3DGates: An Instruction-Level Energy Analysis and Optimization of 3D Printers

Jerry Ajay Chen Song Aditya Singh Rathore Chi Zhou Wenyao Xu*

University at Buffalo, the State University of New York

{jerryant,csong5,asrathor,chizhou,wenyaoxu}@buffalo.edu

Abstract

As the next-generation manufacturing driven force, 3D printing technology is having a transformative effect on various industrial domains and has been widely applied in a broad spectrum of applications. It also progresses towards other versatile fields with portable battery-powered 3D printers working on a limited energy budget. While reducing manufacturing energy is an essential challenge in industrial sustainability and national economics, this growing trend motivates us to explore the energy consumption of the 3D printer for the purpose of energy efficiency. To this end, we perform an in-depth analysis of energy consumption in commercial, off-the-shelf 3D printers from an instruction-level perspective. We build an instruction-level energy model and an energy profiler to analyze the energy cost during the fabrication process. From the insights obtained by the energy profiler, we propose and implement a cross-layer energy optimization solution, called *3DGates*, which spans the instruction-set, the compiler and the firmware. We evaluate *3DGates* over 338 benchmarks on a 3D printer and achieve an overall energy reduction of 25%.

CCS Concepts • **Computer systems organization** → **Embedded and cyber-physical systems**; *Special purpose systems*; Sensors and actuators; Firmware; • **General and reference** → *Cross-computing tools and techniques*

Keywords 3D Printers; Energy Characterization and Optimization; G-code Instruction rofiling

1. Introduction

3D printing, also known as additive manufacturing, is a revolutionary manufacturing technology which allows complex objects to be created in a single piece, layer by layer, bypassing traditional steps of design and production. Due to its elegant concept, 3D printing has become the next-generation manufacturing driving force, bringing a transformative effect across a broad spectrum of industries, including automotive, aerospace, retail and biomedicine. Also, it can be even more versatile as this technology progresses towards being lightweight and low cost for individual use. The global market of 3D-printing-related industries is estimated to reach 20.2 billion by 2021 [64].

Along with rapid market growth, the global energy demand also keeps increasing. According to the report by International Energy Outlook (IEO) in 2016, the manufacturing industry consumes about 54% of the world's total delivered energy [3]. While reducing the energy use in manufacturing is a core problem in industrial sustainability and national economics, we raise a critical concern: *How to optimize the energy consumption of 3D printers?* With the increasing demands towards portable battery-powered 3D printing [4, 5, 2], this challenge becomes more critical.

There is, however, limited work on characterizing and optimizing the energy consumption of 3D printers. Walls *et al.* [61] compared the power consumption among a few low-cost 3D printers. Peng *et al.* [47] quantified the energy consumption of 3D printers by only considering the heating process, leaving other aspects under-explored. Recently, Ajay *et al.* [8] discovered that the energy consumption can account for 32% of the overall 3D printing cost, which emphasizes the urgent demand of an energy optimization solution for 3D printers.

Motivated by the above, we carry out an in-depth instruction-level analysis of the energy consumption in the 3D printing process. There are two main reasons that we choose an instruction-level approach. First, it is independent of lower-level hardware specifications - an important factor when considering the hardware diversity among different 3D printers. Second, it is useful in assigning an accurate power cost to the higher-level system software that generates and schedules these instructions - an important factor when considering reduced generation of power-hungry instruction [55].

* Address comments to wenyaoxu@buffalo.edu or +1 (716)-645-4748.

ASPLOS'17 April 08–12, 2017, Xi'an, China.

ISBN 978-1-4503-4465-4/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3037697.3037752

This work attempts to address two fundamental questions regarding energy-efficiency of 3D printers: (1) *How much energy does one printing instruction consume?* and (2) *How to optimize the energy consumption of an instruction?* To this end, we first investigate the energy consumption of the most commonly used 3D printing instructions. Second, we develop an instruction-level energy model and an energy profiler to accurately simulate the energy usage of a particular printing process. Third, from the insights obtained by the energy profiler, we propose *3DGates*, a cross-layer solution which spans the instruction set, the firmware and the compiler to reduce the energy consumption of the printing process. Additionally, *3DGates* takes into consideration the unique properties of mechatronic cyber-physical systems, such as *instruction-inertia* and *instruction-delay* (see Section 4.4), to ensure the operation correctness. Our intensive experimental evaluation shows that *3DGates* can reduce 25% of the total energy consumption in the 3D printing process. More importantly, it ensures no defects or structural compromises to the product quality and no changes in the printing duration.

In summary, this work has three contributions as follows:

- **Instruction-level energy model and energy characterization**: We build an instruction-level energy model and thereby understand the energy behavior of the typical 3D printing instructions. Based on this model, we develop an energy profiler which simulates the energy consumption for a printing task.
- **Cross-layer energy optimizations**: We propose and implement *3DGates*, a cross-layer solution spanning the instruction-set, the compiler, and the firmware. *3DGates* is an immediately deployable solution that can be applied on commercially off-the-shelf 3D printers without inducing any alteration to the hardware.
- **Real-world evaluations**: We simulate *3DGates* on 338 benchmarks and observe an average reduction in energy consumption by 25%. We implement *3DGates* on real 3D printers (e.g., Ultimaker 2 Go [56]) and evaluate the performance of energy reduction over real-world 3D designs.

The remaining of this paper is organized as follows. In Section 2, we introduce the background of the 3D printing process and its associated instruction-set. In Section 3, we investigate the instruction-level energy model, develop an instruction-level energy profiler, and obtain the insights of the energy consumption of real 3D printers. In Section 4, we elucidate the implementation of *3DGates* for reducing the energy consumption. In Section 5, we evaluate *3DGates* against 338 benchmarks on a software simulator and validate the simulator's results on a real 3D printer. We discuss potential enhancements and other related works in Section 6 and Section 7, respectively. The paper is concluded in Section 8.

2. Background

2.1 3D Printing Process

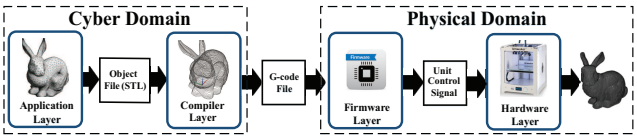


Figure 1: End-to-end flowchart of 3D printing

3D printing refers to a process where the digital design is converted to a 3D physical object. Figure 1 depicts the end-to-end flowchart between the cyber and the physical domains. Each domain involves different functional entities as noted below:

- **Application**: The design is created in the Stereolithography (STL) format by applications belonging to this layer. A computer-aided design (CAD) software qualifies as an example.
- **Compiler**: The compiler processes the STL file and generates a tool path file, called G-code file in most cases. The G-code file contains a series of instructions to direct the printing process.
- **Firmware**: The firmware on the 3D printer interprets the G-code file and generates corresponding control signals to the hardware.
- **Hardware**: The physical units of a 3D printer (e.g., the stepper motors, the heater and the cooling fans) operate according to the control signals.

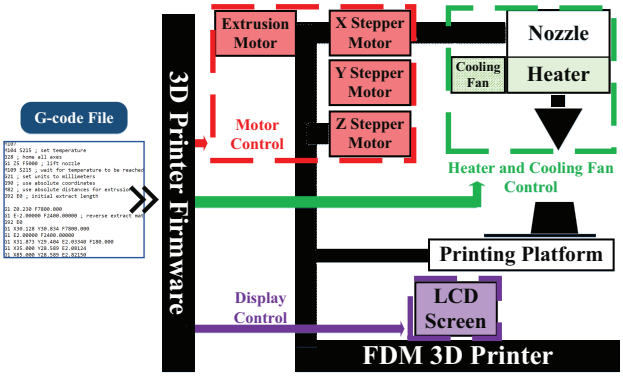


Figure 2: 3D printer hardware architecture.

2.2 3D Printer Architecture

In this study, we investigate 3D printers based on the Fused Deposition Modeling (FDM) technology because it is the most commonly used type in the commodity 3D printers [6].

Figure 2 shows the hardware architecture of an FDM printer. The X, Y and Z stepper motors control the nozzle movement. The extrusion motor (E motor) governs the extrusion of the material. The heater (on the print header) melts

the material and lays it down on the printing platform layer-by-layer. The cooling fan is employed to prevent the print header from overheating.

