ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

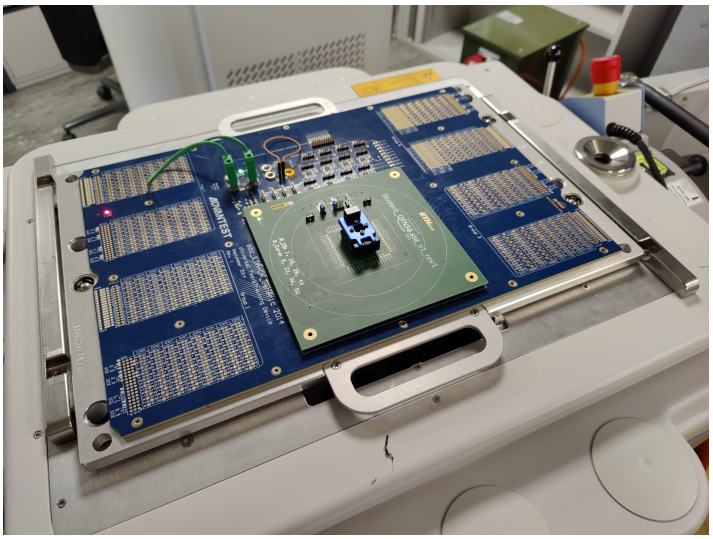
Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and
Electrical Engineering

Spring Semester 2023

Eclipse: Speed Test and Power Measurement

VLSI4 Mini Project



Jiayong Li
jiayli@ethz.ch

July 2023

Supervisors: Luca Bertaccini, lbertaccini@iis.ee.ethz.ch
Tim Fischer, fischeti@iis.ee.ethz.ch

Abstract

As a mini-project from course VLSI4, the chip named Eclipse is tested. Eclipse is a microcontroller using PULPissimo architecture and taped out in TSMC 65nm technology. The basic setup and functional tests are done in other Eclipse-related mini-projects. This report focuses on finding the maximum speed and measuring the chip's power consumption.

There is a short introduction to the chip in Chapter 1. Before performing the speed test, the FLL needs to be configured to the correct frequency. Chapter 2 shows how the FLL is configured and some measurement results to verify that the chip is in the correct frequency. After that, the speed test results are shown in Chapter 3. And Chapter 4 shows the power measurement results.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Eclipse is a microcontroller using PULPissimo architecture and taped out in TSMC 65nm technology. It features a new core-FPU interface, X-interface, a DIV/SQRT unit from T-head, and a stochastic rounding (SR) extension in the SDOTP unit. Figure 1.1 shows the architecture of Eclipse. New features are colored in red.

