JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

Thermally Driven Inversion of Nanomagnetic Logic

von

Timo Pulch

Master Thesis in Physics
submitted to the Department of Physics, Mathematics and Computer Science
(FB 08)
of Johannes Gutenberg-Universität Mainz
on 15th of April 2020

First referee: Dr. Karin Evershor-Sitte
Second referee: Prof. Dr. Mathias Kläui

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Mainz, den 15. April 2020

Timo Pulch
TWIST
Institut für Physik
Staudingerweg 7
Johannes Gutenberg-Universität D-55099 Mainz
tpulch@students.uni-mainz.de

Abstract

In this thesis, we investigate the influence of thermal fluctuation on nanomagnetic logic within the macrospin limit. The nanomagnets that form the logic are modeled as uniaxial macrospins, which are dipolarly-coupled by means of stray fields. The dynamics of these systems are governed by the Landau-Lifschitz equation (LLG).
By arranging them into so-called majority gates and using clocking mechanisms, we are able to imitate the behavior of the AND, OR, NAND, and NOR gates. To control the dynamics of these gates, we use Spin-Transfer torques (STT). When connecting these structures to a more complex circuit, the half adder, we show that the correct behavior of the resulting circuits cannot be guaranteed, because the the states that fullfill the logic of the circuit are not necessarily ground states of the dipolar energy.
To solve this problem, we extend the model by a thermal fluctuation field, which allows us to get closer to applications and, at the same time, allows us to overcome metastable states. We will model the thermal fluctuations as Gaussian white noise, which allows us to extend the LLG to a Langevin equation. The resulting stochastic differential equation of the system is solved by a numerical solver, the Heun scheme.
By adding the thermal fluctuations of the magnetization, we are able to study the reversal of the operating direction of the majority gates. We show that it is possible to predict with high probability states that satisfy the respective truth table of the gate as long as their dipolar energy is low compared to the other realizable states.
As the main result, we propose a NAND gate whose ground states are degenerate from dipolar energy. Due to the degeneration, we are able to invert the operation direction of the gate, whereby the gate fluctuates with approximately the same probability into the logical desired states. Furthermore, we will investigate the possibilities of interconnections between these gates.

Contents

1. Introduction

Nowadays, logical gates surround us in every situation, as they are the basis of all modern electrical devices. The demand for modern and innovative concepts is high due to the progressive digitalization of everyday life. The rising cost of rare earth, especially silicon, is also contributing to the search for alternatives to conventional transistor-based gates.
As an alternative to ordinary complementary metal-oxide-semiconductor (CMOS), promising candidates are gate based on magnets and spintronics [1, 2].
Among the promising candidates for beyond CMOS devices, there is NML, which is operating with ultra-low energy dissipation [3]. The advantage of magnetic devices is that they are not charge-based, are intrinsically radiation-hard and can, therefore, act as both computing and storage devices.
In principle, the time in which data can be stored on such devices is determined by the internal energy barrier (ΔE) of a magnet. Typically the main contribution to the barrier is given by the shape anisotropy. For a barrier of 40 kT, non-volatile storage for one year is assumed, whereas, for a barrier of 60 kT, retention times are almost infinite [4, 2]. Furthermore, the non-volatile nature of NML is allowing, in theory, the realization of ultra-high-density computing systems.
The functionality of NML circuits is based on the dipolar coupling between neighbor particles [5]. Whereby the binary information is in NML encoded into the magnetization direction of nanomagnets. Information is propagated from one or more input magnets through the circuit as antiferromagnetic and ferromagnetic coupling [6]. The relative position and geometry of nanomagnetic particles have a significant influence on the functionality of the NML circuit. In Refs. [7, 8] suitable geometries have been shown experimentally for in-plane coupling nanomagnets. Out-of-plane solutions have been proposed in Ref. [9, 10] as well as NML which consists of magnetic tunnel junctions (MTJs) [11].
Experiments proved that suitable materials for NML devices are cobalt–platinum (Co/Pt) multilayers and permalloy [12, 3].The size of the nanomagnets ranges from a few nm to hundreds of nm [13, 4]. However, the shrinking of the volume is limited by the thermal stability of the magnets [13]. In 2017, Camsari et al. showed a logic memory device based semiconductor and nanomagnets device, which is able to perform boolean logic and is invertible [14]. While inverting the operating direction the output is kept at a constant value, and the network fluctuates among all possible inputs that are consistent with that output. The inversion is a unique property which conventional logic is not capable to.
To briefly explain inversion of logic, we take a look at the binary states and the AND gate. The binary states are usually called 0 and 1, which are realized in technical

applications by two different voltage values high and low.

Input A	Input B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1.1.: Truth table of an AND gate.

The AND gate processes two inputs A and B to an output O according to the logical operation of the conjugation $\wedge$, as shown in 1.1. With a conventional COMS gate, this would mean that there is only a 1 at the output as long as there are also an 1 at each input.
As a consequence, we speak of deterministic systems, since each combination of inputs is assigned a specific output. Moreover, this is the crux of the matter if on is thinking about inverting this logic because, in this case, the output state 0 can have three different states: 0-0, 0-1, 1-0. As a consequence of the inversion, it follows immediately that the system is no longer deterministic but stochastic.
In this work, we will show that it is possible to model nanomagnetic logical gates that can imitate conventional gates [2, 5, 15]. Further, we will show how to invert them. All numerical calculation are numerically solved on a GPU.
This thesis is divided into five parts. The first part is a short introduction to micromagnetics . The second part is an introduction to nanomagnetic logic. The third and fourth parts deal with stochastic differential equations to model thermal fluctuations and how to solve them numerically. In the final part, we discuss approaches to invert the operation direction of the gate.

2. Theory

This work is based on the idea of combining magnetic particles to logical gates, by exploiting the physics of stray fields. In this chapter, we will discuss the fundamental ideas for modeling Nanomagnets. In the following section, we will discuss the energy terms used to model magnets on the *nm* scale.

2.1. Fundamental Assumptions

The fundamental assumption of micromagnetics is the so-called continuum approximation [16]. It assumes that the temperature is well below the Curie temperature T_C so that the length of the local magnetization $\mathbf{M}(\mathbf{x})$ is constant and equal to the saturation magnetization $M_s = |\mathbf{M}(\mathbf{x})|$ of the ferromagnetic material. Furthermore, it is assumed that the angle between the magnetization and any axis can be approximated as a continuous function of the position. Thus the magnetization $\mathbf{M}(\mathbf{x})$, the state of the ferromagnetic system, can be described unitless normalized vectotr $\mathbf{m}(\mathbf{x}) = \mathbf{M}(\mathbf{x})/M_S$ [16].
Form a thermodynamic perspective, the Gibbs free energy of a magnetic system can be expressed as

$$G(\mathbf{H}, T) = F - \mu_0, \mathbf{M} \cdot \mathbf{H} \tag{2.1.1}$$

where $\mathbf{M}$ is the magnetic moment, $\mathbf{H}$ the external magnetic field, $F = U - TS$ the Helmholtz free energy, U is the internal energy of the system, S is the entropy of the system and T the temperature [17]. The equation of state of the system for the conjugate work variables $\mathbf{H}$ and $\mathbf{M}$ is given by

$$\mathbf{H} = \frac{1}{\mu_0}\left(\frac{\partial F}{\partial \mathbf{M}}\right)_T \tag{2.1.2}$$

and

$$\mu_0 \mathbf{M} = \left(\frac{\partial G}{\partial \mathbf{H}}\right)_T. \tag{2.1.3}$$

Regarding the ferromagnetic system, the contributions to the Gibbs Free energy are given by the exchange energy, the dipolar energy, the magnetocrystalline anisotropy energy, the magnetostatic energy, the Zeeman energy in an external field and the magnetoelastic energy [18].
Furthermore, from a theoretical point of view, magnetoelasticity is caused by magnetostriction and can be expressed in the same form as anisotropy. Since the anisotropy

constants are taken from experiments, the contributions of magnetostriction are already included [19] .

2.1.1. Exchange Energy

The exchange interaction is derived from the Coulomb interaction and is a spin selective, quantum mechanical effect that only occurs between identical particles. The origin of spin selectivity is found in the Pauli principle, which states that two electrons with parallel or antiparallel spins behave differently, although the fundamental interaction is the same. The difference lies in the antisymmetry or symmetry of the wave function. This means, for example, that two electrons with parallel spins cannot be at the same location.
In this regard, Heisenberg expressed the energy of this interaction in the form

$$\mathcal{H} = -\sum_{i,j} J_{ij}\mathbf{S}_i\mathbf{S}_j, \tag{2.1.4}$$

where J_{ij} is the exchange integral, and $\mathbf{S}_i$ are the spin operators [20, 21]. The exchange integral describes the overlap of the wavefunctions [20, 21]. In the case $J_{ij} > 0$, adjusted spins align parallel, and we are talking of a Ferromagnet. For $J_{ij} < 0$ we are speaking of an antiferromagnet where the spins are aligning antiparallel.
Since the wavefunction decreases rapidly with the distance between the atoms, we assume that the interactions occur for nearest neighbors, and we can rewrite J_{ij} to J for all nearest neighbors

$$\mathcal{J}_{ij} = \begin{cases} J & \text{if } i,j \text{ are neighbors} \\ 0 & \text{else.} \end{cases}. \tag{2.1.5}$$

By assuming an isotropic system the classical vectors can replace the spin vectors, and the angle is replacing the dot product such that the exchange energy changes to

$$E_{ex} = -JS^2 \sum_{<i,j>} \cos\phi_{i,j}. \tag{2.1.6}$$

By assuming that the angle $\phi_{i,j}$ is small, we can expand the cosine as a Taylor series and utilize the micromagnetics assumption that $\mathbf{m} = \mathbf{M}/M_S$ is a continuous variable. The angle is given by

$$|\phi_{i,j}| = |\mathbf{m}_i - \mathbf{m}_j| \approx |(\mathbf{r}_i \cdot \nabla)\mathbf{m}|, \tag{2.1.7}$$

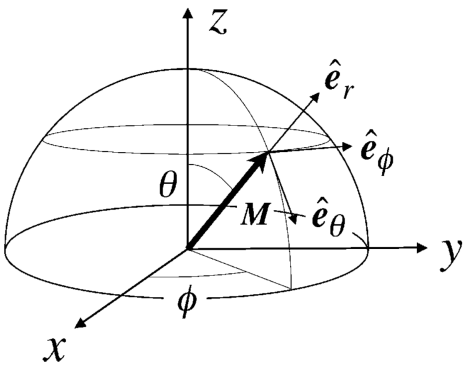
where $\mathbf{r}_i$ is the position vector from lattice point i to j.
The energy for a magnetic system is obtained by summing up contributions from different distance vectors $\Delta\mathbf{r}_i$ depending on the crystal structure and integration over the volume

$$E_{ex} = \int_V A[(\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2]dV. \tag{2.1.8}$$

The $A = \frac{JS^2c}{a}$ is the so-called exchange constant, where a is the distance between nearest neighbors and c = 1,2,4 for simple cubic, body-centered cubic, and face-centered structure [22]. The exchange length is the typical length scale below which atomic exchange interactions dominate typical magnetostatic fields. The exchange length is a material-specific parameter which is described by the exchange constant $l_{ex} = \sqrt{2A/\mu_0 M_S}$.
Since we are interested in modeling materials with a size smaller then the typical exchange length, we can assume that the material occurs in a monodomain state where $E_{ex} = 0$. The typical order of the exchange length for magnetic materials such as Fe, Co, Ni, and Permalloy is $5 - 50\ nm$, which is within the macrospin limit [23, 5].

2.1.2. Magnetocrystalline Anisotropy

The Heisenberg model is isotropic and independent of the spatial direction of the applied field. In real magnets, however, different behavior is observed, where the magnetization shows a preferred direction due to the crystalline structure. This magnetocrystalline anisotropy is caused by the relativistic spin-orbit interaction of electrons. The electron orbitals are coupled to the lattice of the crystallographic structure, and the resulting spin-orbit interaction with the spins causes the latter to align themselves preferentially along well-defined crystallographic axes. The resulting preferred orientation of the magnetization in real systems is called the easy axis. On the other hand, there are well-defined directions in which the magnet is hard to magnetize, the so-called hard axes.
In principle, magnetocrystalline anisotropy can be derived from first principles, starting from the Dirac equation, but it is usually easier to fall back on the known crystal symmetries. With the help of power laws, phenomenological expressions can be derived from these, which take into account the coefficients from the experiment [24].



,

Figure 2.1.: Spherical coordinates: definition of angles. The usual Cartesian magnetization components read: $m_x = \cos\phi\ \sin\theta$, $m_y = \sin\phi sin\ \theta$, $m_z = \cos\theta$

Although the magnetocrystalline anisotropy is typically small compared to the exchange interaction, the magnetocrystalline anisotropy determines the direction in which the magnetic moments are aligned because the exchange interaction minimizes the energy whenever the magnetic moments are parallel, regardless of their actual direction in space. In the case of modeling uniaxial systems, which we regard as hexagonal crystals, the anisotropy energy is a function of only one parameter, namely the angle Θ between the magnetization and the x-axis or easy axis pointing in the z-direction. The magnetic orientation with respect to the azimuthal angle ϕ describes the energy required to rotate the magnetization in a plane perpendicular to the z-axis according to the anisotropy, as shown in Figure 2.1. Due to the symmetry with respect to the base plane, odd powers of $\cos(\Theta)$ can be omitted in the power series expansion for the anisotropy energy. Thus, the energy takes form

$$E_{ani} = \int_V [-K_1 cos^2\theta(\mathbf{r}') + K_2 cos^4\theta(\mathbf{r}')]dV = \int_V [-K_1 m_z^2(\mathbf{r}') + K_2 m_x^2(\mathbf{r}')]dV,$$

where parameter K_1 and K_2 have the unit of energy density and are depending on the composition and the temperature. Previous experimental research has shown that higher-order terms are mostly small and can be omitted, even the K_2. If $K_1 > 0$, then the x-axis is an easy axis where the alignment of the magnetization minimizes the energy. On the other hand, if $K_1 < 0$, we no longer get an easy axis, but an easy plane perpendicular to the z-axis [25].

2.2. Shape Anisotropy

Domain formation can be understood as resulting from the interaction of dipole fields and the shape of a magnetic system. The anisotropy energy will further be influenced by the actual shape of the sample due to the dipolar interaction between the spins, with the magnetization preferably lying in the crystal plane to minimize the stray fields, which leads to domain wall formation. This is called shape anisotropy [19].
In the case of an ellipsoid shape, the energy contribution of both shape and magnetocrystalline anisotropies can be written as

$$E_{ani} = \int_V [N_1 m_x^2(\mathbf{r}) + N_2 m_y^2(\mathbf{r}) + N_3 m_z^2(\mathbf{r})]dV. \tag{2.2.1}$$

The N_i coefficients are the demagnetization coefficients of the body determined by its shape. If we consider our magnetic system to have uniaxial anisotropy with the z-axis as easy axis and the x-axis as hard axis and assume the system as a monodomain, we can simplify this term to

$$E_{ani} = K_e\left(m_z^2 - \frac{K_h}{K_e} m_x^2\right) = K_e(m_z^2 - D m_x^2), \tag{2.2.2}$$

where K_e is the easy axis anisotropy and K_h the hard axis anisotropy, D is the dimensionless ratio of hard- and easy axis anisotropies.

2.3. Dipole Interaction

While the exchange interaction almost exclusively determines the behavior of the interatomic interaction between neighboring atoms, magnetic moments appear to each other at long distances as magnetic dipoles. Magnetic dipole–dipole interaction refers to the direct interaction between two magnetic dipoles and is about three magnitudes smaller than the exchange coupling and decays with the distance according to the power law $1/r^3$ [19].
In contrast to the exchange interaction, which tries to align the magnetic moments in parallel, in the dipolar interaction, the moments are aligned anti-parallel to minimize the energy. The interplay between the shape of the magnetic sample and the dipolar interaction leads to domain formations. In a domain, it is expected that all magnetic moments are oriented in the same direction, while the direction of orientation can change from one domain to another. In Mircomagnetics, the dipolar interaction can be written within the continuum approximation in the following fashion [26]

$$E_{dip} = \int_V \frac{1}{2}\mathbf{H}_d(\mathbf{r}) \cdot \mathbf{M}(\mathbf{r}) dV, \tag{2.3.1}$$

where $\mathbf{H}_d$ is the demagnetization field. The demagnetization field is the sum of all dipolar interactions between the magnetic moments at position $\mathbf{r}$ and all others at $\mathbf{r}'$. As mentioned before, due to domain formation, we need to take shape and the surface of the magnetic system into consideration:

$$\mathbf{H}_d(\mathbf{r}) = \nabla\left(\int_V \frac{\nabla \cdot \mathbf{M}(\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} dV' - \oint_S \frac{\mathbf{n}(\mathbf{r}') \cdot \mathbf{M}(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} dS \right). \tag{2.3.2}$$

2.3.1. Point-dipole Approximation

Since we are interested in studying monodomain magnets, we do not take the dipolar interaction into account for a single magnet. However, we have a particular interest in studying the dipolar coupling behavior of such monodomain magnets with each other. Therefore the dipole field of small magnetic magnets can be estimated using the well known point-dipole approximation [27]. Assuming the magnetic dipole element as infinitesimally small we can rewrite 2.3.2 as

$$\mathbf{H}_d(\mathbf{r}) = \frac{1}{4\pi r^3}\left[3\frac{(\mathbf{M} \cdot \mathbf{r})\mathbf{r}}{r^2} - \mathbf{M} \right]. \tag{2.3.3}$$

Due to the long-range nature of the dipolar interaction, the calculation is a very demanding task from a computational point of view, since the number of calculations to be performed is proportional to the number of moments squared of N^2.