2.3 G-code

G-code, also called RS-274 [63], is the numerical control instruction set employed to control the 3D printing process. Although the G-code standard defines many types of G-codes [17], only a subset is used in 3D printing [7]. G-codes are imperative directives generated as a result of slicing and path planning on the STL file by the G-code compiler. The firmware interprets these G-codes sequentially and then actuates the physical units accordingly.

3. Instruction-Level Energy Model and Profiler

In this section, we present an instruction-level energy model and an instruction-level energy profiler for 3D printers. First, we investigate the energy consumed by different G-code instructions. Second, we formulate an energy model based on the instruction-level energy consumption and the 3D printing operating mechanism. Lastly, we present an energy profiler based on the energy model to accurately simulate the printing energy consumption.

3.1 Instruction-Level Energy Model

3.1.1 G-code Instruction Types

Based on functionality, G-code instructions can be generally classified into three types:

- **Alignment Instruction (G0-Type):** This type of instruction is executed to swiftly align the nozzle from the current location to a specific X-Y coordinate. The format of G0-Type is as follows:

$$G0 \;\; F\langle speedrate\rangle \;\; X\langle coordinate\rangle \;\; Y\langle coordinate\rangle,$$

  where F is the linear speed of the motors. Usually, F is set to a high speed (e.g., 7200 mm/min) to save printing time as well as to avoid the stringing effect [1]. Particularly, there is no material extrusion during the operation.

- **Movement Instruction (G1-Type):** These instructions control the movement of the nozzle during the printing process. The format of G1-Type instructions is similar to G0 as follows:

$$G1 \;\; F\langle speedrate\rangle \;\; X\langle coordinate\rangle \;\; Y\langle coordinate\rangle$$

  In the case of G1, F is often empirically set between $30 \sim 7200$ mm/min, depending on print-resolution and design-intricacies.

- **Control Instruction (C-Type):** These instructions offer primitives to configure the printing environment. The functional scope of these instructions include settings such as the unit of measurement (mm or inches), the representation in absolute or incremental X-Y coordinates, the heater threshold temperature, etc. C-Type instructions attribute is perpetual and can only be altered after reset.

Instruction Category	Count	%
G0-Type: Align Instruction	36184	13.86%
G1-Type: Movement Instruction	224837	86.13%
C-Type: Control Instruction	5	0.001%
Total	261026	100%

Table 1: Instruction Distribution in a typical G-code file.

Table 1 shows an example of the instruction distributions of a typical G-code file. We can observe that the majority instructions are G1-Type.

3.1.2 Instruction-Based Energy Cost

To build an instruction-level energy model, the knowledge of the instruction-based energy cost is fundamental and necessary. Quantifying the instruction-based cost helps to find the printing phenomenon that causes the power consumption and further identify the power-hungry instructions.

Figure 3: Experimental setup for power measurement. A Wattsup power meter connected in series to a power line and a 3D printer.

Our experimental setup is shown in Figure 3. Specifically, we adopt Ultimaker 2 Go [56] because its firmware and hardware component specifications are open-source [57]. A *WattsUp* power meter [40] is connected in series with the power supply to measure the power consumption during the 3D printing process. *ProntherFace* [48] software is used to generate specific G-code(s) for instruction-level and component-level (motor, heater and fans) energy characterization.

The firmware executes the G-code instruction in the order of milliseconds. However, the sampling rate of the *WattsUp* meter is 1s, which is very coarse to measure the energy consumption of a single G-code instruction. To overcome this defect, we put every instruction in an infinite loop to generate a steady reading on the power meter. We remove the

Instruction Category	Avg. Power (W)
Alignment Instruction (G0-Type)	
G0 F7200 X Y	21.30
Movement Instruction (G1-Type)	
G1 F1800 X Y	39.90
G1 F2400 X Y	40.03
G1 F3600 X Y	39.94
G1 F4800 X Y	39.81
G1 F5200 X Y	40.38
G1 F5400 X Y	39.85
G1 F7200 X Y	40.06
Control Instruction (C-Type)	
G4: Pause	2.41
G20: Units to Inches	2.40
G21: Units to mm	2.43
G28: Move to origin	20.84
G90: Absolute Co-ordinates	2.41
G92: Set Position	2.39
M84: Disable Motors	2.40
M106: Fan on	3.61
M107: Fan off	4.44
M82: Extruder Position	2.43
M104: Set Temperature	3.50
M109: Attain Temperature	31.19

Table 2: Base energy cost of typical G-code instructions in the 3D printing process.

outliers and report the average power consumption in Table 2.

G0-Type Energy Cost During the execution of G0 instructions, the material extrusion mechanism is dis-functional and the speedrate is set at 7200mm/min. The average power consumption is 21.3W.

G1-Type Energy Cost G1 instructions move the nozzle to a specific position at different speeds. Particularly, we explore how the energy cost varies with the speed ($1800 \sim 7200$mm/min). As shown in the table, we only observe a marginal change in power consumption with a standard deviation of 0.19W. Therefore, we infer that the power consumption of G1 instructions is not sensitive to the speedrate. Also, G1 instructions consume more energy than G0 because extra physical components are activated, i.e, the heater and the fans.

C-Type Energy Cost The energy consumption of C-Type instructions differs a lot according to the specific operation they refer to. For instance, G28 is responsible for aligning the nozzle to the original point. Therefore, it has a similar power consumption as the G0-type. Another control instruction, M109, consumes high power (31.19W) since it turns on the heating process.

3.2 Instruction-Level Energy Model

We develop an instruction-level energy model for the 3D printing process. Let X_i denote the i^{th} instruction in the G-code file. The corresponding current I_i is the sum of the current drawn by the various components of the printer during X_i. Specifically, I_i is given by:

$$I_i = I_i^{motors} + I_i^{heater} + I_i^{fans}, \tag{1}$$

where I_i^{motors}, I_i^{heater} and I_i^{fans} are the current consumption of the motors, the heater and the fans.

Assume the X and Y coordinates of the instruction X_{i-1} and X_i are $\langle X_{i-1}, Y_{i-1}\rangle$ and $\langle X_i, Y_i\rangle$. Let L_i denote the distance movement of the nozzle during execution of X_i. Based on Eucledian principle, the length of traversal, L_i is given by:

$$L_i = \sqrt{(X_i - X_{i-1})^2 + (Y_i - Y_{i-1})^2}.$$

The time duration of X_i is given by: $T_i = \frac{L_i}{F_i}$, where F_i is the speedrate of instruction X_i. Therefore, the power consumption of X_i is given by: $P_i = I_i \times V_{cc}$. The corresponding energy consumption is:

$$E_i = P_i \times T_i = \frac{I_i \times V_{cc} \times L_i}{F_i}. \tag{2}$$

From Equation (2), the energy consumption for the entire printing process is formulated as:

$$E = \sum_{t=1}^{T} E_i = \sum_{i=1}^{T} \frac{I_i \times V_{cc} \times L_i}{F_i}, \tag{3}$$

where T is the total number of instructions under consideration. In practice, the supply voltage V_{cc} is regarded as a constant (119.5V) with a minor variation of ± 0.5V. Eventually, Equation (3) accurately accounts for the energy consumption of all G0-, G1- and C-Type instructions.

For simplicity, we did not take into account instruction inter-dependency in building the energy model. It is because the instruction execution in the 3D printing architecture contains limited hardware sharing and dependency [43], which is different from the traditional micro-process architecture. Our evaluation results in the following section also confirm this assumption.

Note that our contribution in this section is the methodology of building the instruction-level power model for the 3D printing process. Besides the FDM-type 3D printers, the same methodology can be applied to other types of 3D printers working on different technologies, such as Selective Laser Sintering (SLS) and Digital Light Processing (DLP).

3.3 Instruction-Level Energy Profiler

Based on the energy model established above, we develop a profiler that simulates the energy consumption of the 3D printing process at the instruction-level granularity.