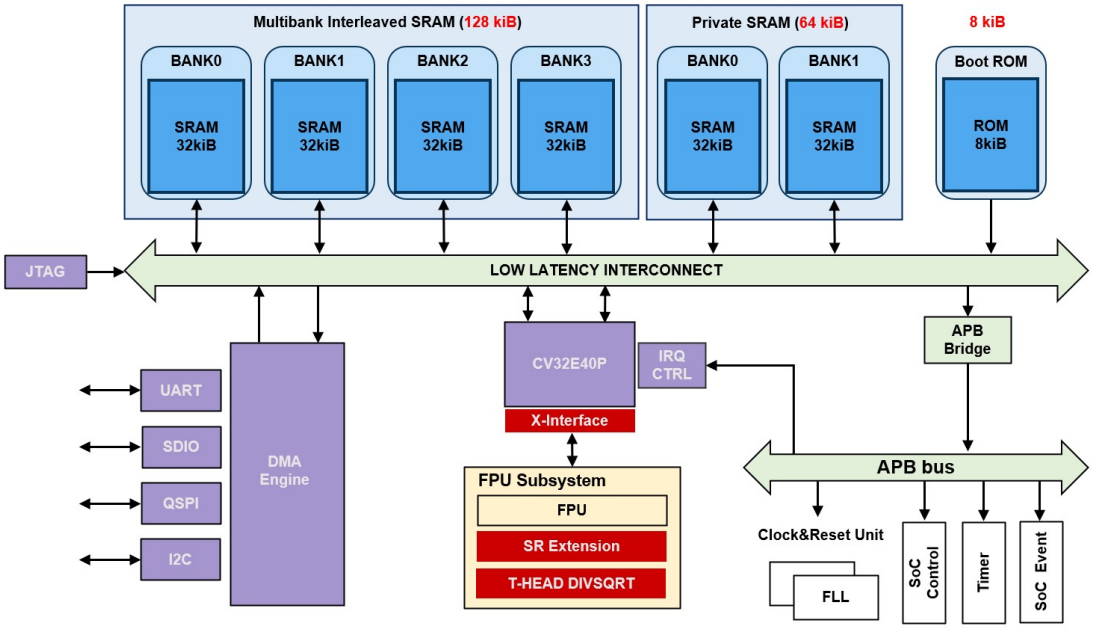


Figure 1.1: Eclipse Hardware Architecture

Eclipse uses an in-order RISCV core, CV32E40P, with 192 kiB on-chip SRAM. The whole SoC is in 1 power domain, and there are 2 clock domains: SoC and Peripherals. In the

testing, the JTAG interface is used to halt the core, load the program, configure FLL, resume the core, and check the return value.

Eclipse uses QFN56 packaging and the pinout is shown in Figure 1.2.

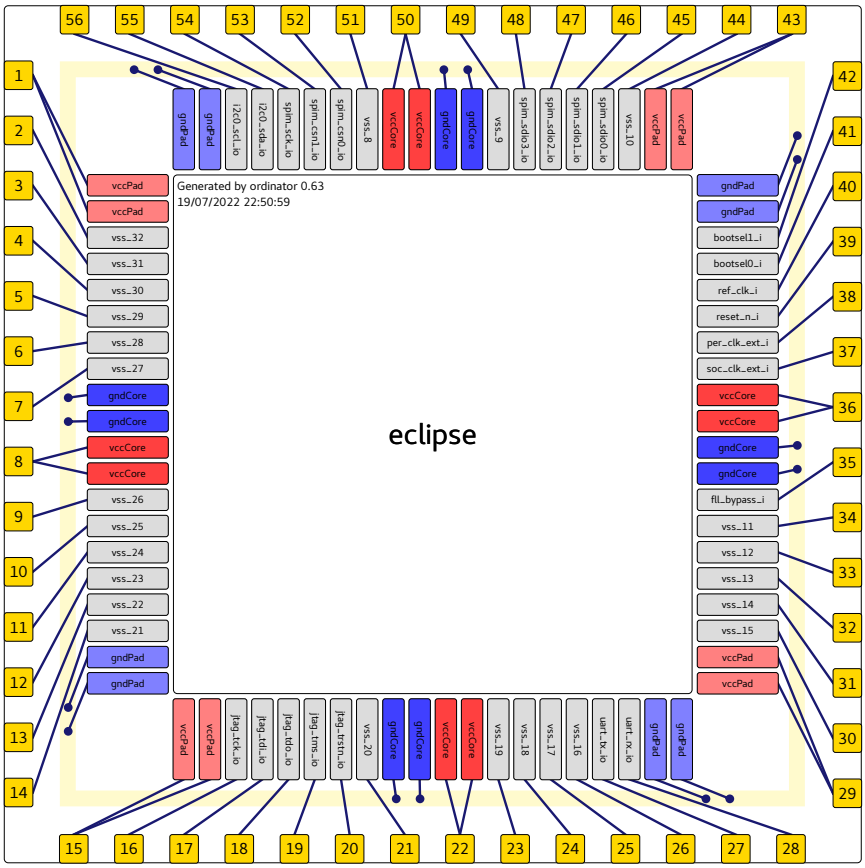


Figure 1.2: Eclipse Pinout

The basic setup and functional tests are done in other Eclipse-related mini-projects. The chip is working properly at low frequencies. This project reuses the existing setup and some functional tests to measure the max frequency and power of the chip.