2.4. Zeeman Energy

So far, we discussed how the magnetic moments interact with each other and with the lattice, but we did not discuss the interaction with an applied field **H**. From observation, it is known that within a ferromagnetic system, the magnetic moments will align along the applied field. This interaction is known as Zeeman interaction and has the form

$$E_Z = -\mu_0 M_S \int_V \mathbf{M} \cdot \mathbf{H} dV. \tag{2.4.1}$$

2.5. Macrospin Model (Stoner-Wohlfarth model)

The model we have discussed so far allows us to describe magnetic samples whose magnetization is composed of several domains. However, we are interested in samples whose magnetization consists of a single domain or monodomain. Stoner and Wohlfarth have introduced such a model, which is nowadays known under their name [28]. In principle, it is a simplification of micromagnetics, which is widely used to describe small magnetic particles [2, 22, 29].
The Stoner-Wohlfarth model (SWM) is an approximation for particles with strong exchange interaction within macrospin-limit (1-50 nm) [23]. In this range, the magnetization will be uniform due to the exchange interaction and perform a coherent rotation.
The uniformity of the magnetization leads to a constant contribution to the exchange term of the Gibbs free energy, which does not change the motion of the magnetization vector. With respect to the model discussed earlier, we assume that in the SWM the magnetization corresponds to the saturation magnetization and rotates around an applied magnetic field **H**. Furthermore, we assume an ellipsoidal shape of the ferromagnetic sample with uniaxial symmetry.
The total energy landscape of the Stoner-Wohlfarth Macrospin is then given by

$$E(\mathbf{m}) = K_h m_x^2 - K_e m_z^2 - \mu_0 M_S V \mathbf{m} \cdot \mathbf{H}_{ext}, \tag{2.5.1}$$

where $V = a^2 l$ is the volume of the ellipsoid, $K_e = (1/2)\mu_0 M_S V H_k$ is the easy axis anisotropy and $K_h = \mu_0 M_S^2 V$ is the hard-axis anisotropy. The $\mu_0 H_k$ is known as the Stoner-Wohlfarth switch field and fixes the coercivity of the magnetic model.
The SWM or macrospin model will build the foundation of our elementary magnetic bits. Since we are interested in doing conventional binary computation, we need to define two states, which we can interpret as '0' and '1' or current-'ON' and '-OFF'.
Since the magnetization is defined with respect to the easy axis, the natural choice of binary states also falls on these axes. Therefore we make the arbitrary choice of interpreting bits which point in $+z - direction$ as '1' and bits which point in $-z - direction$ as '0'.

2.6. Landau-Lifshitz-Gilbert Equation

In 1935 Landau and Lifshitz proposed a phenomenological equation which describes the dynamics of magnetic moments in a ferromagnetic material [30].
The idea underlying the equation is that the magnetic moment is related to the torque via the gyromagnetic moment

$$l = \frac{\mu_0 \mathbf{M}}{\gamma}, \gamma = \frac{\mu_0 g|e|}{2m_e} = 2.210173 \cdot 10^5 \frac{m}{As}, \tag{2.6.1}$$

where $g \approx 2$ is the Landé factor, e the elementary charge, and m_e the mass of an electron. It is known from Newtonian mechanics that the torque is defined as the time derivative of the angular momentum $\tau = d\mathbf{L}/dt$ and as the force acting on the lever arm vector

$$\tau = \mathbf{r} \times \mathbf{F}. \tag{2.6.2}$$

It naturally follows to express these quantities by there magnetic counterparts, and one will find the equation

$$\frac{d\mathbf{M}}{dt} = -\gamma \mathbf{M} \times \mathbf{H}. \tag{2.6.3}$$

This equation can be expressed in terms of the magnetization, where the applied field $\mathbf{H}$ needs to be replaced by an effective field $\mathbf{H}_{\text{eff}}$. Thus, we obtain

$$\frac{d\mathbf{M}}{dt} = -\gamma \mathbf{M} \times \mathbf{H}_{\text{eff}}. \tag{2.6.4}$$

This is the so-called Landau-Lifshitz equation describes the undamped precession of the magnetization vector $\mathbf{M}$ around the direction of the applied effective field $\mathbf{H}_{\text{eff}}$ with the Larmor frequency of $\omega = \gamma \mu_0 |\mathbf{H}_{\text{eff}}|$. In the effective field, $\mathbf{H}_{\text{eff}}$ our macrospin model comes into play. The effective field for the macrospin model can be derived by

$$\mathbf{H}_{\text{eff}} = -\frac{1}{\mu_0 M_S V} \nabla E(\mathbf{m}) = -H_K [K_h/K_e m_x \hat{\mathbf{x}} - m_z \hat{\mathbf{z}} - \mathbf{h}], \tag{2.6.5}$$

where $H_K = 2K_e/(M_S V \mu_0)$.
However, experiments show that this equation does not reflect reality since the relaxation of magnetization is completely neglected. Gilbert added a phenomenological damping term to the LL equation to describe the relaxation process [31]

$$\frac{\alpha}{M_S} \mathbf{M} \times \frac{d\mathbf{M}}{dt}, \tag{2.6.6}$$

with a dimensionless damping parameter α. By adding this term to the LL, Gilbert proposed the Landau-Lifshitz-Gilbert equation, which is nowadays widely used to describe the magnetization dynamics in ferromagnetic materials.

$$\frac{d\mathbf{M}}{dt} = -\gamma' \mathbf{M} \times \mathbf{H}_{\text{eff}} - \gamma' \frac{\alpha}{M_S} \mathbf{M} \times (\mathbf{M} \times \mathbf{H}_{\text{eff}}), \tag{2.6.7}$$

where $\gamma' = \gamma/(1+\alpha^2)$. The resultant dynamics is such that the magnetization is precessing around the effective field until the damping term finally aligns the magnetization in the direction of the field.
The effective field $\mathbf{H}_{\text{eff}}$ will be rescaled by dividing both sides by $\gamma H'_K$. Thus we can rewrite 2.6.7 as

$$\frac{d\mathbf{m}}{dt} = \mathbf{m} \times \mathbf{H}_{\text{eff}} - \alpha \mathbf{m} \times (\mathbf{m} \times \mathbf{H}_{\text{eff}}) \tag{2.6.8}$$

in micromagnetics. In order to obtain physically relevant quantities, it is necessary to introduce a timescale in which the system is evolving. We will call it the natural timescale $\tau = \gamma' \mu_0 H_k t$ [22].
Further experiments have shown that the damping is caused by the interaction of the magnetic moment of the electron with the crystal lattice [32]. Typical spintronic devices have a damping α that is of the order of 10^{-2} [33, 34].

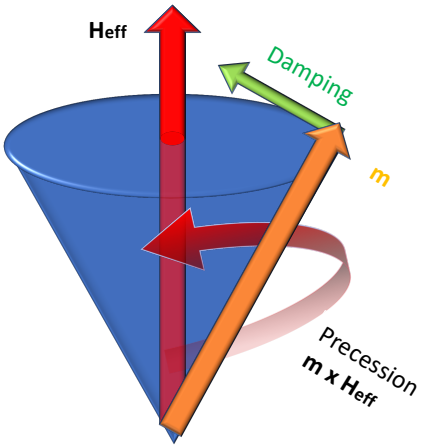


Figure 2.2.: The terms of the Landau–Lifshitz–Gilbert equation: precession (red),magnetization (orange), effective field $\mathbf{H}_{eff}$ (red) and damping (green).

2.6.1. Slonczewski Spin Torque

Slonczewski and Berger proposed the existence of spin-transfer-torques in 1996 [35]. These torques are generated by spin-polarized currents which act on magnetic moments, whereby a transfer of the spin angular momentum from the current to the magnetic moment takes place. This transfer ensures that the angular momentum is conserved in the system.
Typically, the spin-transfer torque (STT) is an effect in which the orientation of a

magnetic layer in a MTJ or spin valve can be changed by means of a spin-polarized current. In general, electric currents are unpolarized. By passing a current through a thicker magnetic layer, the so-called fixed layer, a spin-polarized current can be generated. The resulting angular moment in this current can be transferred to a second thinner layer, the so-called free layer, while passing through it, changing its magnetization orientation. The generated torque can be used to excite vibrations in the magnetization or even reverse the magnetization orientation of the magnet, as shown in Figure 2.3. Normally, this effect occurs in devices of nm size [36].
The non-conservative term derived by Berger and Slonczewski takes the form:

$$\Gamma_S = -\gamma j[\mathbf{m} \times (\mathbf{m} \times \mathbf{n}_p)], \tag{2.6.9}$$

in the macroscopic equation of motion. The spin-torque is assumed to be brought by a flow of current polarized in the direction of $\mathbf{n}_p$, proportional to the spin-angular momentum per unit time $j = (\hbar/2e)\nu J/\mu_0 M_S H_K d$, where $\nu = (J_\uparrow - J_\downarrow)/(J_\uparrow + J_\downarrow)$ is the spin-polarization factor of the incident current J and thickness d of the magnetic free layer.

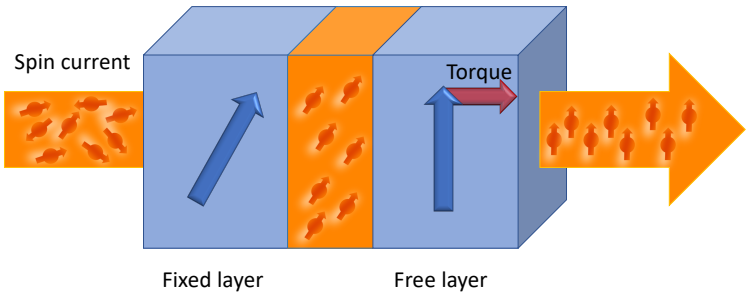


Figure 2.3.: An unpolarized current is injected into a layer in which the magnetization is fixed. When passing through the layer, the current is polarized parallel to its magnetization of the layer. A non-magnetic layer separates the magnetic layer from another. As it enters the layer, the spin-polarized current will apply a torque to the magnetization of the free layer until the magnetization of the two successive layers is aligned in parallel.

In addition, another torque can also arise, the field-like torque

$$\Gamma_S = -\gamma\sigma j(\mathbf{m} \times \mathbf{n}_p), \tag{2.6.10}$$

where σ represents the relative strength of the "field like" torque compared to the STT [36, 37]. This arises from a non-equilibrium spin-accumulation in the magnetic layer. The torques cause the magnetization of the layer to precess about the direction of the spin-current polarization $\vec{n}_p$. Since in Refs. [38, 39] it has been found that $\sigma << 1$ we

will neglect this contribution in the following.
Further works in this field showed that these currents could be used to manipulate and reverse the magnetic moments. In this work, we will use the spin currents to 'kick' the magnetic moments so that they correspond to the logic of the system. To be precise, they will be kicked in the direction of the easy axis z. The resulting Landau-Lifshitz-Gilbert-Slonczewski equation (LLGS) takes the form

$$\frac{d\mathbf{m}}{dt} = \mathbf{m} \times \mathbf{h}_{\text{eff}} - \alpha \mathbf{m} \times (\mathbf{m} \times \mathbf{h}_{\text{eff}}) - \alpha I \mathbf{m} \times (\mathbf{m} \times \hat{\mathbf{n}}_p), \tag{2.6.11}$$

where $I = j/(\alpha \mu_0 M_S H_K)$.

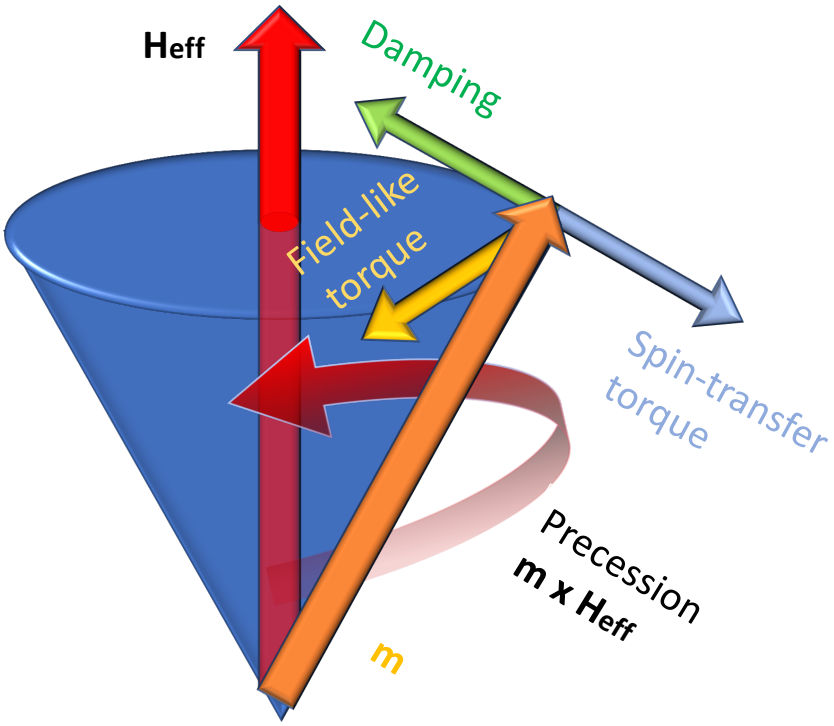


Figure 2.4.: Contributions to the dynamics underlying the LLGS discussed in 2.6.10 and 2.6.11. The damping term wants to align the magnetization $\mathbf{m}$ with the effective field $\mathbf{H}_{eff}$, while the STT is counteracting. The precession and the field-like torque are perpendicular to the STT and damping.

3. Building logic out of ferromagnetic

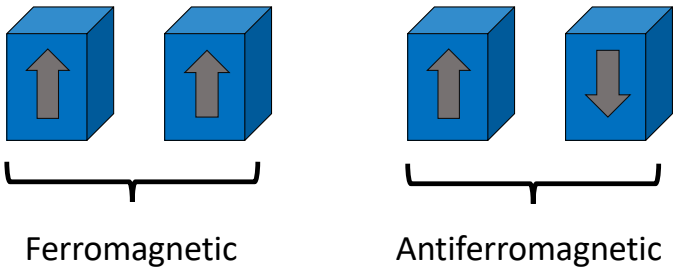


Figure 3.1.: Ferromagnetic and antiferromagnetic coupling of nanomagents.

In this chapter, we want to discuss how one can build logic out of nanomagnets. The fundamental idea of all these NMLs is that we can couple magnets either ferromagnetically or antiferromagnetically to design structures that are acting as gates. In the discussed NML, the binary information is encoded into the magnetization direction of single-domain nanomagnets.
The information is transferred through dipole-field coupling between neighboring nanomagnets and can be manipulated by the application of external magnetic field or spin torques. The simplest model of magnets that are capable of acting as bits is the SWM model, which we discussed in the previous section. Due to the elliptical shape, the magnets show uniaxial anisotropy. Thus they are bistable a prefer to magnetize parallel or antiparallel to there easy axis. This alignment enables us to encode information into the nanomagnets whereby we obtain a bit based on parallel or antiparallel magnetization. Several works already proved that in the macrospin limit, ($< 100\ nm$) the SWM is a suitable assumption to describe such NML systems [2, 4, 5].
In contrast to conventional logic, which operates with electrical voltages, there is a limited number of methods to manipulate the state of the input magnets. The most common method is to apply magnetic fields to the system.
This method leads in our case to several problems, which are rooted in the long-range nature of the dipole interaction. Since our systems are of the order of a few nm, it is difficult to apply magnetic fields locally to individual magnets without affecting adjacent magnets. Nevertheless, several works have been proven that it is possible to build gates in this fashion for larger nanomagnets [4, 15].
We will study NML device consisting of macrospins and use STTs to alter the magnetization of the input magnets, which we introduced in chapter 2. Similar logic has been presented in [11].

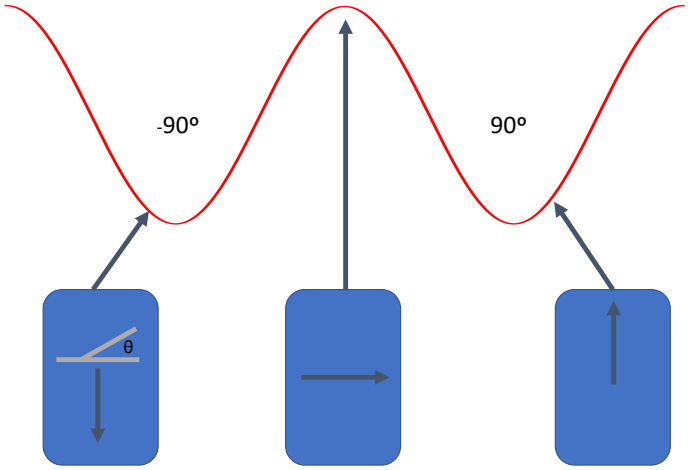


Figure 3.2.: The energy landscape due to the shape anisotropy of an elliptically shaped nanomagnet. Binary information is encoded along its easy axis.

3.1. Information Transfer

In order to transfer information from one area to another within the circuit, one needs some kind of cabling. The simplest wire that can be built for this purpose consists of a chain of identical single-domain magnets aligned perpendicular to their easy axis and next to a fixed bit that is parallel or anti-parallel to the easy axis.
The chain is aligned by an external magnetic field. By switching off the external field, the magnets within a chain will align antiferromagnetically along their easy axis due to the dipole interaction. This process is called 'clocking' and is just to modulate the energy barrier between magnetization states. Depending on the setup, either external magnetic fields or spin-transfer torques can serve as so-called 'clocking fields'. In this way, information can be transmitted from the start bit to the end bit as shown in Figure 3.3.
By modifying the $\mathbf{H_{eff}}$ in 2.6.5 we are able to describe this information propagation process. As previously mentioned, we need to include in the SWM the interaction which drives this process, the dipolar interaction. This can be done by writing the point-dipole approximation in the form of a coupling matrix

$$\mathbf{C}^{ij} = \frac{V^j}{4\pi r_{ij}^3} \begin{pmatrix} 3\hat{\mathbf{r}}_x^2 - 1 & 3\hat{\mathbf{r}}_x\hat{\mathbf{r}}_y & 3\hat{\mathbf{r}}_x\hat{\mathbf{r}}_z \\ 3\hat{\mathbf{r}}_y\hat{\mathbf{r}}_x & 3\hat{\mathbf{r}}_y^2 - 1 & 3\hat{\mathbf{r}}_y\hat{\mathbf{r}}_z \\ 3\hat{\mathbf{r}}_z\hat{\mathbf{r}}_x & 3\hat{\mathbf{r}}_z\hat{\mathbf{r}}_y & 3\hat{\mathbf{r}}_z^2 - 1 \end{pmatrix}, \tag{3.1.1}$$

where $\mathbf{r}_{ij}$ is the distance between two single-domain magnets and $\hat{\mathbf{r}} = (\hat{\mathbf{r}}_x, \hat{\mathbf{r}}_y, \hat{\mathbf{r}}_z)$ is the unit vector pointing from magnet i to j. V^j is the volume of the magnet j [5]. Then the dipolar field which $\mathbf{m}_i$ feels can be written as

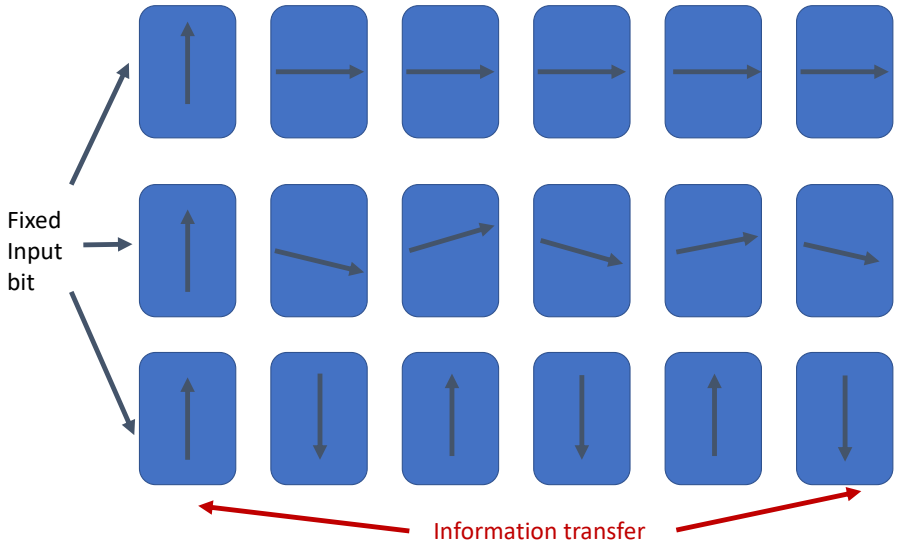


Figure 3.3.: A diagram showing how a magnetic 'wire' consisting out of nanomagnets. The wire can be used to transmit information from a point in one circuit to another. In the first row is a single input nanomagnet that has been fixed, and the other nanomagnets are in the chain that has been clocked along their 'hard' axis with an external magnetic field. The clocked field has been removed in the second row, and the magnets begin to fall back to their preferred axis of magnetization. The direction in which they fall is determined by the dipole Coupling between nearest neighbors. The chain will tend to align antiferromagnetically, and the The magnetization will, therefore, be a function of the input nanomagnet after the field in the last row has been completely removed.

$$\mathbf{H}_{\mathbf{d},i} = \sum_j \mathbf{C}^{ij}\mathbf{m}_j. \tag{3.1.2}$$

So that $\mathbf{H}_{\mathbf{eff}}$ in 2.6.5 will be

$$\mathbf{H}_{\mathbf{eff},i} = \mathbf{H}_{\mathbf{eff},i} + \mathbf{H}_{\mathbf{d},i}. \tag{3.1.3}$$

The extension of the external field allows us now to calculate the energy landscape of two adjusted nanomagnets, see Figure 3.4. By fixing the magnetization of one of them parallel to the easy axis, we observe that the energy maxima of the other nanomagnet is at the point where the magnetization is antiparallel and vice versa when the free spin is initialized perpendicular to the easy axis.