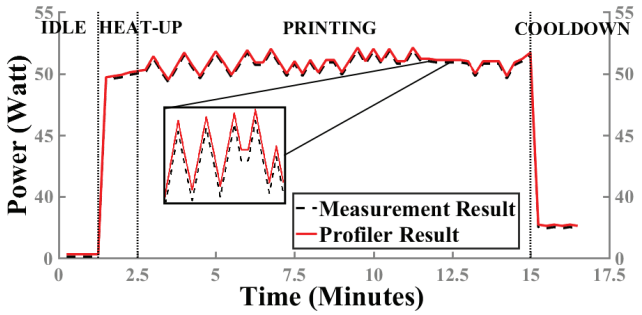


Figure 4: The real measurement and the result of the energy profiler in the 3D printing process.

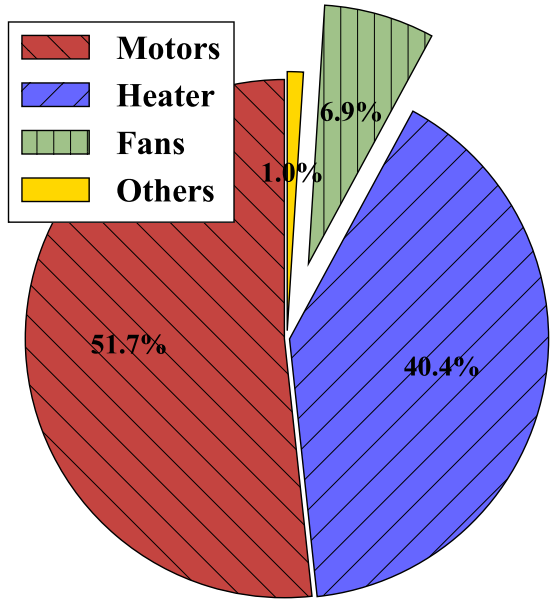


Figure 5: An energy sector diagram denoting the energy consumption of different physical components in a 3D printer.

3.3.1 Model Validation

To prove its correctness, we simulate a sample print on this profiler and validate the result with the ground truth measured by *Wattsup* power meter. The sample printing task is an artifact that takes 15 minutes, containing three types of instructions.

As depicted in Figure 4, the simulated results and the ground-truth measurements fit close to each other. Overall, the energy profiler achieves an error of 3.0%. The result shows that the empirically derived instruction-level energy model can precisely simulate the real power consumption of the 3D printing process.

3.3.2 Energy-Sector Diagram

There are three energy-hungry physical units: the stepper motors, the heater and the fans. We account for the energy consumption of the heater and the fans by considering M109 and M106 for the entire duration of the print. Likewise, we account for the energy consumption of the stepper motors by considering G0- and G1-Type instructions alone (subtracting I_i^{heater} and I_i^{fans} from Equation 1) in the observed readings.

Figure 5 shows the distribution of energy consumption among the physical components over 338 benchmarks (details in Section 5.1). Instead of the heater, we notice that the motors are the dominant energy consuming components. Motors contribute 51.7% of the overall power consumption in the 3D printing process. Therefore, we focus on optimizing the power consumption of the motors, not the heater. Heater power optimization is an orthogonal research problem.

3.3.3 Substantial Amount of Straight-Line Movements

After the in-depth analysis of the instruction movement pattern over 338 benchmark G-code files, we find that the straight line movements along the X-Y axis are dominant. The reason for such an observation is because the G-code compiler slices the CAD design into geometric patterns. These geometric patterns usually align along the axes when oriented correctly: at least 95.0% for "triangles" in-fill pattern and at most 98.7% for "lines" in-fill pattern (explained in Figure 12) - a significant percentage. Based on this observation, we hypothesize that the energy consumption of a 3D printer can be heavily reduced by dynamically power-gating the X and Y motors.

Insights

We obtain three important insights from the above energy characterization study:

- Reducing the motor functioning time has the potential to significantly cut the energy consumption in the 3D printing process.
- The instruction-level simulator provides an accurate account of the energy consumed in the 3D printing process.
- Energy optimization strategies (such as energy-efficient orientation of the design, etc.) can be tested on the simulator rather than conducting time consuming prints on a real 3D printer.

4. A Cross-Layer Energy Reduction Approach for 3D Printers

In this section, we describe *3DGates*: a cross-layer power-gating strategy to reduce energy consumption in 3D printers. *3DGates* extends the G-code standards, the firmware and the G-code compiler to facilitate energy reduction.

4.1 Approach Overview

(A) Design Considerations As discussed in Section 3, motors constitute the dominating factor in the overall energy consumption in 3D printers. For the sake of feasibility and

effectiveness on real 3D printers, we list the design considerations as follows:

- *Integrity (No changes to original CAD Design):* Our approach intends to keep the original STL file intact because any potential alteration to the design might either not result in a correct printing or affect fabrication quality such as mechanical strength.
- *Robustness (No changes in Hardware):* Hardware support, such as adding more energy-efficient controllers, might improve the power management of 3D printers. However, we aim to propose a practical and robust solution that is hardware-nonspecific and can cooperate with any controllers. Therefore, the design considerations of our approach specifically ensures no changes in the 3D printer's hardware.
- *Comprehensiveness (Cross-layered Design):* Referring to the current 3D printing process flow, the compiler should first access the digital design and generate energy-efficient G-code instructions. The firmware then parses these new G-code instructions and controls the physical components accordingly. Therefore, the modifications in different layers constitute a comprehensive approach to reduce energy consumption.

(B) Design Framework An overview of our framework is shown in Figure 6. Specifically, the left part is the existing 3D printing flow and the red circle on the right illustrates our implementation, including new G-code instruction support, firmware support and compiler support.

We first design a set of extended instructions to allow advanced control over the motor. The *re-compiler* takes the original G-code instructions from the compiler and inserts the new power-gating instructions at proper positions. Note that the re-compiler only inserts instructions without deleting any entries from the G-code file, thus ensuring integrity. Afterwards, we develop the *firmware support interface* to enable parsing of the new instructions.

Note that our approach can be integrated into the existing 3D printing flow. Also, the new G-code file still holds backward compatibility towards other non-upgraded 3D printers. The non-upgraded firmware on these printers would just skip parsing the new instructions.

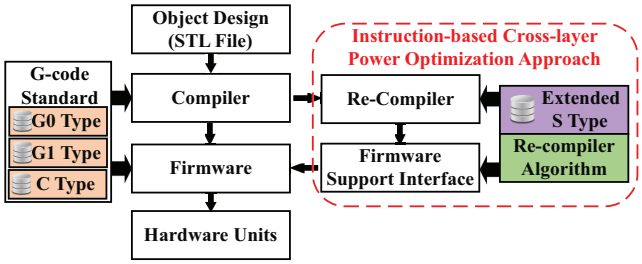


Figure 6: *3DGates* Architecture: The instruction-based cross-layer motor power-gating strategy for 3D printers.

S-series	Function	Hardware Control	Pwr.(W)
$S\ X\ on$	X-motor **on**	GPIO Pin18 High	2.39
$S\ X\ off$	X-motor **off**	GPIO Pin18 Low	2.38
$S\ Y\ on$	Y-motor **on**	GPIO Pin17 High	2.40
$S\ Y\ off$	Y-motor **off**	GPIO Pin17 Low	2.42

Table 3: S-series instructions to support motor power-gating.

4.2 Instruction Support

Instruction support for the fine-grained motor control is critical for *3DGates*. The G-code standard defines M84 to control the motor power supply. However, M84 is only capable of turning on/off all motors together. Therefore, an extension to the G-code standard is required for dynamic power-gating of individual motors at run-time. Note that we define new G-code instructions rather than altering existing ones for the sake of compatibility on different 3D printers.

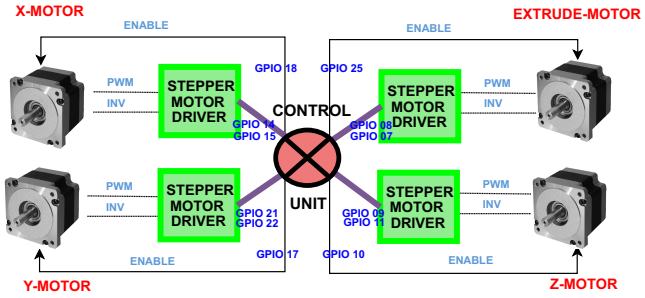


Figure 7: Stepper motor control mechanism - the firmware perspective.

