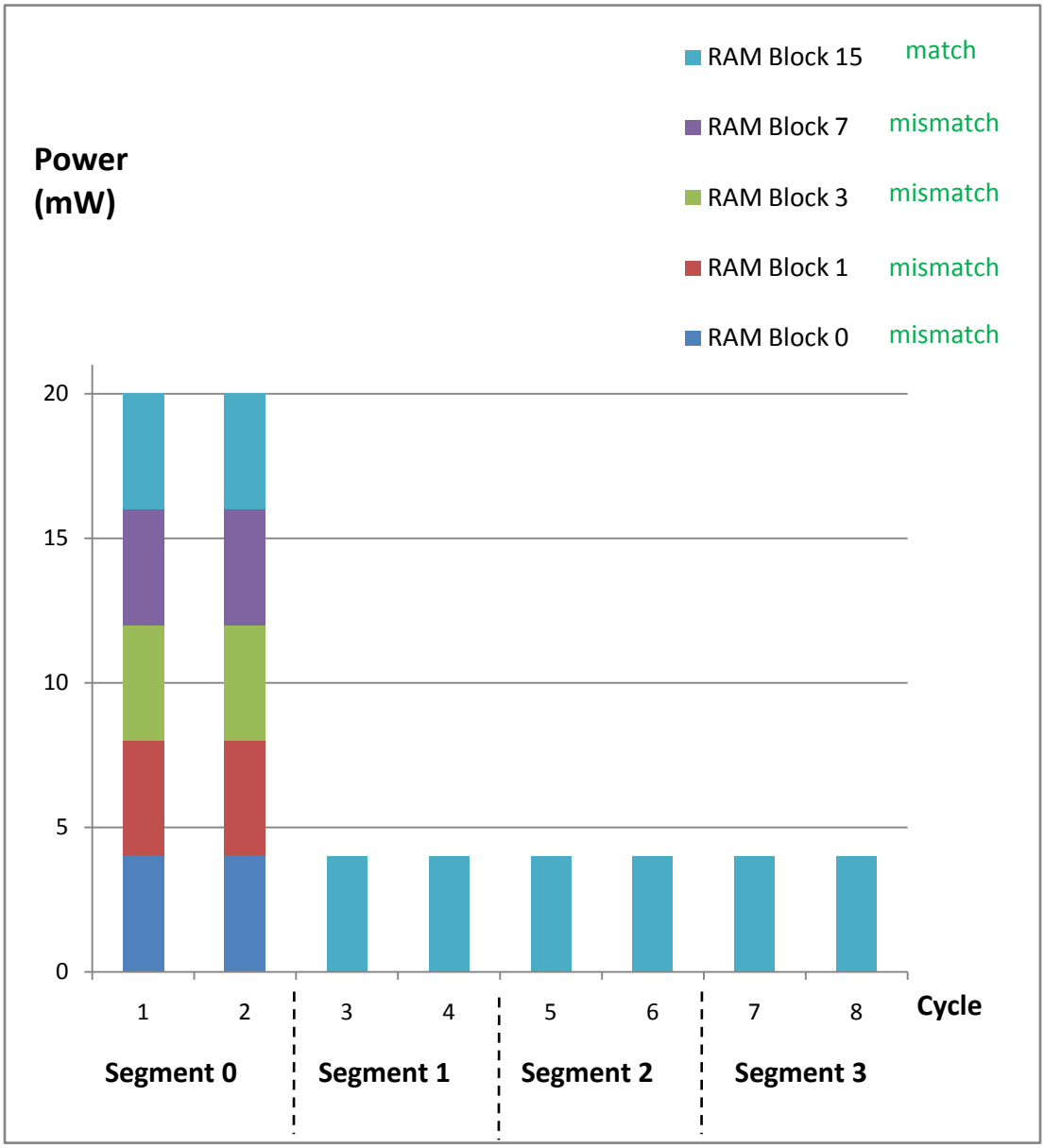


Figure 4.18 Power analysis in the period of input key 4 in precomputation scheme

Chapter 5

Conclusion

This report focused on power optimization to CAM based on FPGA. It reviewed and implemented two low power schemes pipelining and precomputation. Using Xilinx Vivado, a thorough power analysis based on cycle and resource has been expanded and discussed.

For RAM-based CAM, BRAM as a type of hardware resource embedded in Xilinx FPGA is utilized and configured as memory array of CAM values, because it is pre-designed and optimized in terms of power and area. BRAM provides a high volume of storage up to 36K bits and greatly reduces the complexity of interconnection in CAM implementation. Besides, other function logic, such as control and comparison, is comparatively simple and small in size. Therefore most of the power is dissipated on RAM instead of interconnection in RAM-based CAM.

The conventional scheme consumes a huge amount of power because all RAM blocks keep active for the data fetching during the whole period of the search operation. Hence RAM blocks consume most of the power, and the consumption is independent of the value of the input. To optimize power consumption, a pipelining scheme is proposed and shown to save much power by dividing RAM into segments as a pipeline. Mismatched RAM blocks in latter segments are disabled and thus they remain inactive. However, all

RAM blocks must keep active for the search operation in the first segment, and this consumes most of the power in the pipelining scheme. The precomputation scheme aims to further narrow the scope of the active resources by extracting information from CAM words and input keys. It filters out mismatched RAM blocks in the first segment resulting in extra power saving.