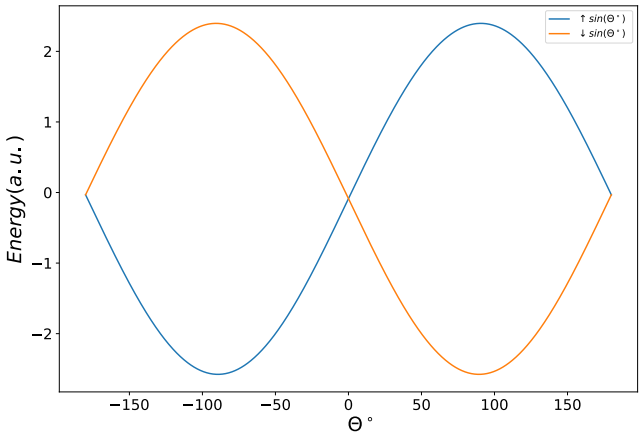


Figure 3.4.: Energy landscape of a free spin within a spinning chain consisting of two nanomagnets. The free spin is initialized perpendicular to the easy axis, whereas the fixed spins are parallel or antiparallel to the easy axis. The different orientations lead to a $180°$ shift in the energy.

3.2. NOT Gate

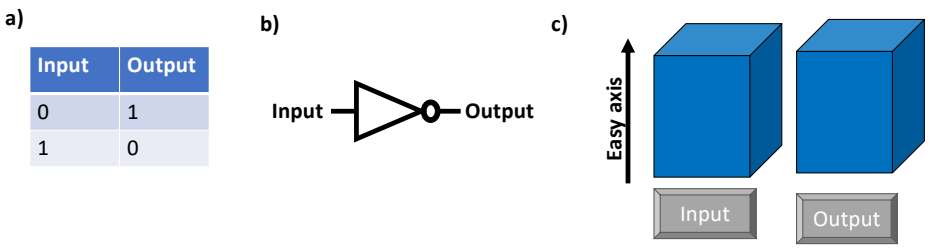


Figure 3.5.: a) Logical table of a NOT gate .b) Circuit diagram of a NOT gate. c) spin chain of two nanomagnets that mimics a NOT gate.

A spin chain can not only be used to propagate information, but it can also be used to invert or negate the information to be propagated. By the antiparallel arrangement of successive bits, each successive bit can be seen as a negation of the preceding one.

When comparing the nanomagnetic setup with conventional logic, one finds that non-gates fulfill this property. This means that every magnet within a spin chain serves as a non-gate within the NML. The circuit diagram, truth table, and magnet configuration are shown in Figure 3.5.

3.3. Fundamental Gates

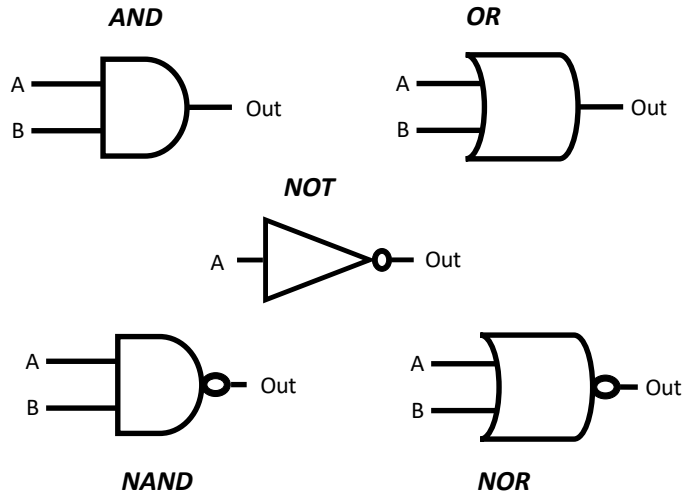
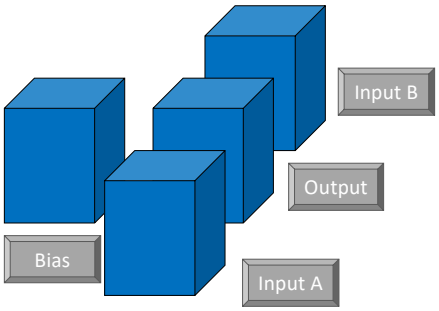


Figure 3.6.: Circuit symbols of AND, OR, NAND,NOR and NOT gates.

From an application-oriented perspective, complex circuits cannot be built from the non-gate. In this section, we will extend the spin chain shown in the previous section to gather more logical functionality.
By adding two more bits to the non-gate, a so-called majority gate is obtained that can function as either a NAND or NOR gate [3]. The gate design is shown in Figure 3.7.
The functionality of this gate design is realized by the majority function 3.3.1. The majority function is a function from n inputs to one output. In our specific case, a function of the two inputs A, B, and the Bias to the output. The value of the operation is false when $n/2$ or more arguments are false and true otherwise

$$\text{Majority}(p_1, ..., p_n) = \left[\frac{1}{2} + \frac{(\sum_{i-1}^{n} p_i) - 1/2}{n}\right]. \tag{3.3.1}$$

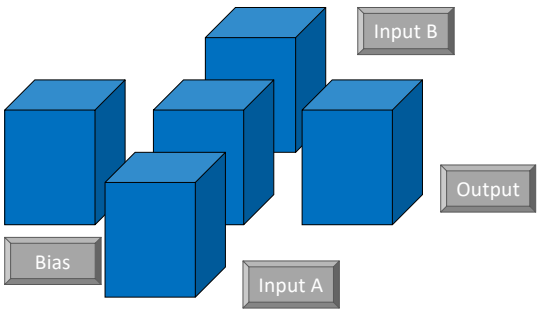
The majority gate consists of at least four nanomagnets arranged, as shown in Figure 3.7. In this arrangement, the magnets above and below the central one act as input,



Bias	Input A	Input B	Output	
0	0	0	1	
0	0	1	1	NAND
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	NOR
1	1	0	0	
1	1	1	0	

Figure 3.7.: Majority gate consisting of four nanomagnets that can act as NAND-,NOR gate.

the central magnet is the output. The magnetization orientation of the remaining magnet determines whether it is a NAND or NOR gate 3.7.
The functionality of the gate can be explained most easily by the following examples. If we assume that the inputs (A,B) and the Bias are pointing in the ↑ direction, we expect that the resulting stray fields will arrange the magnetization of the central magnet antiparallel ↓ according to the dipolar interaction to minimize the energy of the system as shown in Figure 3.9. We will call this configuration of Bias, Input A, Input B and Output ↑↑↑↓.
On the other hand, if we now let one of the input magnets points in the ↓-direction, we will still find that the majority of the surrounding magnets point in the ↑-direction, which will cause the central bit to point in the ↓-direction as well.
Only in the case that both inputs point in the ↓-direction we expect the magnetization of the output to point in the ↑-direction.
If we now identify the ↑-direction with '1' and the ↓-direction with '0', we see that in the cases shown in Figure 3.7, it is the operation of NOR gate. If the bias magnet is oriented in the ↓-direction, the operation of a NAND gate is observed. In Figure 3.7, all possible input/output combinations of the majority gates are listed.
Furthermore, this setup can easily be extended by adding an additional magnet at the right sight. Similar to the spin chain the information will get negated, and the gate acts as AND or NAND Gate, as shown in 3.8. This set of gates will build the foundation on which we can create more complex logical networks.



Bias	Input A	Input B	Output	
0	0	0	0	AND
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	OR
1	0	1	1	
1	1	0	1	
1	1	1	1	

Figure 3.8.: Majority gate consisting of four nanomagnets that can act as AND-,OR gate.

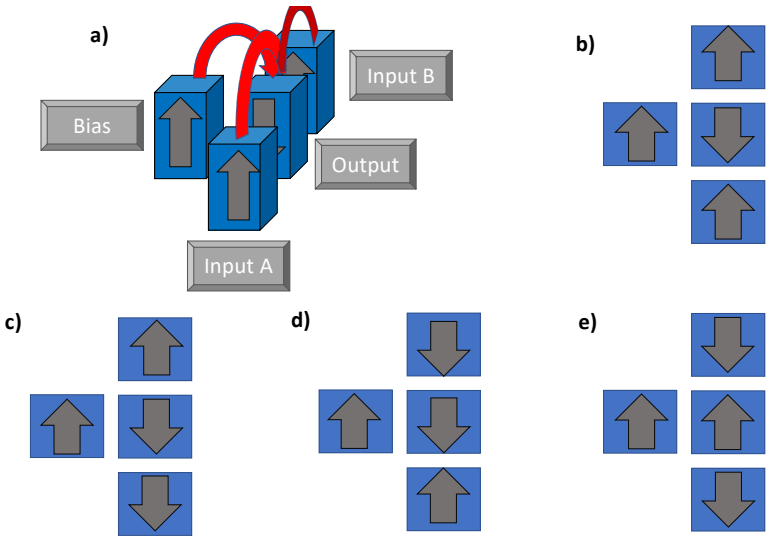


Figure 3.9.: NOR gate configuration of the majority gate: a)Anti ferromagnetic alignment of the output bit due to the dipolar fields of the surrounding Input A, B and the Bias. b) ↑↑↑↓. c) ↑↓↑↓. d) ↑↑↓↓. e) ↑↓↓↑.

3.4. Simulation of Fundamental Gates

Within the macrospin framework we are able to simulate such systems by solving the LLG with an ordinary differential equation (ODE) solver based on the Euler method. Previous work in these field have shown that suitable parameters for nanomagnets for NML are V = $60 \times 90 \times 5\ nm^3$, $M_S = 1000\ 000\ Am^{-1}$, $\alpha = 0.1$, $K_1 = 30000\ \frac{J}{m^3}$ [3]. Simulations have been performed on the scale of the natural time of the system, see Eq. 2.6.8. Usually, the size of a real-time step Δt is of the order of 10^{-12} to 10^{-11} seconds depending on the value of H_K and reads.

$$\Delta t = \gamma' H_K \Delta \tau. \tag{3.4.1}$$

In the presented simulations, a natural timestep of $\Delta\tau$ of 0.04 $s/\gamma' H_K$ has been used. The relaxation time of the magnetic system is about $1250 s/\gamma' H_K$ natural timesteps which means that the gate will relax towards an equilibrium state within a few ns.
Note that in the Stoner-Wohlfarth switching field, $H_K = 2K_e/(\mu_0 V M_S)$, the system-specific quantities are included besides the spacing of the nanomagnets.
All the majority gate simulations have been performed with the given parameters V = $60 \times 90 \times 5\ nm^3$, $M_S = 1000\ 000\ Am^{-1}$, $\alpha = 0.04$, $K_1 = 30000\ \frac{J}{m^3}$ and distance of 25 nm distance in between the dots. The natural timestep is given by this parameters corresponding to a realtime timestep of 84.4 ps.
The macrospins were initialized perpendicular to the easy axis, the z-axis, so that each magnetization points along the z-axis according to the direction of the clocking field. The STT alters the dynamics of the inputs A, B, and the Bias so that they fall into the preferred configuration. To do this, the influence of the STT must be several orders of magnitude greater than the dipolar fields.

3.4.1. NAND- and NOR-Majority Gate

Figure 3.10 shows a simulation in which all possible combinations of inputs A, B are run through according to the order ↓↓, ↓↑, ↑↓, ↑↑ first for the OR gate and then for the AND gate. Every $4.2125 ns$ the configuration of the currents acting on the inputs and bias magnets is changed. The change from one gate configuration to the other is realized by a change of the applied current that acts on the biasing magnet.
When comparing the states of the nanomagnets to the truth table 3.7, we observe that the dipolar fields alter the magnetization such that the setup mimics either the NAND- or NOR gate.

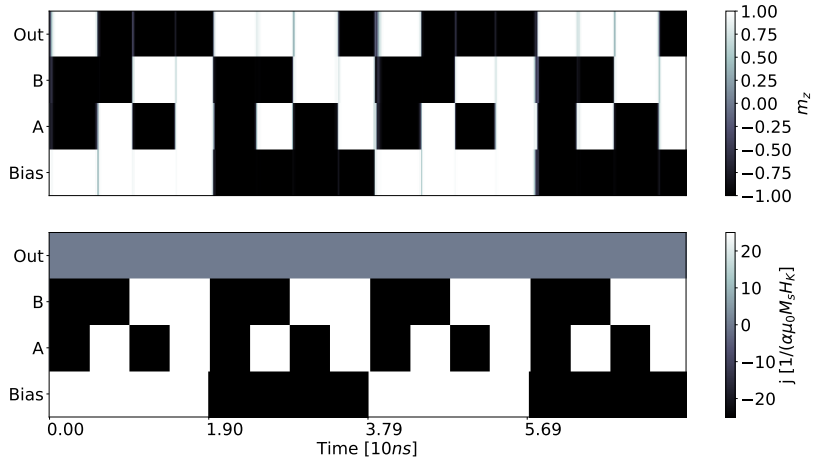


Figure 3.10.: Simulation of the majority gate that acts as a NAND/NOR gate. The current acting on the inputs is periodically changed to alter the magnetization of the inputs and the outputs. The gradient from white to black indicates the orientation of the applied current and magnetization.

3.4.2. AND- and OR-majority Gate

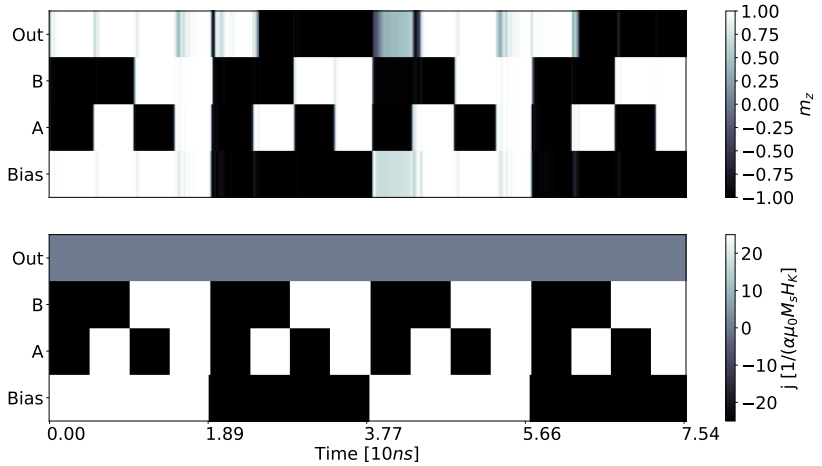


Figure 3.11.: Simulation of the majority gate that acts as an AND/OR gate. The current acting on the inputs is periodically changed to alter the magnetization of the inputs and the outputs. There is no clocking resetting the system in between the changing of the current.

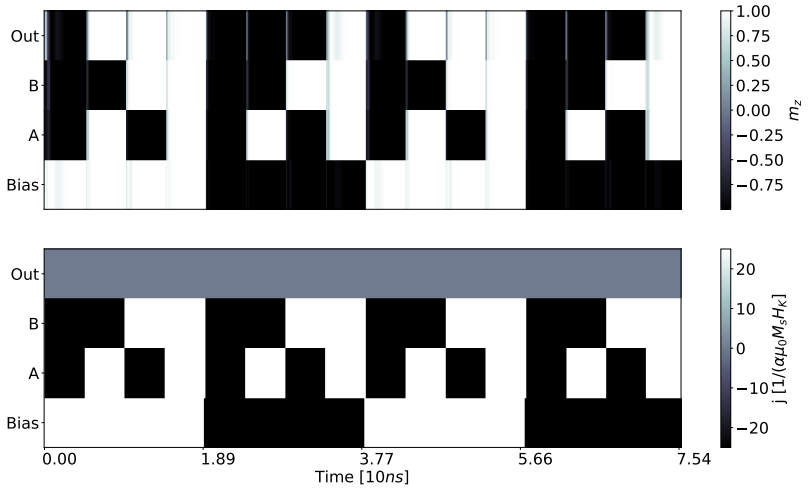


Figure 3.12.: Simulation of the majority gate that acts as an AND/OR gate. The current acting on the inputs is periodically changed to alter the magnetization of the inputs and the outputs. There is a clocking resetting the system in between the changing of the current each 4.2125 ns. We observe the verification of the truth table 3.8 in contrast to 3.11

In contrast to the NAND/NOR majority gate, it is not possible to verify the truth table 3.8 in the simulation of the AND/OR majority gate by changing the currents applied to the inputs and bias. The difference between the two gates is that the NAND/NOR gate has only one free spin, whereas the AND/OR gate has two free macrospins. This allows the free spins to get stuck in a metastable state while transiting from one state to the other. This observation has already been made before [4]. To understand this behavior, we interpret the bias and the two free magnets as a spin chain.
At this point, we can briefly discuss why the clocking field is mandatory for nanomagnetic chains. In the case of a defect group, which are configurations in which next nearest neighbor bits have not the same magnetic orientation $\uparrow x \downarrow$ (or $\downarrow x \uparrow$) the energy landscape is shown in Figure 3.13.
The energy landscape of the defect group shows that the magnetization of the central magnet does not prefer any direction since the stray field of the surrounding magnets cancels out. In comparison, Figure 3.13 also shows the case $\uparrow x \uparrow$ (or $\downarrow x \downarrow$), where the magnetization of the center nanomagnet favours the antiferromagnetic alignment.
By clocking the AND/OR gate whenever we change the applied currents, we observe in the simulation 3.12 that the truth table of an AND/OR can be verified.