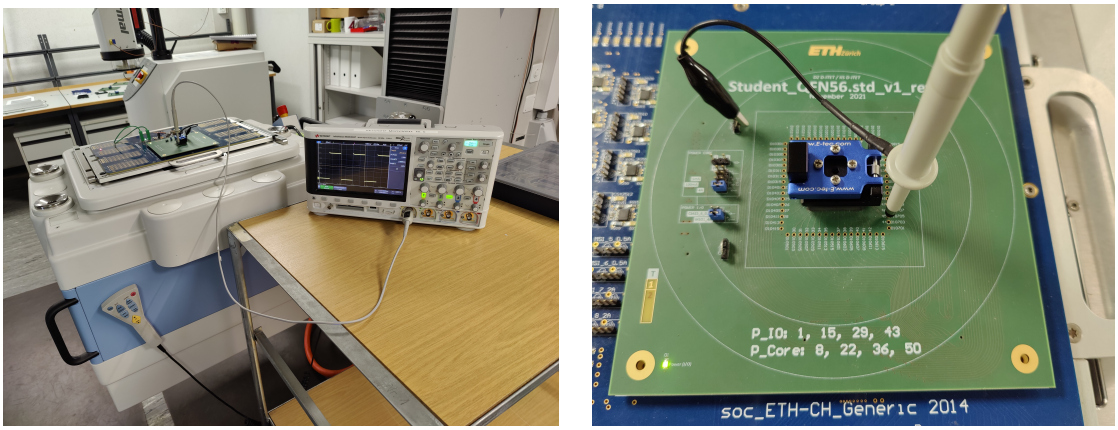
Chapter 2

Configure FLL Frequency

The first step of the whole measurement is configuring the FLL on the chip to achieve different core frequencies. This is done using the IIS test method, `vega_tml.changeFrequency`. The FLL frequency is a parameter in the test method, and we can do a frequency sweep easily in the tester software.

The remaining parts of this chapter will show how to verify the frequencies are set up correctly. All the measurements in this report are done on the chip with the number 3 written on it and the chip is operating at room temperature.

2.1 Setup of Oscilloscope



(a) Stable Waveform of GPIO Toggling

(b) Pin 45 is Measured

Figure 2.1: GPIO Toggling in Oscilloscope

To measure the core frequency, the core is programmed to do an infinite loop on INT32 MATMUL. After every 10 MATMUL, the chip will toggle the output voltage on GPIO 0(Correspond to spim_sdio0_io, pin 45). By measuring the period of the GPIO toggling through an oscilloscope, we can get the core frequency and verify the core frequencies are set to the desired values. The program is shown in pseudo-code 1 and the setup of the oscilloscope is shown in Figure 2.1.

Algorithm 1: MATMUL INT32 with GPIO toggling

```
1 GPIO_level = 1;
2 GPIO_set(GPIO_level);
  // Toggle GPIO
3 while true do
  |  // Do MATMUL x10 and toggle GPIO
4 |  for i ← 0 to 10 do
5 |  |  matmul();
6 |  end
7 |  GPIO_set(!GPIO_level);
8 |  GPIO_level = !GPIO_level;
9 end
```

2.2 Relationship Between GPIO Toggling Period and Core Clock Frequency

To link the core clock frequency to the GPIO toggling period measured by the oscilloscope, the same program is run in both RTL simulation and real chip without the FLL configuration step. In the RTL simulation(both functional and post-layout), GPIO toggling period is obtained by averaging the first 4 toggling periods in the waveform. The measurement results are shown in table 2.1.

Table 2.1: Core Clock Frequency and GPIO Toggle Period without FLL Config

Method	Core Clock[MHz]	GPIO Toggle Period[ms]
Real Chip No FLL Config	—	10.8
Functional Simulation	50.525	10.8
Post-layout (with SDF) Simulation	50.916	10.6

2.3 Change Clock Frequency by the Test Method

To verify the test method sets the FLL to the correct clock frequency, the same program is run on the real chip after FLL configuration step, and the following two things are checked:

1. Starting point: The FLL frequency after reset is 50MHz. We should obtain the same GPIO toggling period when explicitly configuring FLL to 50MHz.
2. Positive relationship: The ratio of GPIO toggling period shrinking should be the same as the ratio between the frequencies we want to set at the two points.

Table 2.2: GPIO Toggling Period and Ratio to 50MHz Case in Different Target Freq

Target Freq[MHz]	**50**	**100**	**150**	**200**	**250**	**300**	**350**
Period[ms]	11.1	5.55	3.70	2.77	2.22	1.85	1.59
Ratio	—	2.00	3.00	4.01	5.00	6.00	6.99

Results from the real chip are shown in table 2.2. Measurement results show a clear positive relationship between GPIO toggling and core frequencies. But at 50 MHz, there are 0.2∼0.3ms differences in the GPIO toggle period obtained from simulation, FLL reset, and test methods. This gives ∼2% difference in the corresponding frequency. The chip does not work(no GPIO toggling at all) at 400MHz and we only need to know the frequency to be accurate to 10 MHz. Since 400MHz $\times$ 2% $<$ 10MHz, we do not need extra calibration on this ∼2% difference.