Under the same test case, the average power consumption of CAM in the pipelining scheme was reduced by 86% compared to the conventional scheme. The precomputation scheme based on the pipelining scheme further optimized the power consumption. It decreased by 91% of average power consumption finally.

Regarding resource utilization, all schemes have almost same utilization ratio. There was a tiny rise in the resource of LUT and FF from low power schemes due to duplicated function logic in segments and extra precomputation logic. The utilization rate of LUT and FF increased by 1.81% and 0.6% respectively, and the overall utilization rate of LUT and FF is still under 5% and 1% respectively which takes a very small portion of entire chip resource. Therefore the low power schemes of the pipelining and the precomputation efficiently reduced power consumption of CAM without sacrificing area.

In future work, we will focus on improving the utilization ratio. BRAM was configured in the simple dual-port mode, which provides one read-only port and one write-only port in the current design. It can be configured in the true dual-port mode to provide double read-only ports and write-only ports. Hence the number of BRAMs will be reduced by half while capacity is preserved. Both area and power will be further

optimized. Another enhancement expected to develop is ternary CAM (TCAM), which has been a growing interest in CAM application. In the current design, the value of each CAM bit is either 1 or 0. In TCAM, there is one more value of “x” which matches both 1 and 0. For TCAM implementation, each bit of CAM value (i.e. 1,0 or x) needs two bits data storage. Thus the width of BRAMs will be double and comparison circuits require modification for the value of “x”. At the architectural level, the low power schemes reviewed in this report still apply to RAM-based TCAM.

Bibliography

[1] K. Pagiamtzis, A. Sheikholeslami, "Content-Addressable Memory (cam) Circuits and Architectures: A Tutorial and Survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712-727, Mar. 2006.

[2] K. McLaughlin, N. O'Connor, and S. Sezer. "Exploring CAM Design For Network Processing Using FPGA Technology". In *Advanced International Conference on Telecommunications AICT*, pages 100–110, New York, NY, USA, 2009. ACM.

[3] Ruzica Jevtic and Carlos Carreras "Power measurement methodology for FPGA devices", *IEEE Trans. Instrum. Meas.*, Jan 2011.

[4] K. Pagiamtzis and A. Sheikholeslami, "Pipelined match-lines and hierarchical search-lines for low-power content-addressable memories," in *Proc. IEEE Custom Integrated Circuits Conf.*, 2003, pp. 383–386.

[5] K. Pagiamtzis and A. Sheikholeslami, "A low-power content-addressable memory (CAM) using pipelined hierarchical search scheme," *IEEE J. Solid-State Circuits*, vol. 39, no. 9, pp. 1512–1519, Sep. 2004.

[6] S. J. Ruan, C. Y. Wu, and J. Y. Hsieh, "Low power design of precomputation-based content addressable memory," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 3, pp. 331–335, Mar. 2008.

[7] C. S. Lin, J. C. Chang, and B. D. Liu,"A low-power precomputation based fully parallel content-addressable memory," *IEEE J. Solid-State Circuits*, vol. 38, no. 4, pp. 622–654, Apr. 2003.

[8] Koji Gardiner. (March 18, 2008). *Comparing Power Consumption of FPGAs with Customizable Microcontrollers*. Retrieved from http://www.eejournal.com/archives/articles/20080318_atmel

[9] S. Rethinagiri, R. Ben Atitallah, S. Niar, E. Senn, and J. Dekeyser. "Hybrid system level power consumption estimation for fpga-based mpsoc". *In Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 239–246, Oct 2011.

[10] J. Anderson and F. Najm. "Power estimation techniques for FPGAs". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(10):1015–1027, October 2004.

[11] "Vivado Design Suite User Guide Power Analysis and Optimization". UG907 (v2016.4) November 30, 2016. Retrieved from https://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_4/ug907-vivado-power-analysis-optimization.pdf

[12] L. Shang, A. Kaviani, and K. Bathala, "Dynamic power consumption in the Virtex-II FPGA family," in *Proc. ACM Int. Symp. Field-Programmable Gate-Arrays*, 2002, pp. 157–164.

[13] “Artix-7 FPGA Product Brief”. Retrieved from https://www.xilinx.com/support/documentation/product-briefs/artix7-product-brief.pdf

[14] “7 Series FPGAs Data Sheet”. DS180 (v2.2) December 15, 2016. Retrieved from https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

[15] “Reducing System Power and Cost with Artix-7 FPGAs”. WP423 (v2.4) September 27, 2016. Retrieved from https://www.xilinx.com/support/documentation/white_papers/wp423-Reducing-Sys-Power-Cost-28nm.pdf

[16] C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson, and B. Hutchings, "RapidSmith: Do-It-Yourself CAD Tools for Xilinx FPGAs," *International Conference on Field Programmable Logic and Applications (FPL)*, 2011

[17] Mamoru Nakanishi, Takeshi Ogura, "Real-time CAM-based Hough transform algorithm and its performance evaluation," *Machine Vision and Applications*, August 2000, Volume 12, Issue 2, pp 59–68