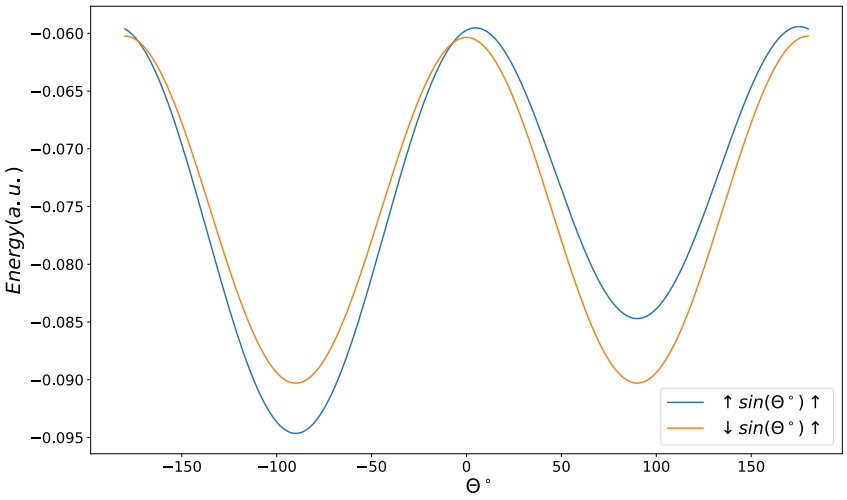


Figure 3.13.: Magnetization energy as a function of the angle for a nanomagnet at the critical field in the presence of two nearby neighbors. A nanomagnet that is part of a defect pair has the same energy profile as an isolated nanomagnet because the dipole fields of its nearest neighbors cancel out. The energy barrier for an antiferromagnetically aligned nanomagnet is increased by the nearest neighbor interactions.

3.5. Half adder

Input A	Input B	Carrier	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Figure 3.14.: Logical table of an half adder'

The fundamental gates NOR, AND, OR, NAND, and NOR, allows us to build complex circuits by connecting them. The circuit we want to investigate is the half adder. It takes two inputs and processes them into two outputs. The half adder allows us to add two single-digit binary numbers. The output S (sum) provides the right and the

output C (carry) the left digit of the result. The corresponding logic table is shown in Table 3.15.
The carry C is realized by an AND gate and the sum by an XOR gate. The XOR gate is a switching network consisting of an AND-, an OR- and a NAND gate. The circuit diagram of the half adder and the arrangement of interconnected majority gates that we proposed are shown in Figure 3.15. We see that the AND gate and the OR gate on the left-hand sight share some dots that originally belong to each of them to guarantee proper behavior. The same holds for the biasing bit of the NAND- and AND gate on the right-hand sight. This decision was made because it simplifies the setup and, more importantly, allows us to connect the gate with each other.

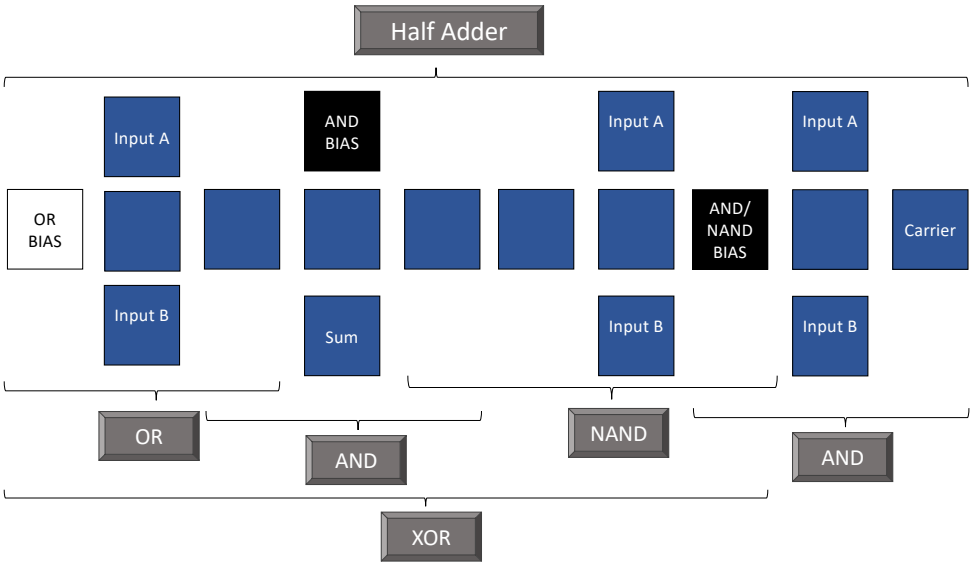


Figure 3.15.: The half adder' setup is consisting of 18 dipolar-coupled nanomagnets. It consists of a combination of XOR- and AND gates. The operations are underlying the logic provided by the logical table 3.14.

We have chosen the same distance and magnet specification as in the simulation of the majority gates. Likewise, we have initialized the free bits, all bits which do not act as Bias, of the coupled system along the x-axis. During the simulation, we altered the current acting on the inputs to go through all four possible input configurations $\uparrow\uparrow$, $\uparrow\downarrow$, $\downarrow\uparrow$, $\downarrow\downarrow$ and observe how the C and S reacted.
We observe that neither the C nor the S flip its state according to what we expect to

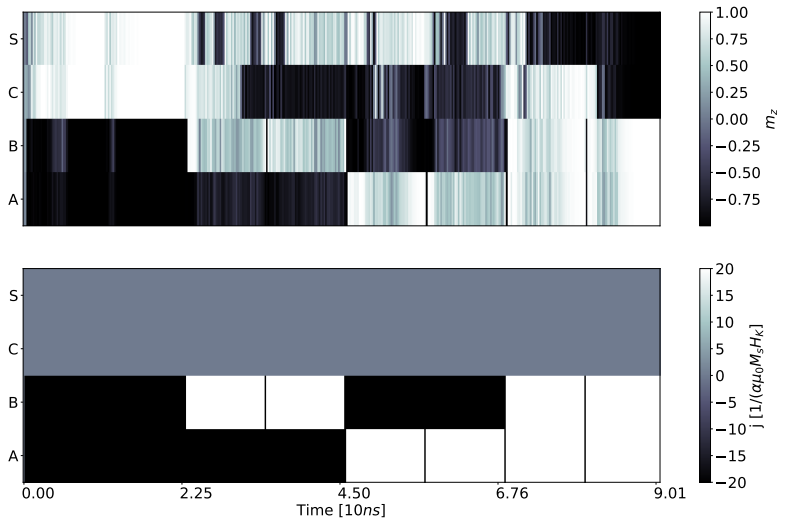


Figure 3.16.: Simulation results of the half adder, which is composed of majority gates. The current acting on the inputs is periodically changed to alter the magnetization of the inputs and the outputs. Furthermore, there is a clocking field acting on the system to reset it every 32,500 natural time steps $\Delta\tau$.

form the observations we made before on the majority gates. Similar to the unclocked AND/OR gate 3.12 we observe that our coupled-majority gate systems get stuck in some kind of metastable state. Despite the clocking of the half adder' gate, we observe that there is no guarantee that the system will encounter states that satisfy the logical table more often than a metastable state. We observe certain randomness due to the numerical calculation, although the simulations have been performed at zero temperature.
Nevertheless, we will use this randomness to our advantage to overcome metastable states and fluctuate into the ground state of our system. Therefore we will expand the calculus in the next chapter to include thermal fields into the LLG.

4. Stochastic Integration

The LLGS allows us to couple macrospins and manipulate them by applying currents. In principle, this will enable us to build macrospin structures that can mimic conventional logic. This fact is made possible in particular by the deterministic nature of the LLGS.
However, the goal of this work is to reverse the operating direction of logical gates. The underlying problem is that the logical gates no longer show such deterministic behavior when the working direction is reversed, as already mentioned. This circumstance forces us to add some stochasticity to our system.
If we look at real magnetic systems, we see that below the Curie temperature, the magnetic moments are ordered. However, when the Curie temperature is exceeded, we see that the magnetic moments arrange themselves randomly. This temperature dependence is also embedded in the Gibbs free energy of the magnetic system. Therefore, the introduction of a temperature bath is the natural choice of stochasticity for the macrospins under consideration.
In this chapter, we will discuss a way to add thermal fluctuations to our system and describe stochastic calculus to solve the extension of the LLGS.

4.1. Adding a Heat Bath

The first question we need to address in this section is how we can incorporate a heat bath into our system. The idea is to add energy to the system by means of a random field that interacts with the system [40]

$$\mathbf{H}_{eff} \rightarrow \mathbf{H}_{eff} + \mathbf{H}_{th}. \tag{4.1.1}$$

The fluctuation field is chosen to be represented by a stochastic process $\nu(t)$. These idea goes back to Brown who stated in [40] that the typical timescale of the fluctuations is much smaller then the precesssion frequency if the magnetisation vector. The physical origin of this assumption is that the classical spin will fluctuate due to excitations caused by the interaction with, i.e., phonons or electrons [41] . A large number of interactions are responsible for spin excitations. All of the excitations are themselves subject to stochastic processes that have different stochastic distributions. As a result of the central limit theorem, the random field is Gaussian distributed. This lead to the conclusion that the mean value of the distribution must vanish

$$\langle \nu(t) \rangle = 0. \tag{4.1.2}$$

Furthermore, we assume that all spins are uncorrelated and that the fluctuation field is uncorrelated in time. This assumption, however, is just valid if the correlation time of subsequent stochastic collisions is short compared to the timescale of the spin motion. The mathematical formulation of the expression reads

$$\langle \nu(t)\nu(t')\rangle = 2C\delta(t-t'). \tag{4.1.3}$$

This is an important assumption that allows us in the following to rewrite the the LLGS equation in the form of a stochastic differential of a Langevin type. By assuming that the energy distribution follows a Boltzmann distribution

$$P(E) \propto \exp(-E/k_BT), \tag{4.1.4}$$

and that we can write the dynamics in the form of a Langevin equation including the Gilbert damping the Fluctuation-Dissipation-Theorem guarantees the following relation between the damping constant α and the temperature T [42], [43]

$$C = \frac{\alpha k_B T}{2K(1+\alpha^2)}. \tag{4.1.5}$$

Nevertheless, we refer to stochastic processes that show these properties as Gaussian-White noise. White noise processes have an independence of the frequency

$$F(\omega) = \int ds \langle \nu(t)\nu(t+s)\rangle \exp(i\omega s). \tag{4.1.6}$$

In principle, it is unphysical to assume white noise because the distribution of white noise would lead to diverging power and a colored noise distribution $g(t)$ with correlation length τ would be more reasonable

$$\langle g(t)g(t')\rangle \propto \exp(-t'/\tau). \tag{4.1.7}$$

Due to the short correlation time of the stochastic processes in comparison to the macrospin correlation time, the Wong-Zakai theorem states that the more physical colored noise goes for $\tau \to 0$ over into white noise [44].

4.2. Stochastic Differential Equation

As soon as we consider thermal fluctuations to be modeled as a stochastic process, we no longer deal with ordinary differential equations but with stochastic differential equations.
In this section, we want to introduce the mathematical framework to solve these stochastic differential equations.

4.2.1. Stochastic differential equations with an additive part

In this section, we want to derive a so-called Wiener process which we will use to integrate the stochastic differential equation. Therefore we start with the simple one-dimensional case containing an additive term, which in physics is know as the Langevin equation [45]

$$\frac{dX(t)}{dt} = a(X(t), t) + \nu(t). \tag{4.2.1}$$

The Langevin equation contains a deterministic drift term $a(X(t), t)$, and a stochastic diffusive term $\nu(t)$. The increment of the equation over a small-time internal dt leads to

$$dX(t) = a(X(t), t) + dW(t), \text{ where } dW(t) = \int_t^{t+dt} \nu(t')dt'. \tag{4.2.2}$$

If we assume that ν corresponds to a white noise process that satisfies 4.1.2 and 4.1.3, then dW is also a Gaussian random variable because it is a sum of those. Thus from calculus follows,

$$\langle (dW(t))^2 \rangle = \int_t^{t+dt} dt_1 \int_t^{t+dt} dt_2 \langle \nu(t_1)\nu(t_2) \rangle \tag{4.2.3}$$

$$= \int_t^{t+dt} dt_1 \int_t^{t+dt} dt_2 2C\delta(t_2 - t_1) \tag{4.2.4}$$

$$= 2Cdt. \tag{4.2.5}$$

As before, as long as the times t, t' are not correlated, we can conclude that $\langle dW(t)dW(t') \rangle = 0$, since there is therefore no temporal correlation between t and t'. The order of *dW(t)* corresponds to the square root of the order dt therefore we can conclude

$$dW(t) = 2C\nu(t)\sqrt{(}dt). \tag{4.2.6}$$

This stochastic process $dW(t)$ is known as a Wiener process.

4.2.2. Integration of a Stochastic Integral

In this section, we will exploit the Wiener process to integrate a more general kind of stochastic differential equation, namely the one-dimensional stochastic differential equation with multiplicative noise

$$\frac{dX(t)}{dt} = a(X(t), t) + b(X(t), t)\nu(t). \tag{4.2.7}$$

For a short time dt the increment dX can be written as

$$dX(t) = \int_t^{t+dt} a(X(t'), t')dt' + \int_t^{t+dt} b(X(t'), t')\nu(t')dt'. \tag{4.2.8}$$

The first term is deterministic and can be integrated as usual. The integration is based on the Riemann integral with a discretization of the time interval in N steps of t_i, where a time step is defined as Δt

$$\sum_{j=1}^{N} a\big(\delta X(t_j) - (1-\delta)X(t_{j-1}), \delta t_j + (1-\delta)t_{j-1}\big) \cdot (t_j - t_{j-1}). \tag{4.2.9}$$

The Riemann integral is converging for arbitrary choices of the function in 4.2.9 between t_{j-1} and t for $\Delta t \to 0$. Thus, for any choice of δ in the interval $\delta \in [0, 1]$ the integral will converge. On the other hand for stochastic differential equations there is the Riemann-Stieltjes integral, which we need to solve in case of 4.2.8

$$\sum_{j=1}^{N} b\big(\delta X(t_j) - (1-\delta)X(t_{j-1}), t_j\big) \cdot (W(t_j) - W(t_{j-1})). \tag{4.2.10}$$

In the case of a stochastic integral, the convergence is not guaranteed in the limit $\Delta t \to 0$. The Ito-Stratonovich controversy states that the integral converges for $\delta = 0$ and $\delta = 1/2$. Furthermore, it is not ensured that the result of the integration of both choices is the same [46, 47, 48].
Nevertheless is it possible to approximate the stochastic part of 4.2.10 for small $\delta t \to dt$ by

$$b\big(\delta X(t + dt) + (1-\delta)(X(t), t\big)dW(t). \tag{4.2.11}$$

The term in the parenthesis can be rewritten as

$$\begin{aligned} \delta X(t + dt) + (1-\delta)X(t) &= \delta(X(t) + dX(t)) + (1-\delta)X(t) \\ &= X(t) + \delta dX(t). \end{aligned} \tag{4.2.12}$$

Thus, the stochastic differential equation is given by

$$dX(t) = a(X(t), t)dt + b(X(t) + \delta dX(t), t)dW(t). \tag{4.2.13}$$

Expanding the second term of RHS to second order in *dx(t)* will then result in

$$
\begin{aligned}
b(X(t)+\delta dX(t),t)dW(t) &= b(X(t),t)dW(t) + \frac{\partial b(X(t),t)}{\partial X(t)} dW(t)(\delta dX(t)) + ... \\
&= b(X(t),t)dW(t) + \delta \frac{\partial b(X(t),t)}{\partial X(t)} a(X(t),t)dtdW(t) \\
&+ \delta \frac{\partial b(X(t),t)}{\partial X(t)} b(X(t) + \delta(X(t),t)(dW(t))^2 + ... \\
&= b(X(t),t)dW(t) + \delta \frac{\partial b(X(t),t)}{\partial X(t)} b(X(t))(dW(t))^2 + O(dW(t)^3),
\end{aligned}
\tag{4.2.14}
$$

where we have substituted once again4.2.13 back in to obtain an expansion in powers of dt. By inserting 4.2.14 back into 4.2.13 one finds

$$
\begin{aligned}
dX(t) &= a(X(t),t)dt + \delta \frac{\partial b(X(t),t)}{\partial X(t)} b(X(t),t)(dW(t))^2 + b(X(t),t)dW(t) \\
&= \left[a(X(t),t) + \delta \frac{\partial b(X(t),t)}{\partial X(t)} b(X(t),t)dW(t)^2(t) \right] dt + b(X(t),t)dW(t).
\end{aligned}
\tag{4.2.15}
$$

In this equation, we find an additional drift term containing α and $dW(t)^2(t)$. The latter can be replaced by 1 for terms up to the order of dt. As mentioned above, depending on the choice of δ and the interpretation of the integral, we get different drift terms.
For setting $\delta = 0$, one finds the so-called Itô interpretation of the stochastic integral

$$
dX = a(X(t),t)dt + b(X(t),t)dW(t), \tag{4.2.16}
$$

which is indicated by writing 4.2.16 as

$$
dX = a(X(t),t)dt + b(X(t),t) \cdot \dot{W}. \tag{4.2.17}
$$

A further option is setting $\delta = 1/2$

$$
dX = \left[a(X(t),t) + \frac{1}{2} \frac{\partial b(X(t),t)}{\partial X(t)} b(X(t),t)\nu^2(t) \right] dt + b(X(t),t)dW(t), \tag{4.2.18}
$$

which is called the Stratonovich interpretation of the integral. To differenciate it from the Itô interpretation one usually writes

$$dX = a(X(t), t)dt + b(X(t), t) \circ \dot{W}. \tag{4.2.19}$$

If we look at both interpretations, we notice that they only add an additional noise-induced drift term

$$[\frac{1}{2}\frac{\partial b(X(t), t)}{\partial X(t)} b(X(t), t)\nu^2(t)]dt. \tag{4.2.20}$$

For example, the Stratonovich interpretation of the Itô integral can be written as

$$dX = \left[a(X, t) + \frac{1}{2}\frac{\partial b(X(t), t)}{\partial X(t)} b(X(t), t)\nu^2(t)\right]dt + b(X, t) \cdot dW(t), \tag{4.2.21}$$

and vice versa for the Itô interpretation of the Stratonovich integral.
There is some leeway in the choice of integrals, but unfortunately, there is no generally valid solution. In the mathematical literature, the Itô interpretation is more present because it fulfills important relationships [49].
In our derivations, we have mainly used the Itô interpretation. However, it turned out that especially dynamic spin systems can be better described by the Stratonovich interpretation due to the colored noise going to white noise in the zero correlation time limit, as mentioned before [50].