For the tests in the following chapters, all frequency sweeps are done by passing core frequency as a parameter to the test method, `vega_tml.changeFrequency`.

Chapter 3

Speed Tests

In the previous chapter, we verified our approach can set the chip to the correct frequency. We choose several functional tests on new features, computations, and IO to see if the chip works correctly. The largest frequency at which all functional tests are passed is the max frequency at the given voltage.

3.1 Set of Functional Tests

Table 3.1 shows the functional tests we chose. Except for the GPIO Toggle, all the other tests are bundled together: first change frequency, then run each functional test and check the return value. The GPIO Toggle is run manually around the biggest frequency found in the previous step. The vector in this test is the same as we measure the FLL frequency: toggle GPIO every 10 MATMUL. The test is passed when a stable toggling waveform is observed in the oscilloscope.

Table 3.1: The Set of Functional Tests Chosen to Check the Chip is Working Correctly

Target	**Test**	**Check**
New Features	SDOTP with SR	return 16 + error
	DIV/SQRT Unit	return 32 + error
Computation	SDOTP with Diff FP Format	return 64 + error
	MATMUL FP32	return 128 + error
	MATMUL INT32	return 160 + error
IO	GPIO Toggle	stable waveform in oscilloscope

The return values of different tests are set to different values plus the number of errors detected in the program. This is to avoid the return value of the previous program remaining in the register and affect our checks. This lets us know that the corresponding test is run and returned.

3.2 Result

Table 3.2 shows the max frequency measured on the chip(Eclipse chip with number 3 on it). The largest frequency for GPIO test and all other tests are shown in rows 2 and 3 separately.

Table 3.2: Max Frequency at Different Core Voltage

Core VDD[V]	0.90	0.95	1.00	1.05	1.10	1.15	1.20
Max Freq[MHz]	180	220	250	280	310	340	370
GPIO Pass[MHz]	180	220	250	280	310	340	370
All Other Pass[MHz]	180	220	260	290	330	350	380

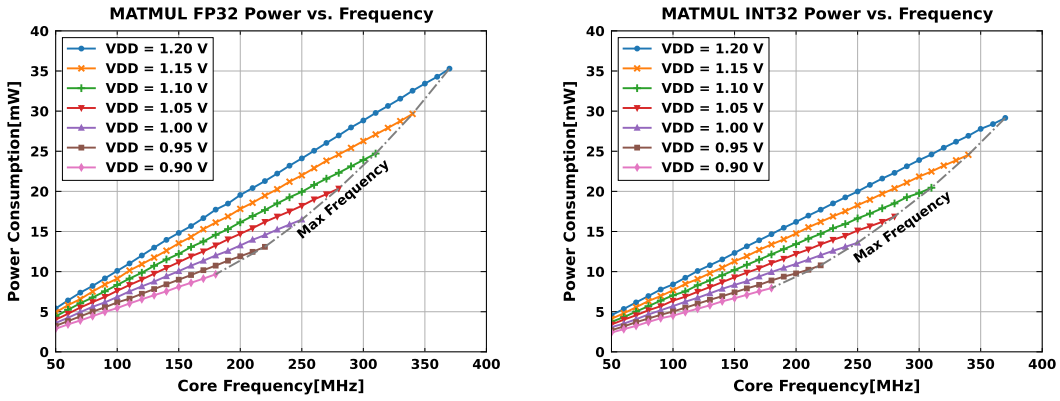
In the measurement, GPIO toggling waveform in the oscilloscope still exists when the frequency is 10 or 20 MHz higher than the frequency shown in the table, but those waveforms are unstable. The toggling will disappear in the oscilloscope and stay constantly low or high after 10 to 20 seconds.

Chapter 4

Power Tests

After knowing the max frequency, a power test is performed. We measured the operating power of the chip in different tasks at different frequencies and core VDD. The chip is first configured to perform an infinite loop on a specific task, and then we measure the operating current by the `dc_tml.DcTest.StandbyCurrent` test method. This test method will measure the current flows to the chip without loading any pattern and the power is obtained by multiplying this current with the core voltage.