S-Series Instruction Implementation We introduce new G-code instructions, called S-series[1] instructions, to achieve the fine-grained control over the motor. As listed in Table 3, the S-series instruction is able to turn on/off a specific motor.

Figure 7 shows the connection graph between the control unit and the stepper motor. Specifically, the control unit is a microcontroller which has multiple GPIO pins for data communication. Each motor has a Power-Enable (PEN) pin to gate the input power. For the Ultimaker 3D printer, GPIO 10, 17, 18 and 25 are connected to PEN pins of the motors. Therefore, based on the S-series instructions, we can alter the voltage on the GPIO pin to control the motor operation.

Feasibility of S-Series Energy Savings S-series instructions enable dynamic power control of the stepper motors at run-time. Preliminarily, we examine the S-series instruction's effectiveness by comparing the power consumption over the instructions listed in Table 2. We consider the instruction along a single axis after alignment. As shown in Table 4, for a specific instruction, the new power column refers the power consumption when the S-series instruction is applied.

[1] S denotes saving of power in motor controls.

Instr.	New Pwr(W)	Old Pwr(W)	% Reduction
		Align Instr.	
G0	12.53	21.3	**41.17%**
		Movement Instr.	
G1	25.81	39.9	**35.3%**
	25.06	40.03	**37.4%**
	24.69	39.94	**38.2%**
	24.50	39.81	**38.4%**
	24.54	40.38	**39.2%**
	24.90	39.85	**37.5%**
	25.30	40.06	**36.8%**
		Control Instr.	
G4	2.41	2.41	-
G20	2.40	2.40	-
G21	2.43	2.43	-
G28	20.84	20.84	-
G90	2.41	2.41	-
G92	2.39	2.39	-
M84	2.40	2.40	-
M106	3.61	3.61	-
M107	4.44	4.44	-
M82	2.43	2.43	-
M104	3.50	3.50	-
M109	31.19	31.19	-

Table 4: Reduction in G-code power consumption with new S-series instruction support.

We can observe that a significant portion of power consumption is reduced for G0 and G1 instructions by dynamically power-gating the motors. The control instructions are not affected because the motors are not involved (G28 activates the motor but we do not consider it along one axis). Because the power consumption is not sensitive to speedrate, the power reduction of S-series instructions across different speedrates is also consistent, with a standard deviation of 1.26%. The average power savings is 38.0% across G0 and G1 instructions. Therefore, the motor power-gating strategy can effectively reduce the power cost. We further explain the necessary firmware support and compiler support for implementation.

4.3 Firmware Support

Firmware Extension to S-Series Instructions To facilitate the interpretation of the new S-series instructions on 3D printers, it is necessary to upgrade the 3D printer firmware with an extended module to interpret the new instructions. As shown in Figure 8, this module processes S-series instructions and governs the functioning of the motors through the GPIO pins. Specifically, the extended module provides

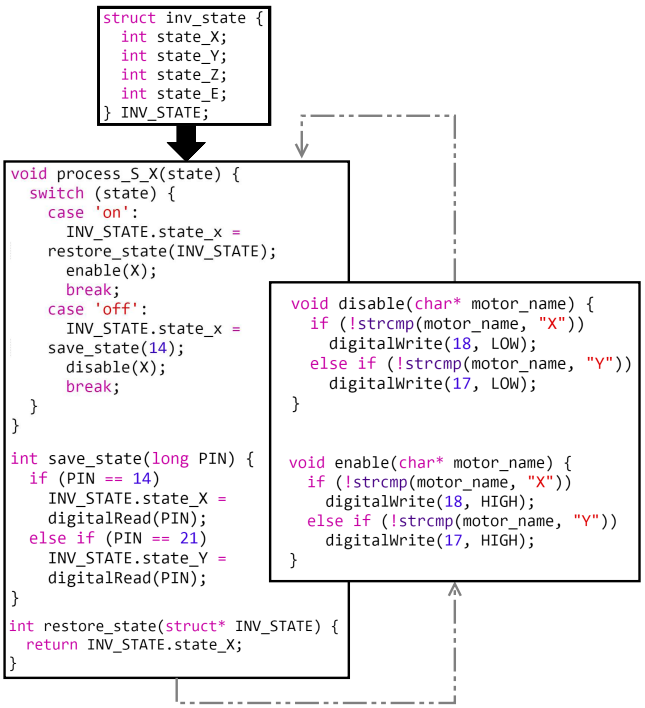


Figure 8: Upgraded firmware: interface to new instructions and the INV_STATE data structure.

interfaces to enable (HIGH) and disable (LOW) the GPIO pins.

State Backup The stepper motor driver on the microcontroller board has three output pins that send commands to the stepper motors, i.e., Power-Enable (PEN), Invert (INV) and Pulse-Width-Modulation (PWM). During our experiments, we notice that when an enable signal (HIGH) is applied to PEN, the state of INV assumes a random value - either HIGH or LOW. This behavior usually leads to the incorrect printing direction when the NEMA17 motors are cold-started because the INV state exists as a register-like implementation in the firmware. Therefore, it is necessary to back up the INV state before turning off the motors and restore the state before turning them back on again. The extended firmware module defines a new structure, INV_STATE, to backup the INV pin state to avoid the incorrect direction error.

4.4 Compiler Support

With the extended instruction and firmware support, 3D printers are capable of fine-grained motor power control. However, the firmware cannot identify when to power-gate the motors because it has no access to the entire design. Therefore, we develop a re-compiler engine to insert the power-gating instructions into the original G-codes file. Figure 9 demonstrates an output of the re-compiling process.

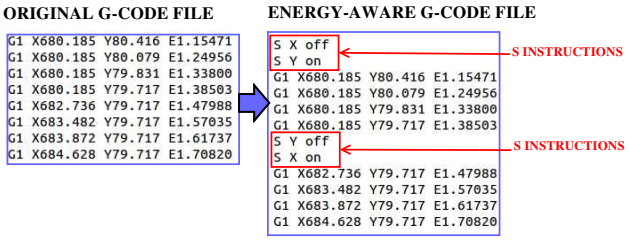
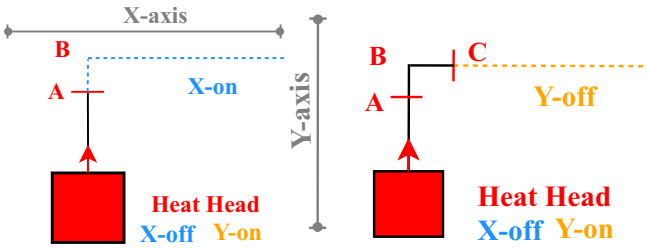


Figure 9: Re-compiler example: inserting power-gating instructions.



(a) **Start-ahead** (b) **End-late**

Figure 10: Control mechanism to account for instruction-inertia and instruction-delay.

Instruction-Delay and Instruction-Inertia The 3D Printer is a cyber-physical system: the cyber part processes the instructions, and the physical part executes the instructions. Due to the intrinsic disparity of response time between cyber (usually in MHz or GHz) and physical (usually in Hz) domains, there is a delay between the instruction interpretation and execution. It is critical to incorporate this delay in the system manipulation to avoid the possible timing error and the printing misbehavior.

To ensure the correctness, we formulate two properties with regard to this time disparity: *Instruction-inertia* and *Instruction-delay*. *Instruction-inertia* is the extra time to actually start the motor from the time the *S-on* instruction is executed; *Instruction-delay* is the extra time required to actually stop the motors from the time the *S-off* instruction is executed. These two properties lead to special considerations in designing the re-compiler.

1. **Start-Ahead** ***(To account for instruction-inertia)***: Because of the delay from instruction parsing to execution, the *S-on* instructions need to Start-ahead before actual usage. Figure 10(a) shows an example of instruction start-ahead. X-motor needs to be switched on before the nozzle reaches the turning corner B. In other words, both X-motor and Y-motor are switched on when the nozzle moves from Position A to B, even though there is no X-axis movement yet. In this study, Ultimaker 3D printers use NEMA 17 stepper motors with a startup delay of 1.8ms [56]. Hence, we set our Start-ahead time as 1.8ms.
2. **End-Late** ***(To account for instruction-delay)***: The inertial property of the motors demands that the instruction under execution needs extra time, called End-late, to actually finish the execution. As shown in Figure 10(b), the Y-motor should stay switched on until the nozzle reaches Position C. In other words, both X-motor and Y-motor are switched on when the nozzle moves from Position B to C, even though there is no Y-movement. If the Y-motor was turned off exactly at B, motor wiggling might occur due to the inertia of the motor. End-late avoids the misalignment caused by motor wiggling and ensures no print defects when power-gating. Compared to Start-ahead, End-late is empirical value and we set it as 1.0ms in this study.