4.3. The Stochastic Landau-Lifshitz-Gilbert Slonczewski Equation (sLLGS)

So far, we have derived a introduced a framework that allows us to integrate a general Langevin equation

$$dX(t) = a(X(t), t) + b(X(t), t) \circ dW. \tag{4.3.1}$$

This framework can now be combined with the micromagnetic framework by replacing the deterministic drift term a from the LLGS equation.

$$\mathbf{a}(\mathbf{m}) = \mathbf{m} \times \mathbf{h}_{eff} - \alpha\mathbf{m} \times (\mathbf{m} \times \mathbf{h}_{eff}) - \alpha I\mathbf{m} \times (\mathbf{m} \times \hat{\mathbf{n}}_p), \tag{4.3.2}$$

where $I = j(\alpha\mu_0 M_s H_K)$.
The stochastic diffusion term is then given by

$$b_{ik} = \sqrt{C} = [-\epsilon_{ijk}m_j - \alpha(m_i m_k - \delta_{ik})]. \tag{4.3.3}$$

The form of the diffusion term can be derived by assuming isotropic noise and recasting the Langevin equation into a Fokker-Planck equation and collecting all contributing terms 4.3.1.
With this, the stochastic dynamics of a macrospin is given by

$$\frac{d\mathbf{m}_i}{dt} = \mathbf{a}_i(\mathbf{m}, t) + b_{ik}(\mathbf{m}, t) \circ H_{th,K}(t). \tag{4.3.4}$$

The influence of the stochastic contribution is shown in Figure 4.1.

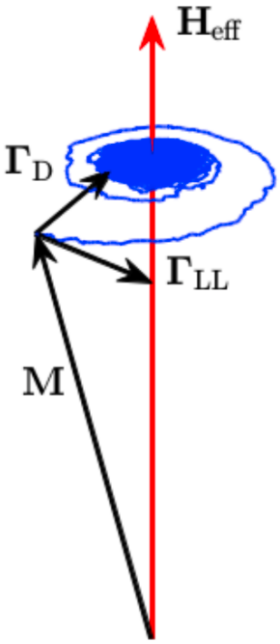


Figure 4.1.: Schematic representation of the LLGS showing the influence of thermal fluctuations on the trajectory [22].

5. Numerical Methods

The previously discussed methods and models allow us to build the equations for coupled macrospins, but we are interested in describing their dynamics throughout several hundred microseconds. Therefore we have to use a suitable integration method that allows us to integrate the equations in simulations as efficiently as possible. In this chapter, we will introduce the Heun scheme as an extension of the Euler method for stochastic differential equations and discuss the advantages of using graphics processing units (GPUs) in this context.

5.1. Integration Scheme

In this section, we will present the Heun scheme as the numerical integrator of our choice, which includes the Euler-Maruyama scheme as a particular case.

5.1.1. Euler-Maruyama Scheme

The Euler-Maruyama method (also called the Euler method), is in the Itô calculus technique allows us to approximate the numerical solution of a stochastic differential equation (SDE). It is a simple generalization of the Euler method for ordinary differential equations to stochastic differential equations. It is named after Leonhard Euler and Gisiro Maruyama. Unfortunately, the same generalization cannot be made for any deterministic method [49].
As mentioned, we consider a stochastic differential equation in the Itô interpretation

$$dX_t = a(X(t), t)dt + b(X(t), t)dW_t, \tag{5.1.1}$$

with initial condition $X_0 = x_0$, where W_t represents the Wiener process on $t_0 \leq t \leq T$. For a given discretization $t_0 = \tau_0 < \tau_1 < ... < \tau_n < .. < \tau_N = T$ of the time interval $[t_0, T]$, the so-called *Euler approximation* of a continuous time stochastic process $Y = Y(t), t_0 \leq t \leq T$ that satisfies the iterative scheme [49].

$$Y_{n+1} := Y_n + a(Y_n, \tau_n)\Delta_n + b(Y_n, \tau_n)\Delta W_n \tag{5.1.2}$$

for $n = 0, 1, 2, ..., N-1$ with inital value $Y_0 = X_0$. The time discretization is then given by $\Delta_n = \tau_{n+1} - \tau_n$, the increment of the stochastic process reads $\Delta W_n = W_{\tau_{n+1}} - W_{\tau_n}$ and $Y_n = Y(\tau_n)$.
In the section 4.1, we already discussed that the random increments $\langle \Delta W_n \rangle$ are Gaussian distributed because we will deal with white noise which satisfies the following relations for the mean

$$\langle \Delta W_n \rangle = 0, \tag{5.1.3}$$

and the variance

$$\langle (\Delta W_n)^2 \rangle = 2C(\tau_{n+1} - \tau_n). \tag{5.1.4}$$

The Euler scheme results for the Macrospin model with a constant time step Δt in

$$\mathbf{m}_i = \mathbf{m}_i(t) + \mathbf{a}_i(\mathbf{m}, t)\Delta t + b_{ik}(\mathbf{m}, t)\Delta W_k, \tag{5.1.5}$$

with

$$\langle \Delta W_l \rangle = 0, \ \langle \Delta W_k \Delta W_l \rangle = 2C\delta_{kl}\Delta t. \tag{5.1.6}$$

Since we assumed the Itô interpretation, we need to take in the Stratonovich calculus an additional noise drift term into account

$$\mathbf{m}_i(t + \Delta t) = \mathbf{m}_i(t) + \left[\mathbf{a}_i(\mathbf{m}, t) + 2C\frac{1}{2}b_{jk}\frac{\partial b_{ik}}{\partial M_j}\right]\Delta t + b_{ik}(\mathbf{m}, t)\Delta W_k, \tag{5.1.7}$$

.

As a short remark in the case of $b = 0$, the Euler-Maruyama scheme is reduced to the deterministic Euler scheme for the ordinary differential equation.

The most important theoretical result concerning the Maruyama scheme describes its strong convergence (or stochastic convergence) against the solution S :

Definition: sequence of stochastic processes (S_t^n), $0 \leq t \leq T$, $n \in \mathbb{N}$ on a common probability space converges by definition strongly with order q against a process (S_t), $0 \leq t \leq T$ if there is a constant c, so that for all $t \in [0, T]$

$$E(|S_t^{(n)} - S_t|) \leq cn^{-q} \ \ \forall t \in [0, T]. \tag{5.1.8}$$

In the case of the Maruyama scheme it can be now shown that the discretization (S_t) converges for $n \to \infty$ strongly with order $\frac{1}{2}$ against the solution S of the stochastic initial value problem, if for all real numbers x and all positive s, t the following bounds applies

$$|a(s, x) - a(t, x)| + |b(s, x) - b(t, x)| \tag{5.1.9}$$

$$\leq K(1 + |x|)\sqrt{(|t - s|)}.$$

On the other hand, weak or distributional convergence with order q is is given when the bound

$$|E(f(S_t^{(n)}) - E(f(S_t))| \leq cn^{-q} \ \ \forall t \in [0, T] \tag{5.1.10}$$

applies for a fixed constant c and all functions f being are at least $(2q + 2)$-times continuously differentiable and all derivatives of which are bounded by polynomials.

For sufficiently smooth functions a and b, the Euler-Maruyama method typically has the weak order of convergence $q = 1$ [51].
In practice strong convergence is seldom of interest since in most cases it is not a specific solution to a particular Wiener process that is sought, but rather a sample from the probability distribution of the process described by the weak convergence.
The Euler-Maruyama scheme gives good numerical results when the drift and diffusion coefficients are almost constant. However, this is rarely the case in practical applications, and then good numerical results cannot be expected. Therefore the use of higher integration schemes is common.

5.1.2. Heun Scheme

The point here is to reinterpret Euler's method in terms of an integral equation for the solution. The improved Euler-Maruyama or Heun method is thus a predictor-corrector method with the forward Euler method as a predictor and the trapezoidal method as a corrector [52].
In the case of the Langevin equation, the predictor is:

$$\tilde{m}_i = m_i(t) + A(\mathbf{m}, t)\Delta t + B_{ik}(\mathbf{m}, t)\Delta W_k, \tag{5.1.11}$$

where Δt is as before the time step discretization and ΔW_k is a Gaussian distributed random number, which satisfies the relations 5.1.6 [22].
Thus, the Heun scheme can be written as

$$\begin{aligned} m_i(t + \Delta t) = m_i(t) &+ \frac{1}{2}[a_i(\tilde{\mathbf{m}}, t + \Delta t) + a_i(\mathbf{m}, t)]\Delta t \\ &+ \frac{1}{2}\Big[b_{ik}(\tilde{\mathbf{m}}, t + \Delta t) + b_{ik}(\mathbf{m}, t)\Big]\Delta W_k. \end{aligned} \tag{5.1.12}$$

In contrast to the Euler-Maruyama scheme, the stochastic Heun scheme converges with a strong order 1 and weak order 2 to solve the general system of Langevin equations. There are two arguments why the Heun's scheme for the numerical integration of the sLLGS equation is preferable:
The first is that for the Heun scheme, it can be shown that its solutions converge to Stratonovich's solutions of stochastic differences without adding the noise drift term. Secondly, the solution of the method is numerically more stable due to the higher order of convergence.
Therefore, in the course of this thesis, we will use the Heun scheme to integrate the stochastic differential equation.

5.2. GPU

Graphics Processing Units (GPUs) are a particular class of calculation units that were developed in the past mainly to render graphics for computer games. The resulting

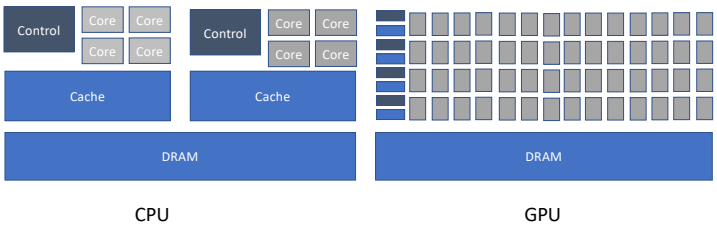


Figure 5.1.: Schematic representation of the architerture of a CPU and a GPU. In the case of the CPU, the connection of the processors to the cache and the associated DRAMs is shown. In case of the GPU multiple of these systems are located within a single GPU.

GPU architectures are very specialized in their task to create 3-dimensional graphics that massive amounts of parallel floating-point calculation. This continuous, competitive development has led to hardware that is optimized for parallelized calculations. As a result this hardware has become very efficient, but also very cost-effective. In 2007, the hardware manufacturer NVIDIA made the existing GPU hardware available to software developers via a CUDA interface. Since then, there has been great interest in using GPUs to accelerate highly parallel parts of scientific computing.
In this section, we want to show the advantages of a parallel implementation of the macrospin model.

5.2.1. GPU Architecture

The central processing unit (CPU) has few cores, but they calculate very fast. As a result, CPUs are suitable for complex computations that require the result of the previous calculation, commonly known as sequential computation. In contrast, the GPUs have a large number of processing cores connected by a high-bandwidth bus over a large amount of high-speed memory (see Figure 5.1 for a comparison with a traditional CPU architecture). This arrangement means that large memory blocks can be moved to the series of processors, each processor ideally work on an element of this memory. Such a transaction is the most efficient method for using a GPU. Therefore it is useful to use algorithms that compute vectors to benefit from parallelization.

5.3. CUDA Work Flow

The CUDA interface to a programming language allows the user to calculate directly on the GPU.
The workflow of CUDA is shown in 5.2 and requires the user to transfer data to be processed to the GPU in advance. After the data is transferred, the user can interact

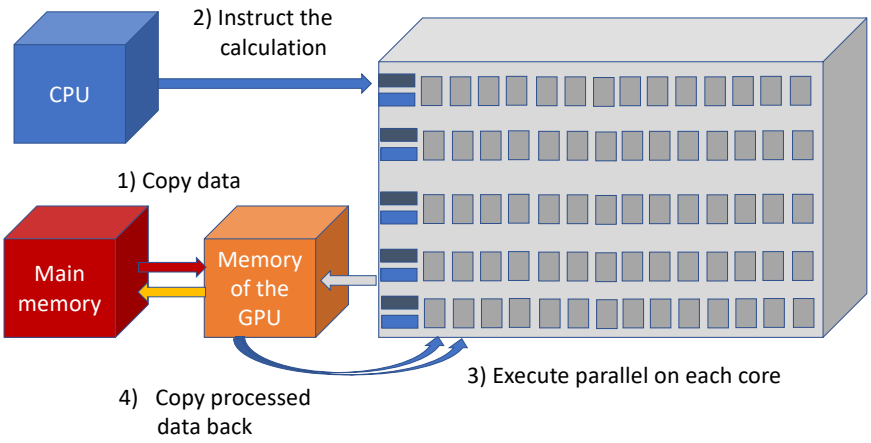


Figure 5.2.: Illustration of the CUDA Workflow: The diagram shows how the main memory interacts with the CPU and GPU

with the data using the CPU. The CPU can either run a kernel on the GPU, i.e., process the data, or copy the data from the GPU back into main memory.
In our case, we split the code into two separate parts that interact with each other. There is a Python code that mimics the tasks of the GPU. It initializes the magnetic system, calculates the interacting matrix, handles the reading and writing of data, and calls the kernel that does the integration. The second part of the code is the kernel code, written in C, which basically performs the temporal development of the system by executing the Heun scheme until a final condition is reached.

6. Inversion of the Nanomagnetic Logic

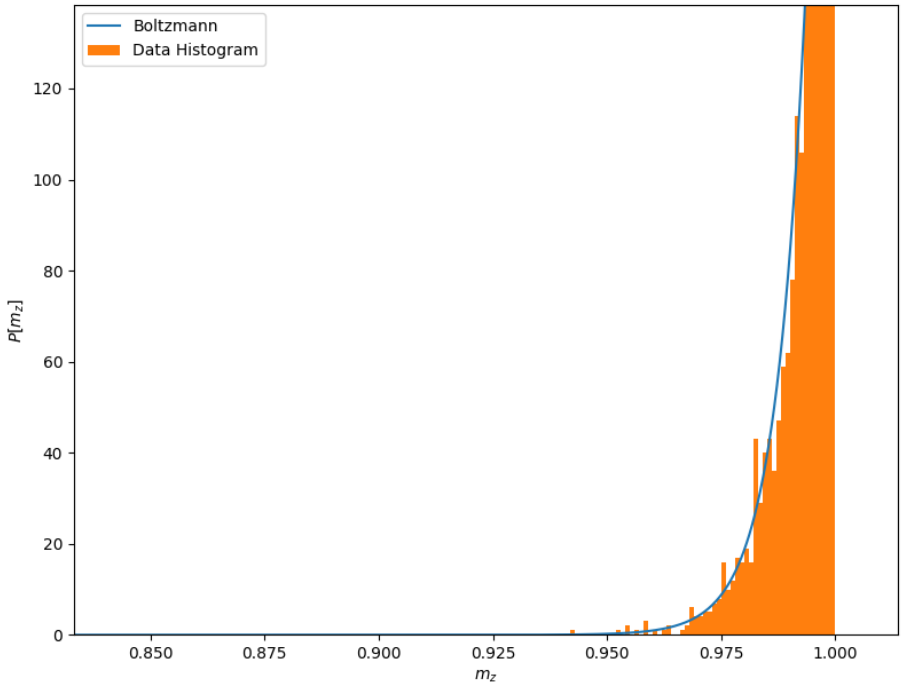


Figure 6.1.: Histogram distribution of m_z after the relaxation of the magnetic system into thermal equilibrium (10^3 natural time units). The superimposed blue line is the Boltzmann distribution of the theoretical equilibrium. We show that the data $\chi \equiv K_e V/k_B T = 80$, the ratio between total anisotropy and thermal energy, scales.

First of all, we will consider how stochasticity affects the magnetization dynamics. The starting point for this theoretical consideration is a single macrospin. For simplicity, we model the nanomagnet as a uniaxial sample. Such samples can be realized by a circular magnetic domain, which suppresses the shape anisotropy of the magnet.
In order to model such samples, we need to set $K_h = 0$ in the macrospin energy term 2.6.8 discussed in chapter 2. Thus we can write the energy as

$$E(\mathbf{m}) = -K_e m_z^2. \tag{6.0.1}$$

Without applying any external field, we expect that the spin will relax towards the minimum energy configuration ($m_z = 1$ or $m_z = -1$) according to the dynamics described by 5.1.12. We initialize the spin by letting the magnetization point in the $\mathbf{x} - \mathbf{y}-$plane.
As long as the energy ratio between the anisotropy and the thermal energy $\chi = K_e/k_BT$ is large enough, the flipping of the spin due to random fluctuations is suppressed, and the relaxation behavior can be studied by observing the $\mathbf{z}-$component of the magnetization vector [22] . Thus, the switching dynamics are given by

$$\begin{aligned}\dot{m}_z = \alpha[(I\cos(\theta) + m_z)(1 - m_z^2) + I\sin(\theta)cos(\phi)m_z(m_x - \tan(\phi m_y)] \\ + \alpha^2 I\sin(\theta)m_y + \sqrt{\frac{\alpha}{\chi}(1 - m_z^2)} \circ \dot{W},\end{aligned} \tag{6.0.2}$$

where θ and ϕ are defined in reference to Figure 2.1. $\dot{W}$ is a Wiener process with the standard mean zero, variance 1, and its prefactor expresses the strength of the compounded stochastic effects derived from the FDT. For small ones of θ the 4.3.4, the dynamic equation of m_z decouples from m_x and m_y, and we are left with a 1-D problem [22]. The first test of our numerical scheme will be the successful reproduction of the thermal equilibrium properties of the uniaxial model of the magnetic bit. A typical histogram of thermalized magnetic orientations resulting from this simulation is shown in 6.1 and confirms the proper functioning of the numerical stochastic model by accurately reproducing the expected Boltzmann equilibrium distribution.
We show that the implementation of the thermal fluctuations corresponds to the Boltzmann distribution. Starting from this, we want to investigate the switching behavior of the nanomagnets. In Ref. [29], it is discussed how the height of the energy barrier $\chi = K_eV/(k_BT)$ affects the number of switching events that are given by the noise amplifying term in 4.2.13. In the noise amplifying term, the volume is the only material parameter that can be reliably tuned to generate enough switching events at room temperature. The shrinking of the volume lead towards more switching events. The Néel-Arrhenius equation

$$\tau_N = \tau_0 \exp\left(\frac{K_eV}{k_BT}\right), \tag{6.0.3}$$

describes the mean time between two flips. The τ_N is called Néel relaxation time, and τ_0, which is characteristic of the material, is called the attempt time, which is typically between 10^{-9} and 10^{-10} seconds. In Figure 6.2, we calculated the number of switching events for 1.5 μs as a function of $1/\chi$. As expected, we see that for smaller particles and higher temperatures, the number of switching events is increasing. Furthermore, we can fit the data to the Néel-Arrhenius equation 6.0.3 and extract an attempted time of 3.793 ns.