4.1 Power vs. Frequency



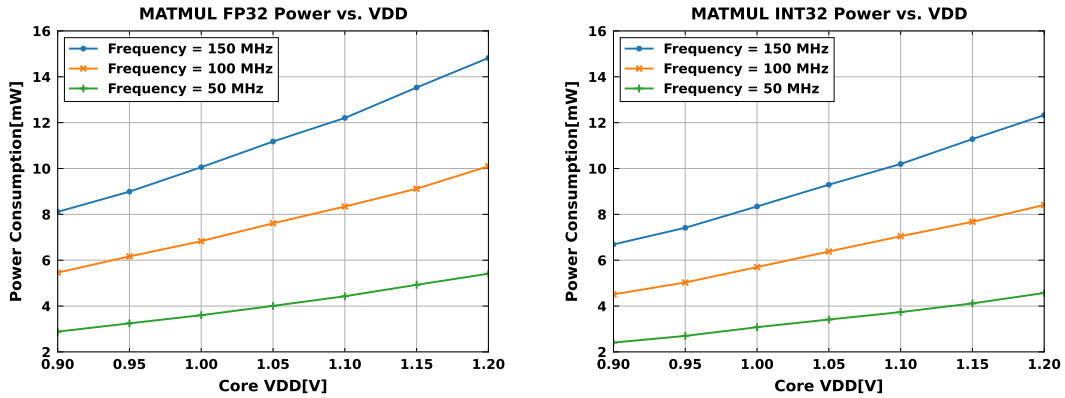
(a) MATMUL FP32

(b) MATMUL INT32

Figure 4.1: Eclipse MATMUL, Power vs. Frequency

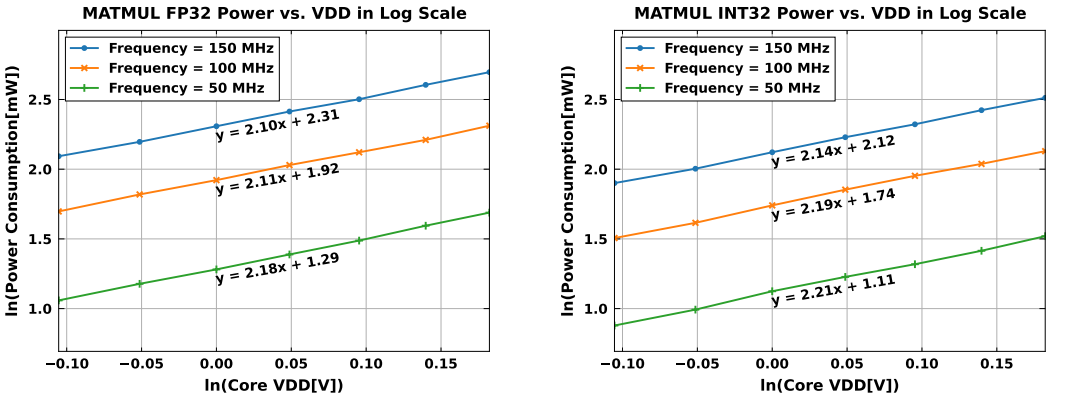
Figure 4.1 shows the relationship between power consumption and core frequency at different core VDD. The chip is asked to run matrix-matrix multiplication on FP32 and INT32 respectively. For each voltage level, the frequency starts at 50MHz and ends at the maximum frequency the chip can achieve at that voltage level. The max frequencies are labeled by dash lines in the plots. The linear relationship between power and frequency can be observed in the plots.

4.2 Power vs. VDD



(a) MATMUL FP32 (b) MATMUL INT32

Figure 4.2: Eclipse MATMUL, Power vs. VDD



(a) MATMUL FP32 (b) MATMUL INT32

Figure 4.3: Eclipse MATMUL, Power vs. VDD in Log Scale

The relationship between power consumption and voltage is shown in Figure 4.2. Voltage

is changed from 0.9V to 1.2V. The same data is plotted in a log scale to check the linear relationship between power and the voltage square. Figure 4.3 shows the linear relationship between power and voltage in log scale and the coefficient is approximately 2.