Re-compiling Algorithm The goal of the re-compiler engine is to populate the G-code files with S-series instructions. There are two guarantees when re-compiling the G-code files. *First*, minimum quantity of S-series instructions are inserted. *Second*, no explicit pause or latency is introduced. These two guarantees are achieved as follows:

- *Instruction Grouping:* The first phase of the re-compiling algorithm is to find consecutive instructions with similar motor movements. Given a raw G-code file, we group the instructions, layer-by-layer, that can be power-gated by the same axis motor. We use a labeled-weight approach to group the instructions [53]. In the first step, based on the motor movements in the G-code file, we assign an index to each instruction with a weighted label. The label annotates the path taken by the motors, and the weight indicates the number of active motors for that path. Instruction movements within 0.5mm are assigned the same weight. In the second step, we search the longest common subsequence [44] among the labels with the same weight. The instructions within each common subsequence are grouped as one block. This phase ends when all instructions are grouped.
- *Model Rotation in the X-Y Plane:* The second phase of the re-compiling process rotates the entire model in the X-Y plane to align the grouped instructions to either the X or Y axis. The model is rotated in 30° increment on the X-Y plane. The orientation containing the maximum quantity of the total straight X and Y movements is selected. Phase two ensures minimum insertion of S-series instructions at the right places, thereby guaranteeing maximum energy-efficiency. Note that we do not rotate the model in 3D-space since it would lead to additional support material generation.
- *S-Series Instruction Insertion:* The third phase of the re-compiler inserts S-series instructions to power-gate the motors. The re-compiler will go through the instruction groups produced by the second phase and then sandwich them between S-ON and S-Off instructions. The design considerations of *Start-ahead* and *End-late* are applied

here by appropriately splitting up the instruction if required. In cases where some instruction groups have the same label weight, the re-compiler processes the group with more instructions first.

The re-compiler algorithm is listed in Algorithm 1. The re-compiler is efficient, and its time complexity is polynomial. Note that the re-compiler algorithm is implemented towards three degrees-of-freedom (DOF), and can be applied to additive manufacturing processes with a higher DOF [36].

Algorithm 1 Re-compiler Algorithm

procedure RE-COMPILING_ALGORITHM($naive\_Gcodes$)
  $neigh_t \leftarrow 0.5$ ▷ Neighborhood Threshold = 0.5mm
  $X\_clusters \leftarrow \phi$
  $X\_grp\_naive \leftarrow Group\_X\_terms(naive\_Gcodes)$
  $Y\_grp\_naive \leftarrow Group\_Y\_terms(naive\_Gcodes)$
  for all deg in 30° increments **do** ▷ X-Y Rotation
    $(X\_groups, Y\_groups) \leftarrow axes\_align(X\_grp\_naive, Y\_grp\_naive)$
  end for
  for all X **in** X_groups **do**
    if $|(X_{prev}+neigh_t)| \leq X \leq |(X_{next}-neigh_t)|$ **then**
      $X\_cluster \leftarrow X\_cluster + X$
    else if $X_{next} = Y\_term$ **then** ▷ Instr. delay
      $X \leftarrow X_1 + S\ Y\ on + X_2$
      $X\_cluster \leftarrow X$
    else if $X_{prev} = Y\_term$ **then** ▷ Instr. inertia
      $X \leftarrow X_1 + S\ Y\ off + X_2$
      $X\_cluster \leftarrow X$
    end if
  end for
  for all $cluster$ **in** $X\_cluster$ **do**
    $cluster_{front} \leftarrow S\ X\ on$
    $cluster_{back} \leftarrow S\ X\ off$ ▷ X power-gating
  end for
  repeat $9:23\ for\ Y\_clusters$
  return $energyOptimized\_Gcodes$
end procedure

5. Evaluation

In the section, we comprehensively evaluate the performance of *3DGates*. First, we investigate the energy reduction through both a simulated study and a real-world experiment. Second, we explore the possible process factors to impact the system performance. Lastly, we examine the impact of *3DGates* on printing duration and printing quality.

5.1 Evaluation Setup

Benchmark Preparation: To comprehensively evaluate *3DGates*, we select 338 benchmark designs from Thingverse [54]. All benchmarks are real printable products and cover domains ranging from daily household replacements to specialized industrial components. The benchmarks comprise about 71 million lines of G-code instructions, and the total estimated fabrication time is around one year on one 3D printer.

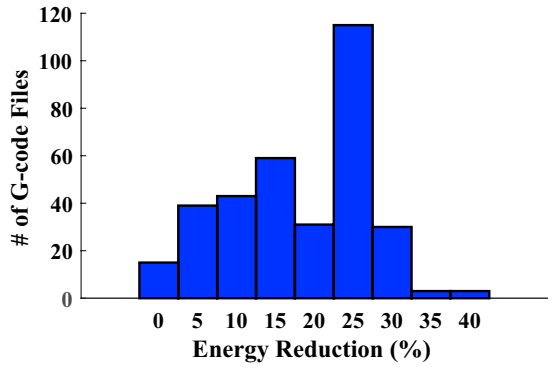


Figure 11: Histogram of energy reduction after applying power-gating on 338 G-code benchmark files.

Specifically, we adopt the developed energy profiler to simulate the energy consumption of 338 benchmarks for practical concern. The simulator is implemented in Python with 487 LoC. The experiment tests are performed on a desktop with a quad-core Intel CPU, 4GB RAM, 2TB SSD Hard Disk. For the experiments with acceptable printing time, we conduct the real measurement with the designs on Ultimaker 2 Go [56] to explore the characteristic of *3DGates*. The firmware upgrade patch is implemented in the C language, and the re-compiler algorithm is integrated into the Cura [19] engine, a publicly available 3D printing compiler.

5.2 Performance Evaluation

We evaluate *3DGates* by employing power-gating strategy to 338 benchmark G-code files and simulating the power consumption by the energy profiler. For each file, we compare the energy consumption before and after the power optimization. Figure 11 shows the distribution of the power savings over 338 G-code files. On average, *3DGates* offers an energy reduction of 25%. Specifically, 180 (53.2%) of the benchmark achieve the energy reduction above 20%. The largest energy saving reaches up to 37.9%.

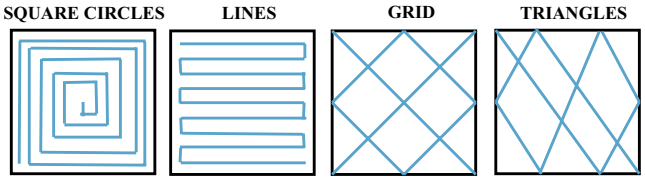


Figure 12: Different object in-fill patterns.