Based on the results of 6.2 we perform numerical simulations of the gates at $M_S = 100000\ Am^{-1}$ and $= 8$. Choosing results in a sufficient number of switching operations to collect statistics. In addition, the choice requires nanomagnets in the size of several nm for relevant materials and room temperature. At this size, the

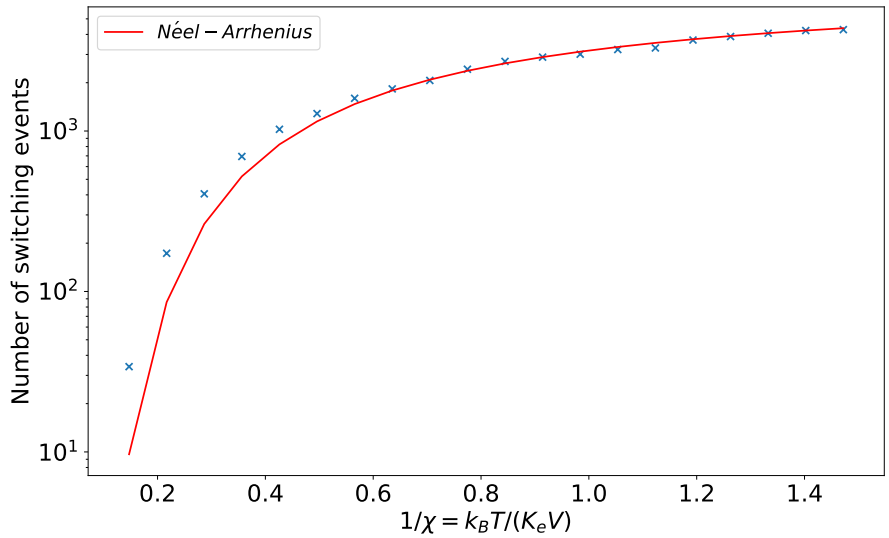


Figure 6.2.: Numerical simulation of the number of thermal switching events of the m_z component as a function of $1/\chi$. The solid line is a fit of Néel-Arrhenius equation 6.0.3. Number of switching events is increasing with the temperature and decreasing with growing volumes and higher anisotropy values.

nanomagnets become thermally unstable, as described in Ref. [13], which is necessary to introduce sufficient stochasticity into the system. The size of a natural time step $\Delta\tau$ is of the order of ps.

In the following, we investigate how these fluctuations affect the stability of the majority gate. Since for NAND/NOR and AND/OR majority gates, only the number of magnets used differs, we consider how the number of magnets affects the stability of the system. If we assume that the magnets within the coupled system always relax along the easy axis, we can assume that the number of possible states scales to $2^{(n+1)}$. This means that we are talking about 16 possible states in the case of the NAND/NOR majority Gate and 32 possible states in the case of AND/OR majority Gate.
However, only four of these states are realized by Boolean algebra, which means that we can get states in our system that are not suitable for conventional computing. Therefore, in the following, we will refer to states of the magnetic system which are agreeing with Boolean algebra as logically consistent states. By adding thermal fluctuations to the dynamic, we expect that the system will more often enter states where the total energy is minimized by the dipole-dipole interaction due to random spin flips. The thermal fluctuations must be strong enough to overcome the energy barrier of metastable states.

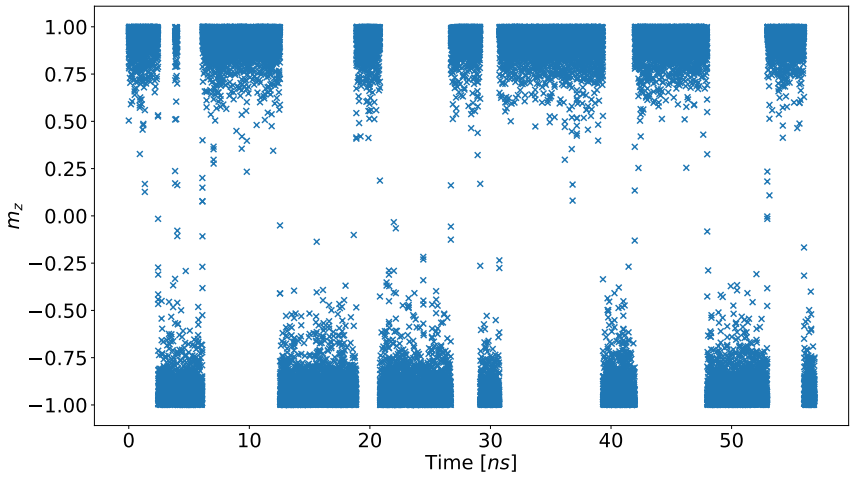
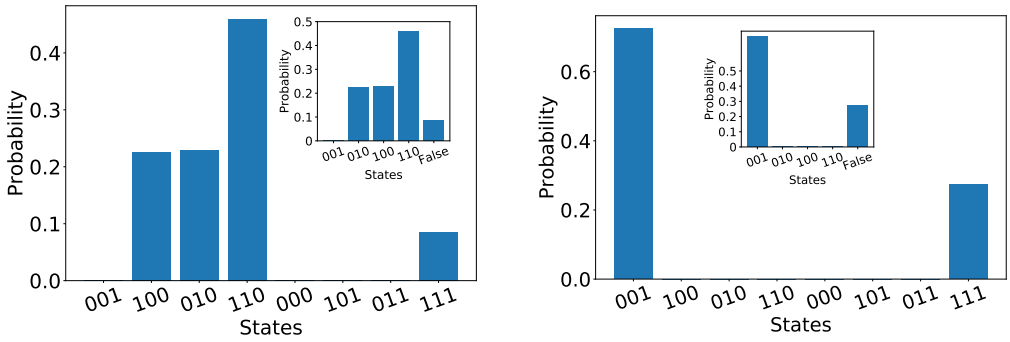


Figure 6.3.: Numerical simulation of the m_z-component of a nanomagnet at 300 K with $K_e = 1e5\frac{J}{m^3}$ J/m, $V = 5 \times 3 \times 3$ nm^3 $M_S = 1000000$ Am^{-1}.

6.1. Inversion of the Majority Gate



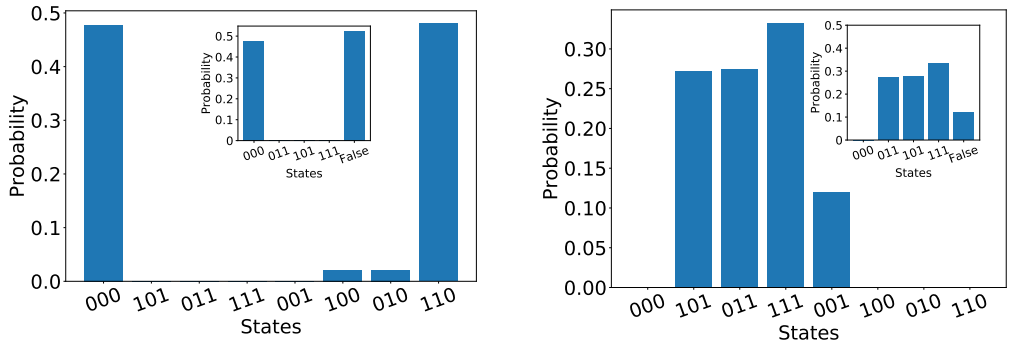
(a) NOR gate with output set to 0.

(b) NOR gate with output set to 1.

Figure 6.4.: The probability distribution of the OR gate in the backward operation direction while the output has been kept constant to either 1 or 0. Simulation parameter: $\chi = 8$ and $M_S = 1000000$ Am^{-1}, $\alpha = 0.04$, $\Delta\tau = 0.04$ $s/\gamma' H_K$ and a distance $r_{ij} = 2.5$ nm between the nanomagnets.

Since the main part of this work is about the inversion of such magnetic logic, we will first take a look at the AND/OR- and NAND/NOR-majority gate.
For reasons of symmetry, the energy of the states of both versions of a majority gate is the same; only the interpretation of these states is different. Nevertheless, the results

for the AND and NAND gate are shown in the appendix A.2. We keep the Bias magnet aligned parallel or antiparallel to the $\mathbf{z}-$axis so that we get an OR or NOR gate configuration.
As we are interested in investigating the inversion, we will not fix the inputs, but the outputs of the gates in contrast to the previous simulations. Thus the released input magnets can be manipulated by the stray fields, whereby we want to examine whether they fall into the logically consistent states.
For this purpose, we calculate histograms from thousands of relaxation processes of the OR- and NOR gate for $\chi = 8$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets. The results are shown in Figures 6.4,6.5.



(a) OR gate with output set to 0.

(b) OR gate with output set to 1.

Figure 6.5.: The probability distribution of the OR gate in the backward operation direction while the output has been kept constant to either 1 or 0. Simulation parameter $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

In both histograms of both gates, we observe that despite the thermal fluctuations, it is not possible to exclude not logically consistent states. In addition, the histograms 6.5a and 6.4b shows that for output values with several combinations of input values, the probability for each of them can be different. The additional magnet in the OR-gate configuration leads towards more logical inconstant states as we expected from the forward simulations. Therefore we will concentrate on the OR gate for the time being, whereas the argumentation for the NOR gate will be the same. In this particular case, we observe that the state '111' of the OR-gate, where the first value refers to input A, the second to input B and the third to the output, is more likely to then the '011' or '101' state. This is quite obvious if we look at the arrangement of the OR-gate in Figure 6.6 and argue that dipolar interaction wants adjacent magnets to align the magnetization of adjacent magnets antiparallel. Thus, state '111' minimizes the total energy since the dipolar energy is minimized, and the anisotropy is independent of the orientation direction along the z-axis. However, if one looks at the states '011' and '101', one can see that one of the input spins breaks the desired antiparallel of

the dipolar interaction. This can be quantified by calculating the stationary dipolar energy for all possible combinations of possible states 6.6. As expected, the dipolar energy for the state '111' is at its lowest, and the states '011' and '101' are degenerate, which can be seen in the corresponding histograms 6.5b.

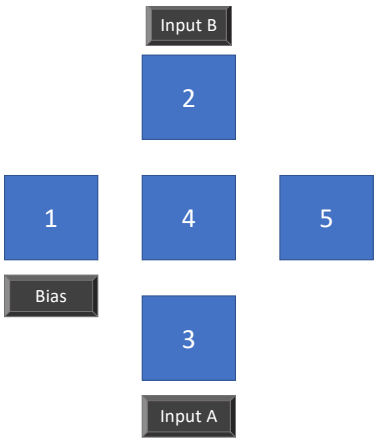


Figure 6.6.: Numbering of the arrangment of the OR gate.

Energies $[J/H_K]$	States [1,2,3,4,5]
-0.371752	11101
-0.318310	10010
-0.318310	10101
-0.318310	11001
-0.318310	11100
-0.185290	10001
-0.185290	10011
-0.039789	11010
-0.039789	10110
-0.039789	10100
-0.039789	11000
0.318310	11110
0.318310	10000
0.318310	10111
0.318310	11011
0.901488	11111

Table 6.1.: States according to the numbering shown in the Fig 6.6 including there dipole energy.

The inequality of the probability distribution of the input combinations means that if several of these gates are connected to a circuit, this inequality is accumulated. Thus, the input combination is composed mainly of the paths that consist of the most probable combinations.

6.2. Inversion of the Half Adder

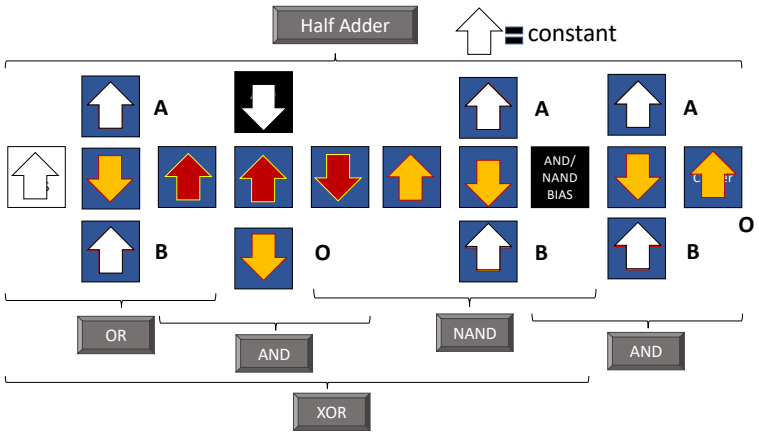


Figure 6.7.: A logically consistent state of the half adder where inputs A and B are set to 1. The red arrows indicate a defect group.

If we invert the half adder construction presented in chapter 3, we can use this consideration to think in advance about feasible states of the system. A possible realization is shown in Figure 6.7. In this realization, the magnetization of the bias and inputs of the AND, NAND, and OR gates are set. We see that just setting these magnetizations leads to an energy barrier in the XOR gate. From this, we can conclude that this logically consistent state is not a ground state of dipolar energy. Likewise, the remaining logical consistent states will not be ground states of dipolar energy, which is obvious considering the symmetrical arrangement of the gate. Thus energy barriers will also occur in these states.
In Figure 6.8, a possible inversion of the gate is shown where the energy barriers are a problem. In this situation, it cannot be guaranteed that the upper magnets marked with the question mark, which make up input A, always assume the same state. The orientation of the macrospin in the OR gate forces its inputs to the state 0, while the inputs of the NAND gate are forced to the state 1. The resulting states are not logical consistent states of our half adder. The findings are based on the histograms shown above, and the simulation results are shown in Figure A.5.

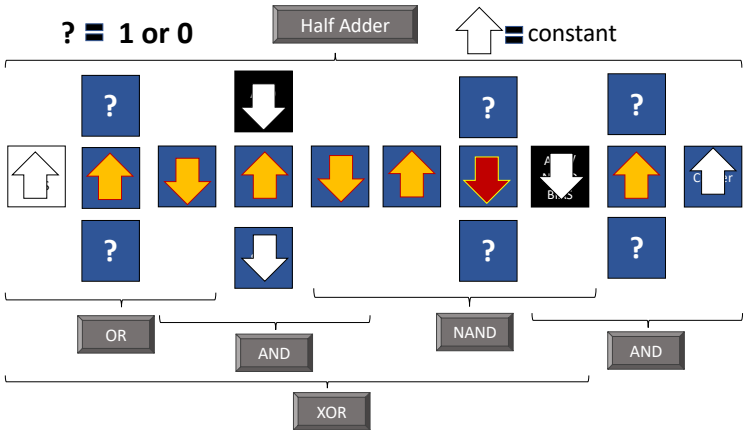


Figure 6.8.: Inversion process of the half adder where inputs $C = 0$ and $S = 0$. The red arrows indicate a defect group.

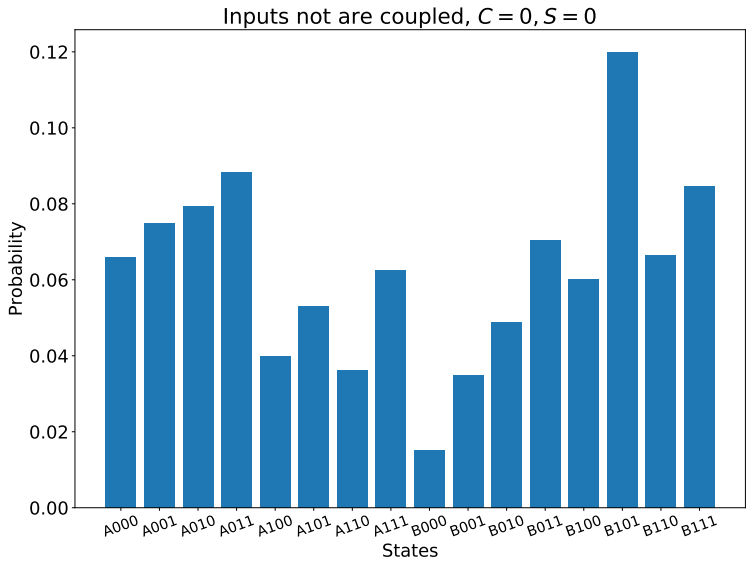


Figure 6.9.: Probability distribution of half adder states for the carrier $C = 0$ and sum $S = 0$. Bars show given states combinations for all three magnets which are acting as input A or B. Simulation parameter $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

However, in a theoretical study, we can still investigate the characteristics of this circuit by introducing another clocking mechanism to align the inputs. In this specific case, we will apply a further spin current to the inputs that change the magnetization

orientation along the z-axis if it is different from the orientation of the remaining two. The results are shown in Figure A.6. All possible for the carrier and the sum combination are in the appendix A.2.
As expected, we observe that certain states are much more likely due to the inequality of the input distribution of a single gate.

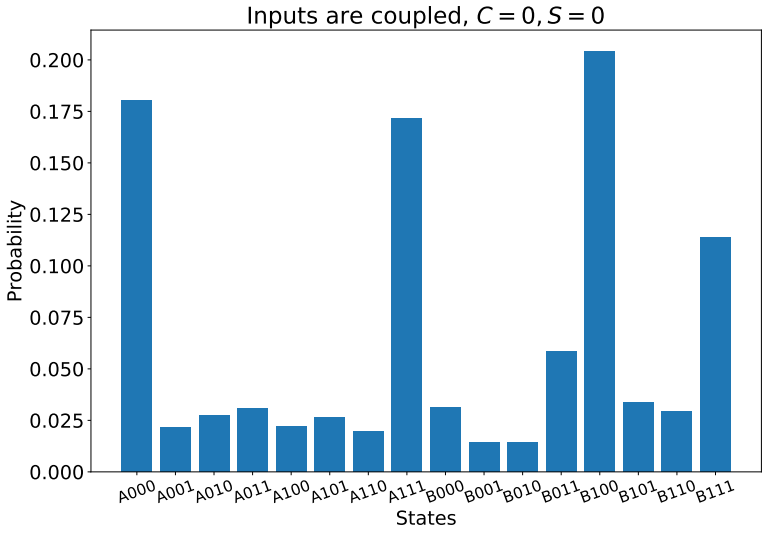


Figure 6.10.: Probability distribution of half adder states for the carrier $C = 0$ and sum $S = 0$. Bars show given states combinations for all three magnets which are acting as input A or B. A current is applied to couple the respective magnets of the Half adder gater inputs. Simulation parameter: $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

$$I_{bias} = I_C \cdot \text{sign}(\sum^{Inputs} m_z), \tag{6.2.1}$$

where $I_C = \frac{6q}{\hbar}\alpha k_B T$ is the strength of the pinning current.
Although we have been able to adjust the inputs, we still see that those with a high probability will take the wrong state. Nevertheless, we are still able to predict the most likely state correctly with a high probability.
It should be noted that scaling up these gates to complex structures will ultimately make it impossible to predict the most likely state of the system.
Based on the studies of this majority gate systems, we realized that we are looking for structures that tend to have a high logical consistency to minimize the accumulation of errors and have degenerated logical consistent states. Furthmore, we are looking for structures with a less connections as possible. For this reason, we are able to rely on a different gate design.