4.3 Power Consumption at Max Frequencies vs. VDD

Figure 4.4 and Table 4.1 summarize the speed and power test results. The max frequencies at different voltage level is shown in blue line and the power consumption at the max frequencies in different tasks and voltage levels are shown in red lines.

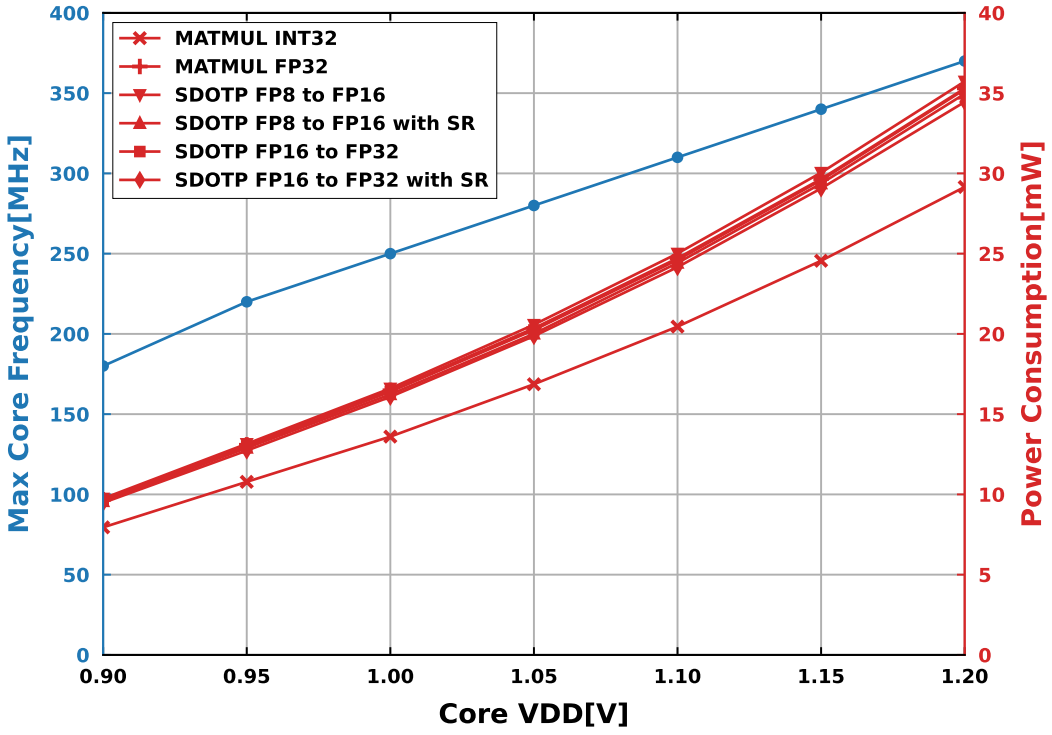


Figure 4.4: Voltage Sweep vs. Max Frequency vs. Power Consumption

There is a clear difference between floating-point operations and integer operations. Floating-point operations consume more power than integer operations. But the differences between different floating-point programs are not too much.

This does not mean there are no differences in different FPU instructions. The programs executed on the chip also contain data loading and branches for looping, FPU is also

Table 4.1: Eclipse Test Summary

Technology	TSMC 65nm
Chip Area	$4mm^2$
VDD Range	0.9 - 1.2V
Frequency Range	180 - 370MHz
Max FP Power	35.73mW
Max INT Power	29.16mW

not 100% utilized, so the differences between FPU instructions may not appear in the measured results. To measure the difference between FPU instructions, some further tests are needed.

Chapter 5

Conclusion

This project measures speed and power consumption for the Eclipse chip. First, GPIO toggling is used to characterize the core frequency and we verify that the chip is working at the correct frequency based on the measurements.

After that, several functional tests on different units are performed and the max frequencies at each voltage level are measured. Then, the chip's power consumption is obtained by measuring the current flows to the chip in an infinite loop of a specific task. We also checked the relationship between power, frequency, and voltage are as expected.

The summary of all measurements is shown in Figure 4.4 and Table 4.1. The chip works at 370MHz in a 1.2V power supply. The max power consumption of the floating-point computation task is 35.73mW and the integer computation task is 29.16mW.