5.3 Sensitivity Analysis

There is a large configuration space in the 3D printing process. For the sake of comprehensiveness, we study the per-

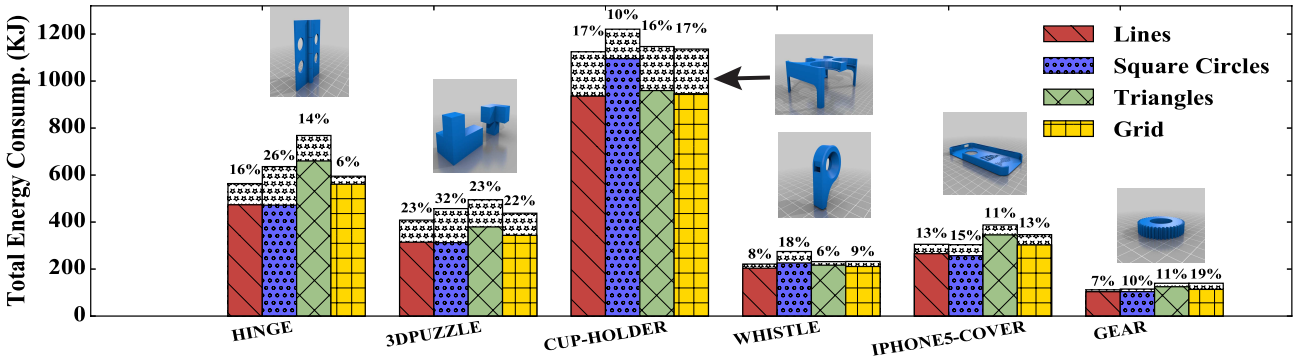


Figure 13: Sensitivity of different infill patterns on different designs. The colored bars refer to the energy consumption with the power-gating strategy. The dotted bars on the top denote the energy savings compared to the cases without energy optimization. The percentage numbers above the dotted bars is the absolute energy reduction by our approach.

formance sensitivity to 3D printing process configurations. Considering that the power-gating strategy is highly related to the nozzle movement style, we specifically explore two aspects in this study, i.e., *in-fill pattern* and *object orientation*.

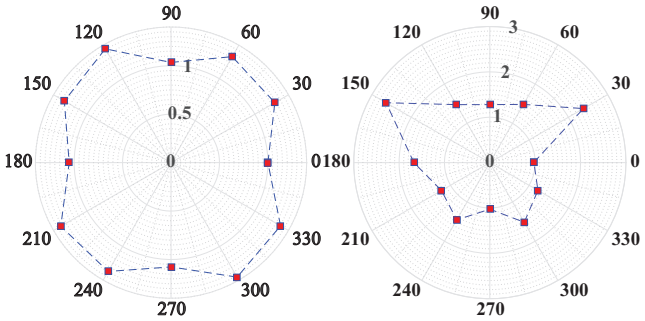
In-Fill Pattern: The in-fill pattern specifies the toolpath pattern the nozzle follows when it fills the interior of the design in each layer. There are four typical in-fill patterns: (1) square-circle, (2) line, (3) grid and (4) triangle, in 3D printing. Figure 12 demonstrates the trajectory of each type. Note that no specific in-fill pattern requires a significantly less energy consumption than others due to that the total volume of filament use, i.e., the length of the tool path, is the same among different in-fill patterns. Nevertheless, the in-fill pattern will affect the performance of *3DGates* because different patterns might provide distinct opportunities to power-gate motors and reduce the energy consumption during the 3D printing process.

We evaluate the energy impact of four in-fill patterns on six designs: hinge, 3D-puzzle, cup-holder, whistle, iphone5-cover and gear. The designs are printed by the printer and the power consumption in each case is measured. Figure 13 shows the energy comparison results after implementing *3DGates* with different in-fill patterns. We observe that the performance varies with different in-fill patterns. The average energy reductions are 18.67%, 23.50%, 12.00% and 17.00% for line, square-circle, triangle and grid, respectively.

Square-circle pattern achieves the best performance with respect to energy efficiency because it consists of orthogonal movements which can be aligned along the X and Y axes with a proper orientation adjustment. It indicates that there is a higher probability to switch off one motor during the entire process. The case study also confirms that about half of the motor energy is saved during the printing process in the square-circle pattern.

Line pattern also consists of orthogonal movements. Different from the square-circle pattern, the majority of the movements fill along one direction only. Therefore, the short segments on the other direction cannot take advantage of power-gating motors due to the constraints from instruction inertial/delay. Note that this is one exceptional case, i.e., cup-holder, where the design is round shape, line out-performs square-circle by 7% due to reduction of non-extrusion movements.

The in-fill patterns of grid and triangle can make the printed object with a better mechanic strength. Due to orthogonal segments, grid pattern achieves similar performance as square-circle in certain situations. Triangle pattern, on the other hand, comprises non-orthogonal segments operated by both X and Y motors. As a result, power-gating is not activated most of the time. Therefore, the average energy reduction of triangles is the lowest among the four in-fill patterns.



(a) Gear (b) IPhone5-cover

Figure 14: Energy consumption with power-gating strategy under different orientations. The markers represent the normalized energy consumption.

Object Orientation: In the re-compiler algorithm, we brute-force the orientation angle of the design by assuming that it will affect the efficiency of power-gating. For the purpose of

demonstration, we select gear and iphone5-cover as examples and measure the optimized power consumption under different orientation angles in the 2D X-Y plane. To minimize the effect of the in-fill pattern on our orientation results, we select the triangle in-fill pattern. Specifically, we increase the angle from 0° to 360° with an increment of 30°.

Figure 14 depicts the optimized power consumption with different orientations for each design. Note that the data is normalized that the result at 90° equals to 1. We observe that the optimized power consumption doesn't vary too much with the orientation (the standard deviation is only 0.14). This is because gear is very symmetric in shape and hence the power-gating results are quite the same with all angles. However, for the asymmetric designs such as the iphone5-cover, the power consumption results in a larger variation, obtaining maximum at 150° and minimum at 0°. Therefore, the orientation bruteforce is necessary to achieve the largest energy reduction.

5.4 G-code File Size

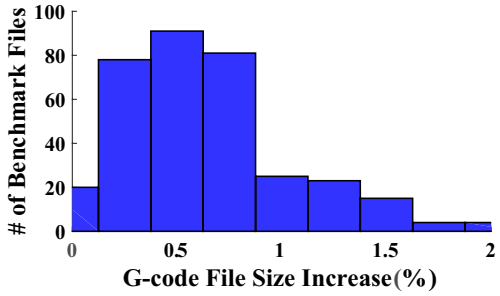


Figure 15: Impact on G-code file size.

Our S-series instruction addition has negligible impact on the G-code file size. We summarize a file size increase on all 338 benchmarks after applying *3DGates*. As shown in Figure 15, the maximal increase of file size is less than 2%, and the average is only around 0.5%. Therefore, our solution doesn't introduce overhead in data storage or transmission.

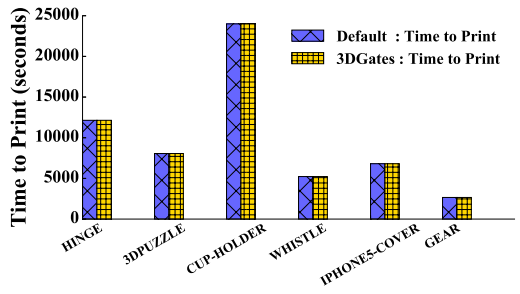


Figure 16: Comparison of printing-efficiency. There is no observable change of printing time in the *3DGates* solution.

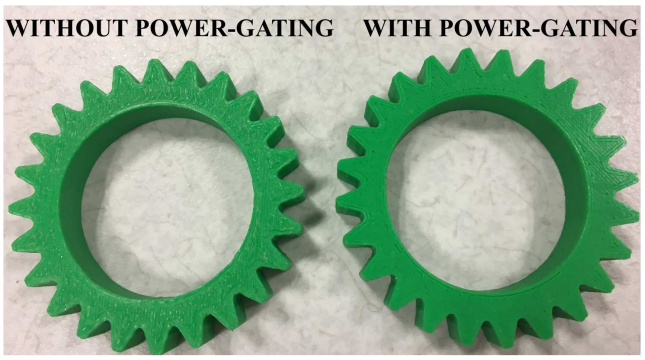


Figure 17: Comparison of printing quality. No any significant visual differences can be observed.

5.5 Impact on Real-world Printing

We employ *six* 3D designs mentioned above to examine the impact of *3DGates* on the real-world printing performance from two aspects: printing duration and printing quality.

Impact on Printing Duration: For each design, we print it twice by Ultimaker 2 Go with the G-code files with and without power-gating. We record and compare the printing duration in each case. As shown in Figure 16, there is negligible difference in printing time across all cases (less than 1% on average). Therefore, *3DGates* is fully compatible with current printing practices without slowing the process.

Impact on Printing Quality: We also examine whether *3DGates* will affect the printing quality. Figure 17 depicts one pair of examples from printed designs. The left one is the original process, and the right one is the power-optimized process. We observe there is no noticeable differences in quality. The structure and the texture from two processes are almost identical.