6.3. Inversion of a Well-Balanced Gate

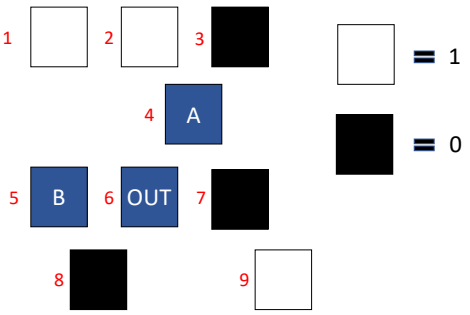


Figure 6.11.: Design and Numbering of the arrangment of the NAND gate.

Energies $[J/H_K]$	States [1,2,3,4,5]
-0.464449	110
-0.464449	011
-0.464449	101
-0.464449	001
-0.341932	111
0.049653	100
0.049653	010
0.686273	000

Table 6.2.: Terminal states of NAND gate in the Fig 6.11 including there dipole energy.

In the previous section, we discussed the stability of the majority gate in the case of inversion and found that it is not suitable for reversing the direction of operation. The decisive point was that the logically consistent states were not the basic states of stationary dipole interactions. As stationary, we refer to system states where the magnetizations of all magnets are aligned along the **z**-axis. In addition, one can see from the histogram of the OR gate and the dipole energies that it is not sufficient to require that the desired states have the lowest energies. One must also demand that they are degenerate. In 2018 Gypens Leliaert, Waeyenberge coined the term balancedness for this type of gates. In Ref. [53], they used fixed magnets to create dipole fields that stabilize the state of the terminals. They called the structures found well balanced as long as they were stable against thermal fluctuations, and the logically consistent states were degenerate. The degeneration allows a thermal change between the states. However, in contrast to Gypens et al., we are interested in coupling out-of-plane magnets together.
Under this premise, we have searched for a gate design that exhibits both the logically consistent states as minima of dipole energy and the degeneration of these states. From a system that has these two properties, we expect that the histogram of input probabilities is equally distributed.
Our search for a NOR gate starts with a gate that initially consists of three magnets that serve as terminals for the two inputs A and B, as well as the output. For this system, we had the same problem as for the majority gate, that the ground states are not degenerate.
As a result, we adapted the idea that additional magnets could be added to the system to stabilize the state of the terminals by means of their dipole fields. Thus we were

able to find a system consisting of nine magnets, three of which act as terminals, and the remaining six have a fixed magnetization that stabilizes the ground state of the system to the logically consistent states. Figure 6.11 shows this gate design.
In the table 6.11 the dipolar energies of the NAND gate design are shown. Similar to the majority gate, the gate can act es NOR gate by fliping the magnetization of the biasing magnets. We see that the logically consistent states that make up the four lowest inputs are degenerated.
We expect that through the degeneration of the logically consistent states, it should be possible to switch back and forth between the states with medium thermal fluctuations. Therefore we simulate this gate with the same parameters as for the majority gate. In contrast to the simulation of the majority gate, we do not apply bias current to the terminals but only bias the bias magnets that generate the dipole fields for stabilization. A histogram of such a simulation is shown in Figure 6.12

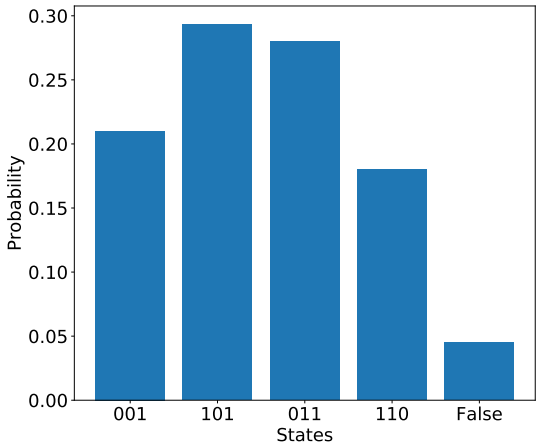


Figure 6.12.: Probability distribution of states of the NAND. Simulation parameter: $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

Although the logically consistent states are evenly distributed, the gate's susceptibility to errors is quite high, since the energy of state '111' is quite close to ground states and only requires a spin-flip. The choice of temperature determines the strength of the thermal fluctuations, so a reduction of the temperature leads to fewer switching events, and the probability of fluctuating to state '111' is reduced because more energy is needed to reach this state. Basically, this can also be observed for the majority gate, only that the most likely state '001' is assumed.

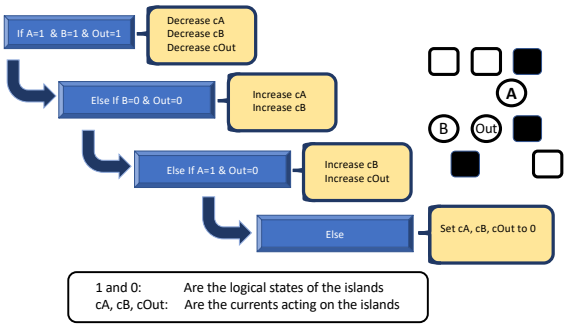


Figure 6.13.: Flowchart of clocking algorithm.

Now the question arises on how to reduce the error susceptibility of the gate. As already mentioned, it is possible to reduce the temperature, which in turn leads to a reduction of switching events and thus slows down the collection of statistics. However, slowing down the collection of statistics is counterproductive, since we need a large number of data points to be able to make a statement about the actual probability distribution of our gates. So it is not an option to reduce the temperature, as it is the engine that drives the system.
Unlike the forward logic that has been clocked in between readouts, this gate is continuously undergoing a transition from one state to the other. One possibility to reduce the error rate is to introduce a clocking mechanics similar to the forward logic.
Therefore, we have implemented a clocking that checks after a given time whether the terminals are in a logically consistent state. In case the terminal magnets are not in such a state, a current is applied, which increases the probability of transition to a logically consistent state. A flowchart of the algorithm is shown in Figure 6.13. As a test of the clocking, we set up a system where three free nanomagnets were clocked. A plot for different biasing currents which acts on the terminals and times is shown in Figure 6.14. We see that for large checking times, the three gates are in principle freely fluctuating while, for low coupling times, they are in logical, consistent states with a high probability. As a result the gates acts in this case as a NAND gate. Applying the same biasing scheme to the NAND gate design in Figure 6.15, we see that we can raise the logical consistency by $2\% - 4\%$ depending on the frequency of the checks. At this point, we should mention whenever the status of the system is checked more frequently we observe that the well-balancedness is decreasing, this is based on the fact that the biasing counteracts the random fluctuations whereas the previous logical, consistent state is favored.

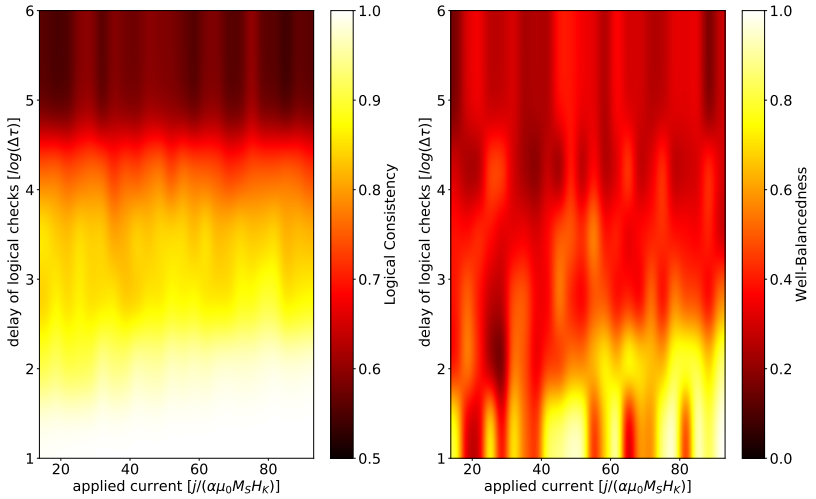


Figure 6.14.: Simulation of three uncoupled nanomagnets that act due to the biasing currents as a NAND gate. The time in which the logical consistency is checked as well as the current has been varied during the simulation. Simulation parameter: $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

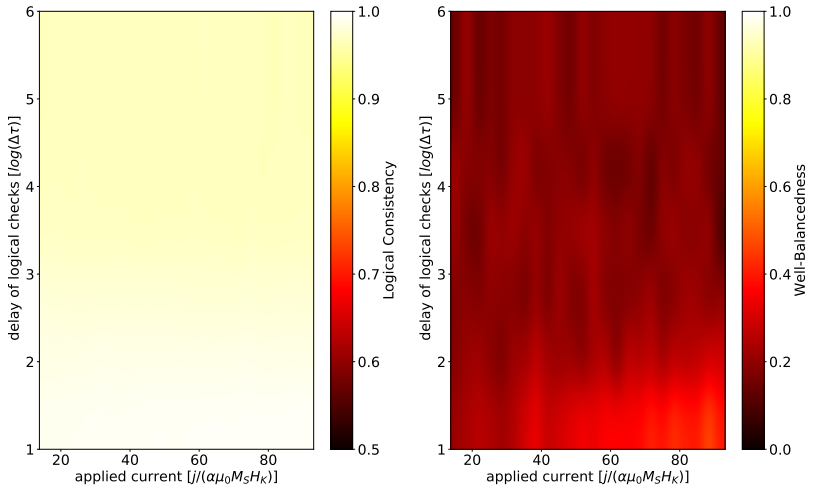


Figure 6.15.: Simulation of the NAND gate design. Logcial consistency and well-balancedness has been checked while the time for the logical check and current has been varied. Simulation parameter: $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

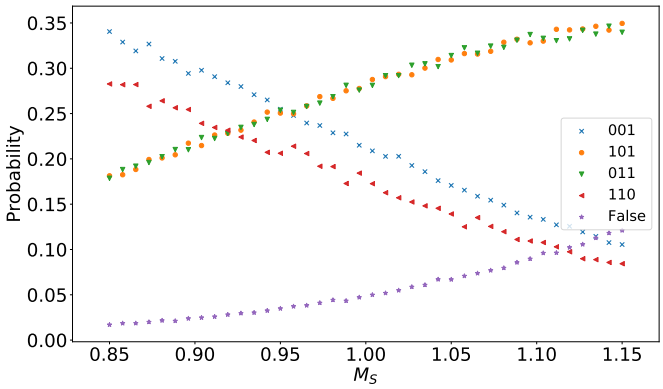


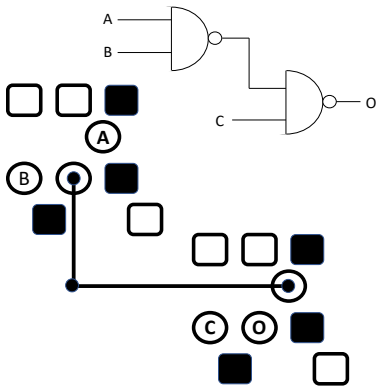
Figure 6.16.: Simulation of NAND gate where the M_S values of the surrounding biasing magnets are scaled.Simulation parameter: $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

Another possibility is the tuning of the dipole fields, which serve for stabilization. In our model, these are rescaled by the Stoner-Wohlfarth switching field

$$h_{dip} = H_{dip}/H_K = H_{dip}\mu_0 M_S/(2K_e V), \tag{6.3.1}$$

which makes them dependent on the parameters r, V, M_S and K_e. For simplicity, we have adjusted the parameter M_S of the stabilizing magnets, although, in experiments, one would rather change the volume or distance of the biasing magnets. To do this, we left the M_S factors of the terminals unchanged and manipulated the M_S factors of the biasing magnets. For each scale, we recorded a histogram as before. The results of the simulation are shown in Figure 6.16.
For small M_S values of the surrounding biasing magnets, we see that their influence is virtually negligible and that the dynamics are determined mainly by the interaction of the terminals. Thus we see that the dynamics mainly leads to either the state '001' or the state '110'. We would expect the same behavior from only three coupled magnets. Nevertheless, as we increase the influence of the biasing magnets, we see that the interaction of the three terminals less and less determines the dynamics until we reach a point where the probability of the four logically consistent states is about the same. Above that, the dynamics are increasingly determined by the biasing magnets, and we see that the system often falls into the states '101' and '011' or even bottle states, which results from the arrangement of the biasing magnets in Figure 6.11. The region in which the probability of the logical, consistent states are $22\% - 28\%$ is in between 0.92 to 0.95 of the M_S value of the terminal gates.
With this approach, we can influence the logical consistency of the system and, at the same time, keep the error rate low. Therefore we can speak of a fairly well-balanced NAND gate.

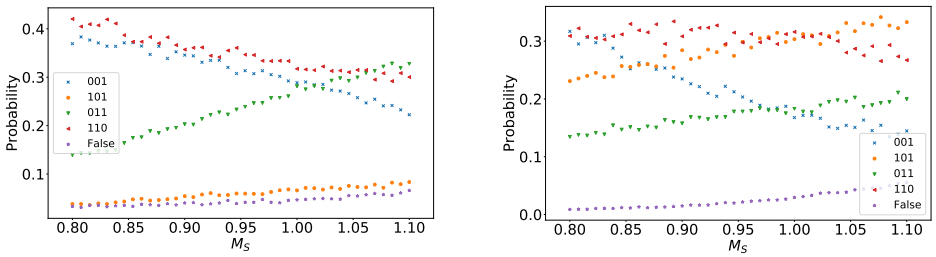
6.4. Interconnectability



A	B	C	O
0	0	0	1
0	0	1	0
0	1	0	1
1	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1
1	1	1	1

Figure 6.17.: Connection of two NAND gates with logical table.

Since the terminals of the gates are highly dependent on the dipole fields surrounding them, we cannot make a connection as we do in case of the majority gate by placing them next to each other to share terminals.
From a theoretical point of view, we can apply biasing currents to the connected terminals to make them the same as we did with the majority gate. This connection enables us to construct a circuit from NAND gates. The simplest circuit we can build from two NAND gates is shown in Figure 6.17, it connects the output of one NAND gate to one of the inputs of the other. The gate which is connected to the inputs is called the input gate and the gate which is connected to the outputs is called the output gate. The numerical simulations show that the connection of both



(a) Input Gate (b) Output gate

Figure 6.18.: Probability of the logical consistent states where the M_S-value of the biasing magnets is scaled compared to the terminal magnets. Simulation parameter: $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

gates has a significant influence on their behavior when the uncoupled terminals are allowed to fluctuate freely. The logical consistency is nearly unchanged, whereas the well-balancedness is changing significantly. The connection leads a suppression of the state '110' in the input gate, and in the output gate, the states with Input $A = 1$ are favored. This is expected since the NAND gate has three configurations in which the output is equal to '1'. Contrary to our expectations, the probability of 25% of the output in the input gate to be in a state '0' dropped to 2%, which will alter the well-balancedness of the connected gate as seen in Figure 6.17. Also, the simulation shows that the connection does not affect the logical consistency of the connected gate substantially; we are not able to speak of a well-balanced gate anymore. The application of the biasing algorithm will not change that fact since it mainly influences logical consistency.
At this point, we should mention that we studied the probability of the connected magnets to be orientated in the same direction while running the simulations shown in 6.19. We can show that with a probability of 97.3%, the connected gate is pointing in the same direction along the easy axis.

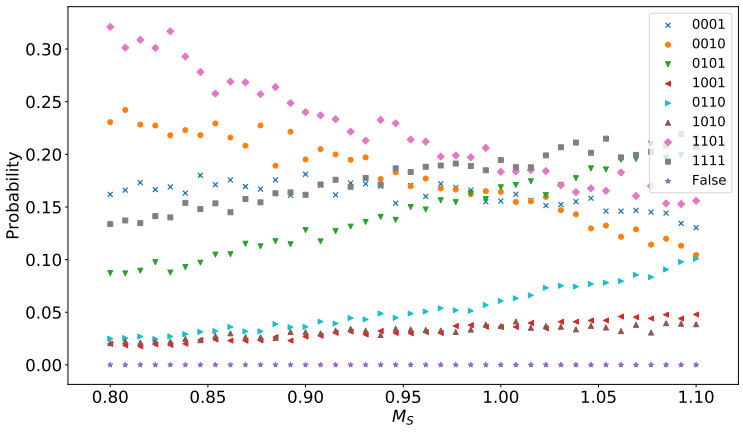


Figure 6.19.: Probability of the logical consistent states of the connected gate where the M_S-value of the biasing magnets is scaled compared to the terminal magnets.Simulation parameter: $\chi = 8$ and $M_S = 1e6\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

6.5. Inversion of the NAND Half Adder

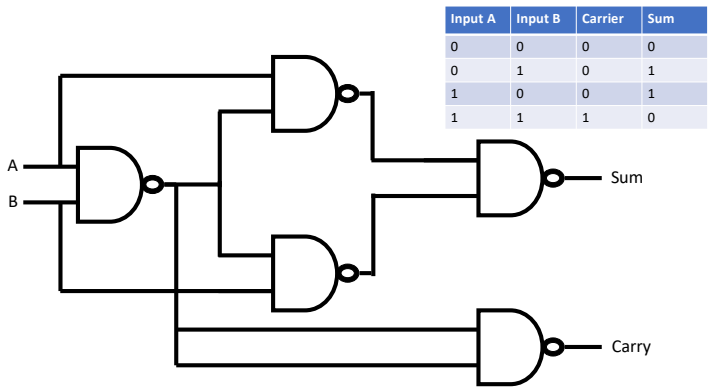


Figure 6.20.: Half adder design consisting of NAND gates.

In this section, we interconnect multiple NAND gates to a half adder. The half adder design 6.20 slightly differs from the previously shown design **??** because the XOR gate is build of three NAND gate instead of an OR, AND and NAND. The interconnection of the terminals is shown in Figure 6.21.