Summary: The aforementioned results show that 3DGates is effective and practical to reduce the power consumption of 3D printers. Moreover, 3DGates has no noticeable effects on printing duration and quality.

6. Discussion

Finer Stepper Motor Power Control: Instruction-level power optimization can be further enhanced with fine-grained motor power control. For example, DVFS mechanisms [26, 31, 60, 62] can offer more aggressive controls over the stepper motor to improve energy-efficiency with respect to different speeds. Dynamic voltage scaling of the motors, however, requires extra hardware support (e.g., configurable voltage regulator [35]) and might affect the printing quality [12]. Our proposed instruction-level solution will neither affect the printing quality nor need any additional hardware support.

3D Orientation Optimization: Figure 14 implies that the efficiency of power reduction varies with the design rotation in the 2D plane. We anticipate that the power reduction

can be further improved by altering the object orientation in the 3D space. Principle component analysis (PCA) [29] on path directions can adjust the orientation to maximize the opportunity of power-gating the motors. However, 3D orientation will also affect other aspects, such as structure support generation [50]. In this work, we focus on power reduction without generating any additional structure support.

Interdisciplinary Methods in Energy Reduction: As an emerging cyber-physical system, the 3D printer [58] is a holistic design of mechanical engineering, electrical engineering, material science and computer science. The energy reduction on 3D printers demands an interdisciplinary effort. For example, low melting-temperature materials can reduce heating power consumption [21] and better mechanical design of the motor can improve motion energy efficiency [22]. This work only concentrates on power optimization from the system design perspective. Methods of altering electrical, mechanical or material behaviors of 3D printing are beyond our consideration.

7. Related Work

Energy-Efficiency in 3D Printers: There has been some preliminary work on better path planning and G-code generation strategies to reduce energy consumption. These works can be classified into two categories. The first category focuses on minimizing the printing time to reduce energy consumption. For example, Jin *et al.* investigated an adaptive path planning algorithm to decrease nozzle travel distance for energy savings [28]. The second category is to avoid unnecessary motions to cut energy consumption. Volpato *et al.* proposed an optimization algorithm to reduce repositioning distances in FDM 3D printers [59]. Lensgraf *et al.* presented a new path planning algorithm to minimize "untrutionless" movement in a single layer by printing nearest neighbors [37]. Existing approaches, however, require a recompilation of the entire design and generation of completely new path planning results (i.e., G-codes), which might affect the design and mechanical properties of the fabricated objects. To the best of our knowledge, *3DGates* is the first study to reduce energy consumption without changing the path planning in 3D printers.

System Power Modeling: Power modeling has a long and rich research history and is a core research topic in computer systems. There are many research works on modeling micro-processor systems (e.g., CPU [13, 30] embedded platforms [18]) to end-user systems (e.g., smartphones [67, 15]). For example, Tiwari *et al.* developed an instruction-level power model of a micro-processor and studied software optimizations for power reduction [55]. Pathak *et al.* proposed a system-call-based power modeling approach [46] and profiled energy consumption of smartphone apps [45]. These works on power models provide excellent references to study the instruction-level power model of 3D printers. As a mechatronic cyber-physical system, the power model of 3D printers also includes mechanical motion and heating, which is beyond computing power models.

System Power Optimizations: In general, power management and optimization include both online and offline approaches. Online approaches are applicable to applications where inputs and use-conditions are unpredictable. Related system techniques include device driver control [66, 32], device interface design [9] and run-time system configuration [25]. Offline power optimization is used for predictable application tasks. Related work includes task scheduling [52], partitioning [10], configuration [20], and resource management [11]. In this study, we choose offline power optimization, which can reach the global optimality and usually provide better results. Binary instrumentation in compiler [42, 49] has the analogous concept to our re-compiler approach. However, our approach to mechatronic cyber-physical systems takes into consideration properties such as instruction-inertia and instruction-delay.

Better Material Design: One benefit of 3D printing is that a wide range of materials can be used to acquire a product [24]. Polymers [16, 34, 51], metals [23, 38] and ceramics [39, 41] are the common materials in use. Using better materials with lower melting temperatures is an active area of research to reduce energy consumption by the heater in 3D printers. However, most existing solutions come with tradeoffs such as incorporating specific hardware [33], compromising printing time [14], reducing product life-time [27], and even lowering product quality [65]. *3DGates* does not affect the printing time or the printing quality. Moreover, it does not propose any hardware changes or new material usage.

8. Conclusion

This paper conducts an instruction-level power analysis on FDM 3D printers. Specifically, we present a unified cross-layer power optimizing approach encompassing instruction, hardware, firmware and compiler layers. Leveraging the knowledge of the 3D printing mechanism, we power-gate the motors and achieve 25% energy reduction in 3D printers. More importantly, it is accomplished without any modification in hardware, increase in printing time or defect in printing quality. To further reduce the energy consumption, we envision solutions across domains including material, industrial and electrical sciences.

9. Acknowledgments

We thank Dr. Felix Xiaozhu Lin at Purdue University for constructive suggestions in this work. We thank the anonymous reviewers and their paper shepherd, Dr. Xipeng Shen, for providing insightful feedback. This work was in part supported by a National Science Foundation grant CNS-1547167 and a seed grant from the UB Sustainable Manufacturing and Advanced Robotics Technologies Community of Excellence (SMART CoE).

References

[1] 3D Printing Stringing. https://www.matterhackers.com/articles/retraction-just-say-no-to-oozing. Accessed: 2016-11-8.

[2] Currently available Portable 3D Printers. https://3dprint.com/tag/portable-3d-printer/. Accessed: 2016-10-22.

[3] International Energy Outlook 2016. http://www.eia.gov/outlooks/ieo/. Accessed: 2016-05-11.

[4] ONO 3D Printers. http://www.ono3d.net/. Accessed: 2016-10-22.

[5] TekBot Portable 3D Printers. https://3dprint.com/107375/tek-bot-portable-printer/. Accessed: 2016-10-22.

[6] 3D Hubs. 3D Printig Industry Trends Q3-2016. https://www.3dhubs.com/trends. Accessed: 2016-11-8.

[7] Adrian Bowyer. G-code. http://reprap.org/wiki/G-code. Accessed: 2016-05-22.

[8] J. Ajay, A. S. Rathore, C. Song, C. Zhou, and W. Xu. Don't Forget Your Electricity Bills! An Empirical Study of Characterizing Energy Consumption of 3D Printers. In *ACM SIGOPS Asia-Pacific Workshop on Systems (APSys)*, pages 1 – 8, Hong Kong, China, August 2016.

[9] M. Anand, E. B. Nightingale, and J. Flinn. Ghosts in the machine: Interfaces for better power management. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 23–35. ACM, 2004.

[10] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 9–pp. IEEE, 2003.

[11] E. Bini, G. Buttazzo, J. Eker, S. Schorr, R. Guerra, G. Fohler, K.-E. Årzén, V. Romero, and C. Scordino. Resource management on multicore systems: The actors approach. *IEEE Micro*, 31(3):72–81, 2011.

[12] T. Brajlih, B. Valentan, J. Balic, and I. Drstvensek. Speed and accuracy evaluation of additive manufacturing machines. *Rapid prototyping journal*, 17(1):64–75, 2011.

[13] J. A. Butts and G. S. Sohi. A static power model for architects. In *Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture*, pages 191–201. ACM, 2000.

[14] T. Campbell, C. Williams, O. Ivanova, and B. Garrett. Could 3d printing change the world. *Technologies, Potential, and Implications of Additive Manufacturing, Atlantic Council, Washington, DC*, 2011.

[15] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*, volume 14. Boston, MA, 2010.

[16] L. G. Cima and M. J. Cima. Preparation of medical devices by solid free-form fabrication methods. *Robotics and Computer Integrated Manufacturing*, 4(12):371, 1996.

[17] CNC Cookbook. CNC-code. http://www.cnccookbook.com/CCCNCGCodeRef.html. Accessed: 2016-05-22.

[18] G. Contreras and M. Martonosi. Power prediction for intel xscale® processors using performance monitoring unit events. In *ISLPED'05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.*, pages 221–226. IEEE, 2005.

[19] Cura. 3D Printing Slicing Software. https://ultimaker.com/en/products/cura-software. Accessed: 2016-05-22.

[20] A. Emadi, K. Rajashekara, S. S. Williamson, and S. M. Lukic. Topological overview of hybrid electric and fuel cell vehicular power system architectures and configurations. *IEEE Transactions on Vehicular Technology*, 54(3):763–770, 2005.

[21] D. Frear and P. Vianco. Intermetallic growth and mechanical behavior of low and high melting temperature solder alloys. *Metallurgical and Materials Transactions A*, 25(7):1509–1523, 1994.

[22] J. F. Gieras. *Permanent magnet motor technology: design and applications*. CRC press, 2002.

[23] D. Godlinski and S. Morvan. Steel parts with tailored material gradients by 3d-printing using nano-particulate ink. In *Materials Science Forum*, volume 492, pages 679–684. Trans Tech Publ, 2005.

[24] M. Greulich, M. Greul, and T. Pintat. Fast, functional prototypes via multiphase jet solidification. *Rapid Prototyping Journal*, 1(1):20–25, 1995.

[25] H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard. Dynamic knobs for responsive power-aware computing. In *ACM SIGPLAN Notices*, volume 46, pages 199–212. ACM, 2011.

[26] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, et al. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In *2010 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 108–109. IEEE, 2010.

[27] S. E. Hudson. Printing teddy bears: A technique for 3d printing of soft interactive objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 459–468, New York, NY, USA, 2014. ACM.

[28] G. Jin, W. Li, C. Tsai, and L. Wang. Adaptive tool-path generation of rapid prototyping for complex product models. *Journal of manufacturing systems*, 30(3):154–164, 2011.

[29] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[30] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 135–140. ACM, 2001.

[31] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134. IEEE, 2008.

[32] K. Klues, V. Handziski, C. Lu, A. Wolisz, D. Culler, D. Gay, and P. Levis. Integrating concurrency control and energy management in device drivers. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 251–264. ACM, 2007.

[33] S. H. Ko, J. Chung, N. Hotz, K. H. Nam, and C. P. Grigoropoulos. Metal nanoparticle direct inkjet printing for low-temperature 3d micro metal structure fabrication. *Journal of Micromechanics and Microengineering*, 20(12):125010, 2010.

[34] C. X. F. Lam, X. Mo, S.-H. Teoh, and D. Hutmacher. Scaffold development using 3d printing with a starch-based polymer. *Materials Science and Engineering: C*, 20(1):49–56, 2002.

[35] H.-P. Le, J. Crossley, S. R. Sanders, and E. Alon. A sub-ns response fully integrated battery-connected switched-capacitor voltage regulator delivering 0.19 w/mm 2 at 73% efficiency. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 372–373. IEEE, 2013.

[36] W. Lei and Y. Hsu. Accuracy test of five-axis cnc machine tool with 3d probe–ball. part i: design and modeling. *International Journal of Machine Tools and Manufacture*, 42(10):1153–1162, 2002.

[37] S. Lensgraf and R. R. Mettu. Beyond layers: A 3d-aware toolpath algorithm for fused filament fabrication. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3625–3631, May 2016.

[38] J. Liu and M. Rynerson. Method for article fabrication using carbohydrate binder, July 1 2003. US Patent 6,585,930.

[39] M. C. Melican, M. C. Zimmerman, M. S. Dhillon, A. R. Ponnambalam, A. Curodeau, and J. R. Parsons. Three-dimensional printing and porous metallic surfaces: A new orthopedic application. *Journal of biomedical materials research*, 55(2):194–202, 2001.

[40] W. U. P. Meters. Watts up? .Net Power Meter Specifications. `https://www.wattsupmeters.com/secure/products.php?pn=0&wai=0&spec=2`. Accessed: 2016-05-22.

[41] J. Moon, A. C. Caballero, L. Hozer, Y.-M. Chiang, and M. J. Cima. Fabrication of functionally graded reaction infiltrated sic–si composite by three-dimensional printing (3dp™) process. *Materials Science and Engineering: A*, 298(1):110–119, 2001.

[42] N. Nethercote and J. Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. In *ACM Sigplan notices*, volume 42, pages 89–100. ACM, 2007.

[43] OhmEye. Print fans and hotend heaters. `http://ohmeye.com/2015/print-fans-and-hotend-heaters/`. Accessed: 2016-11-8.

[44] M. Paterson and V. Dančík. Longest common subsequences. In *International Symposium on Mathematical Foundations of Computer Science*, pages 127–142. Springer, 1994.

[45] A. Pathak, Y. C. Hu, and M. Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 29–42. ACM, 2012.

[46] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168. ACM, 2011.

[47] T. Peng. Analysis of energy utilization in 3d printing processes. *Procedia CIRP*, 40:62–67, 2016.

[48] Pronterface. Pronterface Website. `http://www.pronterface.com/`. Accessed: 2016-11-8.

[49] V. J. Reddi, A. Settle, D. A. Connors, and R. S. Cohn. Pin: a binary instrumentation tool for computer architecture research and education. In *Proceedings of the 2004 workshop on Computer architecture education: held in conjunction with the 31st International Symposium on Computer Architecture*, page 22. ACM, 2004.

[50] G. Strano, L. Hao, R. Everson, and K. Evans. A new approach to the design and optimisation of support structures in additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 66(9-12):1247–1254, 2013.

[51] J. Suwanprateeb. Improvement in mechanical properties of three-dimensional printing parts made from natural polymers reinforced by acrylate resin for biomedical applications: a double infiltration approach. *Polymer international*, 55(1):57–62, 2006.

[52] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos. Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1458–1472, 2008.

[53] F. A. Thabtah, P. Cowling, and Y. Peng. Mmac: A new multi-class, multi-label associative classification approach. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 217–224. IEEE, 2004.

[54] Thingverse. Digital Designs for Physical Objects. `http://www.thingiverse.com/`. Accessed: 2016-05-22.

[55] V. Tiwari, S. Malik, A. Wolfe, and M. T.-C. Lee. Instruction level power analysis and optimization of software. In *Technologies for wireless computing*, pages 139–154. Springer, 1996.

[56] Ultimaker Inc. Ultimaker 2 Go. `https://ultimaker.com/en/products/ultimaker-2-go`. Accessed: 2016-05-22.

[57] Ultimaker Inc. Ultimaker 2 Go Components Specification. `https://github.com/Ultimaker/Ultimaker2`. Accessed: 2016-05-22.

[58] S. Vinodh, G. Sundararaj, S. Devadasan, D. Kuttalingam, and D. Rajanayagam. Agility through rapid prototyping technology in a manufacturing environment using a 3d printer. *Journal of Manufacturing Technology Management*, 20(7):1023–1041, 2009.

[59] N. Volpato, R. Nakashima, L. Galvão, A. Barboza, P. Benevides, and L. Nunes. Reducing repositioning distances in fused deposition-based processes using optimization algorithms. In *High Value Manufacturing: Advanced Research in Virtual and Rapid Prototyping: Proceedings of the 6th International Conference on Advanced Research in Virtual and Rapid Prototyping, Leiria, Portugal, 1-5 October, 2013*, page 417. CRC Press, 2013.

[60] G. Von Laszewski, L. Wang, A. J. Younge, and X. He. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–10. IEEE, 2009.

[61] S. Walls, J. Corney, A. Vasantha, and G. Vijayumar. Relative energy consumption of low-cost 3d printers. In *12th International Conference on Manufacturing Research*, 2014.

[62] L. Wang, G. Von Laszewski, J. Dayal, and F. Wang. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 368–377. IEEE, 2010.

[63] Wikipedia. G-code. `https://en.wikipedia.org/wiki/G-code`. Accessed: 2016-11-8.

[64] T. Wohlers. *Wohlers report 2016*. Wohlers Associates, Inc, 2016.

[65] C. Wright, A. Buchan, B. Brown, J. Geist, M. Schwerin, D. Rollinson, M. Tesch, and H. Choset. Design and architecture of the unified modular snake robot. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4347–4354. IEEE, 2012.

[66] C. Xu, F. X. Lin, Y. Wang, and L. Zhong. Automated os-level device runtime power management. *ACM SIGARCH Computer Architecture News*, 43(1):239–252, 2015.

[67] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010.