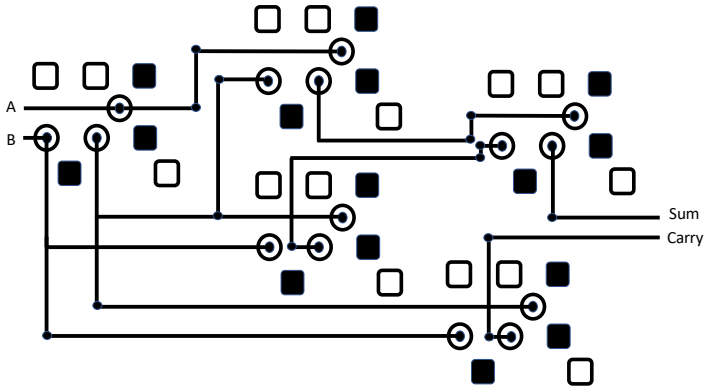


Figure 6.21.: Half adder design consisting of our NAND gate design.

Numerics in Figure 6.22 is showing that the gate is not acting probably. There are several reasons for this. One is that with the number of connections, the error probability of each one will accumulate. This ultimately leads to a lower probability of reaching a logically consistent state. Another reason is the way we have modeled the connection of multiple inputs to one output of the NAND. Since we assume that the majority of inputs determine the state of the output, as an example, we will show such a situation on the gate on the left side in Figure 6.21. Assuming that two of the three connected input terminals to the output of this gate are in a specific state can lead to the breaking of the logic because the third input terminal is not independent of the state of the others. The breaking can occur due to the connection of the gate to the input of the gate on the left-hand side that is not necessarily consistent with the output state. Thus the input is independent of the output value of the gate, which will lead to a similar situation as discussed in the spin chain case such that no proper logical computation or inversion is possible with this gate. Nevertheless, we see in Figure 6.22 that we can predict with a high probability a logically consistent state.

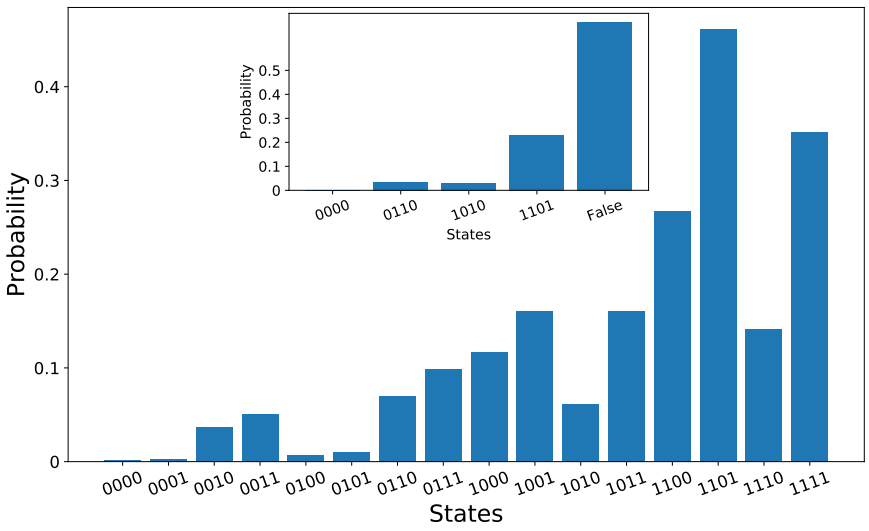


Figure 6.22.: States and logical states of the NAND half adder for a $M_S' = 0.94 \cdot M_S$, $\Delta\tau = 0.04\ s/\gamma' H_K s/\gamma H_K$, Simulation parameter: $\chi = 8$ and $M_S = 1000000\ Am^{-1}$, $\alpha = 0.04$, $\Delta\tau = 0.04\ s/\gamma' H_K$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

7. Conclusion

In this thesis, we investigated the stability of logic gates constructed from NML under the influence of thermal fluctuations. In particular, we were interested in investigating the reversal of the directional dependence of such gates. The starting point of this work was the work of Refs. [22, 3, 2, 5].
We extended the macrospin model by using the dipole-interaction to couple the gates. This coupling allowed us to simulate the majority gate shown in Ref. [3] and to reproduce the results. We could show that it is necessary to use a clock mechanism that can be implemented either by STT or by external fields. These majority gates, which act as OR/AND and NAND/NOR gates, were the basis for modeling more complex structures.
As a result, we have presented a half-added design, which consists of several majority gates. While investigating this connected gate, we encountered the problem that our gate design can get stuck in metastable states.
Therefore, we extend our model to include thermal fluctuations to overcome the metastable states while reversing the direction of action of the gates. It could be shown that the logistically consistent states of the majority gate are not degenerated basic states. As a consequence, it is more likely that the majority gates fluctuate into the energetically lower states, which are not necessarily logically consistent states. It is precisely this property of the majority gates that make it impossible to invert a more complex structure, since it cannot be ensured that the spectrum of logically consistent states of the more complex structure is covered, as the example of the half adder shows. Furthermore, this gate design showed that it is necessary to connect the multiple input terminals by currents to ensure that they have the same state.
Therefore, we have introduced a new type of NOR/NAND gate design that has degenerated ground states that are logically consistent. We were able to show that this gate can fluctuate with a high probability between the logically consistent states due to thermal fluctuations and that the distribution of these states is evenly distributed. In terms of Gypens et al. in Ref. [53] we have found a well-balanced gate that is invertible.
Furthermore, we could show that we were able to increase the logical consistency of this gate by means of another clocking mechanism acting on the biasing of the terminals of these gates by STT.
However, we could not show for this gate that linking these gates with more complex structures allows their inversion. This is mainly due to the fact that it is unclear how the connection from one output to several inputs can be modeled under inversion.
Especially, the interconnection mechanism from one input to multiple inputs in regards to the inversion of NML is interesting for further research.

List of Figures

List of Tables

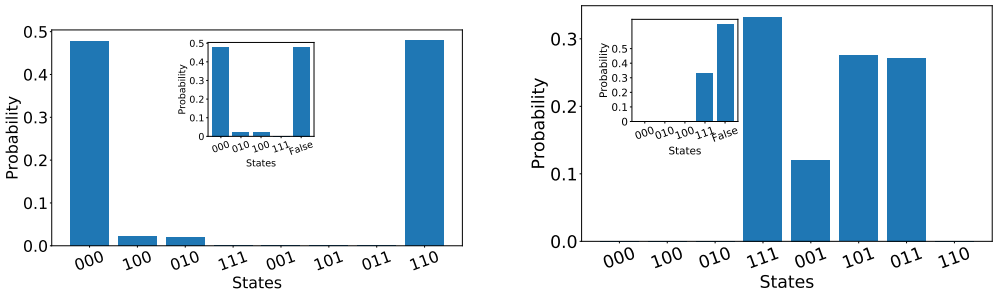
Listings

A. Appendix

A.1. Nomenclature

FDT	Fluctuation–Dissipation theorem
LL	Landau–Lifshitz equation
LLG	Landau–Lifshitz–Gilbert equation
LLGS	Landau–Lifshitz–Gilbert–Slonczewski equation
NML	Nanomagentic Logic
ODE	Ordinary differential equation
SDE	stochastic differential equation
sLLGS	stochastic Landau–Lifshitz–Gilbert–Slonczewski equation
SST	Spin-transfer torque
SWM	Stoner–Wohlfarth model

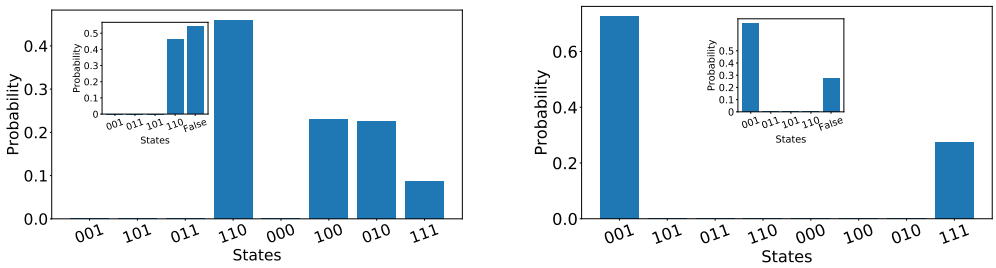
A.2. Additional figures



(a) AND gate with output set to 0.

(b) AND gate with output set to 1.

Figure A.1.: Probability distribution of the AND gate in the backwards operation direction.Simulation parameter 300 K with $K_e = 1e5\frac{J}{m^3}$ J/m, $V = 5 * 3 * 3$ nm^3 and $M_S = 1000000$ Am^-1 and a distance $r_{ij} = 2.5$ nm between the nanomagnets.



(a) NAND gate with output set to 0.

(b) NAND gate with output set to 1.

Figure A.2.: Probability distribution of the NAND gate in the backwards operation direction.Simulation parameter 300 K with $K_e = 1e5\frac{J}{m^3}$ J/m, $V = 5 * 3 * 3$ nm^3 and $M_S = 1000000$ Am^-1 and a distance $r_{ij} = 2.5$ nm between the nanomagnets.

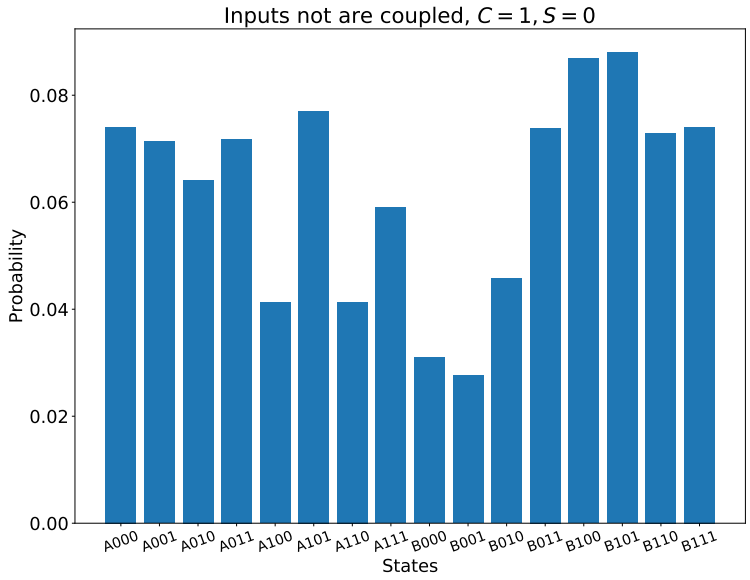


Figure A.3.: Probability distribution of half adder states for the carrier $C = 1$ and sum $S = 0$. Bars show given states combinations for all three magnets which are acting as input A or B.Simulation parameter 300 K with $K_e = 1e5\frac{J}{m^3}\ J/m$, $V = 5 * 3 * 3\ nm^3$ and $M_S = 1000000\ Am^-1$ and a distance $r_{ij} = 2.5\ nm$ between the nanomagnets.

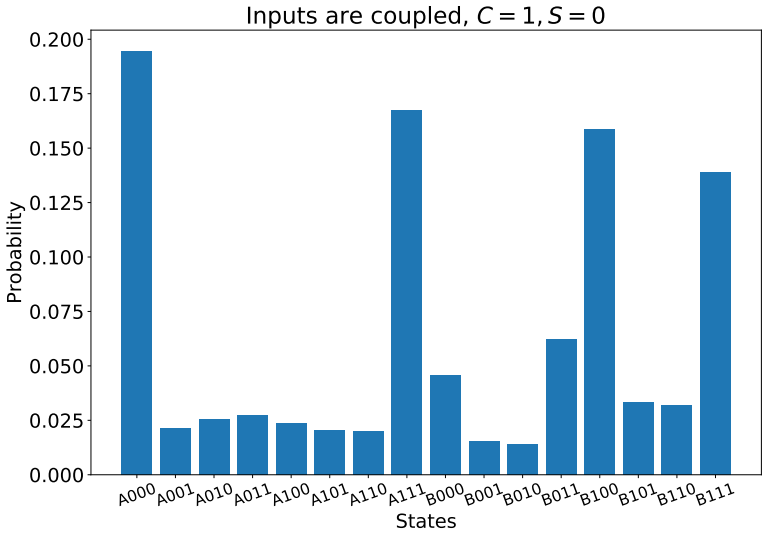


Figure A.4.: Probability distribution of half adder states for the carrier $C = 1$ and sum $S = 0$. Bars show given states combinations for all three magnets which are acting as input A or B. A current is applied to couple the respective magnets of the Half adder gater inputs.Simulation parameter 300 K with $K_e = 1e5\frac{J}{m^3}$ J/m, $V = 5 * 3 * 3$ nm^3 and $M_S = 1000000$ Am^-1 and a distance $r_{ij} = 2.5$ nm between the nanomagnets.

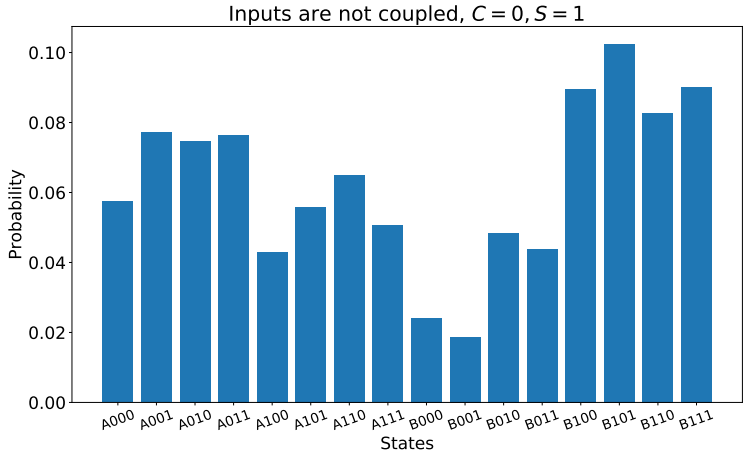


Figure A.5.: Probability distribution of half adder states for the carrier $C = 0$ and sum $S = 1$. Bars show given states combinations for all three magnets which are acting as input A or B.Simulation parameter 300 K with $K_e = 1e5\frac{J}{m^3}$ J/m, $V = 5 * 3 * 3$ nm^3 and $M_S = 1000000$ Am^-1 and a distance $r_{ij} = 2.5$ nm between the nanomagnets.

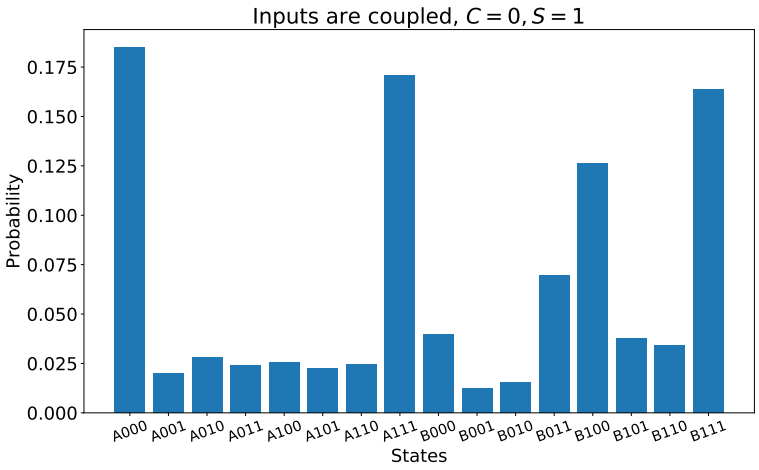


Figure A.6.: Probability distribution of half adder states for the carrier $C = 0$ and sum $S = 1$. Bars show given states combinations for all three magnets which are acting as input A or B. A current is applied to couple the respective magnets of the Half adder gater inputs.Simulation parameter 300 K with $K_e = 1e5\frac{J}{m^3}$ J/m, $V = 5 * 3 * 3$ nm^3 and $M_S = 1000000$ Am^-1 and a distance $r_{ij} = 2.5$ nm between the nanomagnets.

B. Bibliography

[1] D. B. Carlton, N. C. Emley, E. Tuchfeld, and J. Bokor, Nano Letters **8**, 4173 (2008).

[2] S. Kurtz, E. Varga, M. J. Siddiq, al, R. L. Stamps, S. Breitkreutz, J. Åkerman, M. T. Niemier, G. H. Bernstein, G. Csaba, A. Dingler, X. S. Hu, S. Liu, J. Nahas, W. Porod, and M. Siddiq, J. Phys.: Condens. Matter **23**, 493202 (2011).

[3] G. H. Bernstein, A. Imre, V. Metlushko, A. Orlov, L. Zhou, L. Ji, G. Csaba, and W. Porod, Microelectronics Journal **36**, 619 (2005).

[4] A. Imre, G. Csaba, L. Ji, A. Orlov, G. H. Bernstein, and W. Porod, Science **311**, 205 (2006).

[5] G. H. Bernstein, W. Porod, V. Metlushko, G. Csaba, and A. Imre, IEEE Transactions on Nanotechnology **1**, 209 (2003).

[6] R. P. Cowburn and M. E. Welland, Science **287**, 1466 (2000).

[7] M. T. Alam, S. J. Kurtz, M. A. J. Siddiq, M. T. Niemier, G. H. Bernstein, X. S. Hu, and W. Porod, IEEE Transactions on Nanotechnology **11**, 273 (2012).

[8] E. Varga, A. Orlov, M. T. Niemier, X. S. Hu, G. H. Bernstein, and W. Porod, IEEE Transactions on Nanotechnology **9**, 668 (2010).

[9] M. T. Niemier, E. Varga, G. H. Bernstein, W. Porod, M. T. Alam, A. Dingler, A. Orlov, and X. S. Hu, IEEE Transactions on Nanotechnology **11**, 220 (2012).

[10] K. Haughan, M. T. Niemier, W. Porod, and G. Csaba, in *2014 International Workshop on Computational Electronics, IWCE 2014* (IEEE Computer Society, 2014).

[11] A. Lyle, A. Klemm, J. Harms, Y. Zhang, H. Zhao, and J. P. Wang, Applied Physics Letters **98**, 092502 (2011).

[12] G. Csaba, P. Lugli, M. Becherer, D. Schmitt-Landsiedel, and W. Porod, Journal of Computational Electronics **7**, 454 (2008).

[13] V. Skumryev, S. Stoyanov, Y. Zhang, G. Hadjipanayis, D. Givord, and J. Nogués, Nature **423**, 850 (2003).

[14] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, Physical Review X **7**, 031014 (2017), arXiv:1610.00377 .

[15] M. T. Alam, S. J. Kurtz, M. A. J. Siddiq, M. T. Niemier, G. H. Bernstein, X. S. Hu, and W. Porod, IEEE Transactions on Nanotechnology **11**, 273 (2012).

[16] W. F. Brown, *Micromagnetic* (Interscience-Wiley, 1963).

[17] G. Bertotti, *Hysteresis in Magnetism - For Physicists, Materials Scientists and Engineers* (Academic Press).

[18] A. H. Morrish, *The physical principles of magnetism* (Wiley).

[19] N. W. Ashcroft and N. D. Mermin, (Oldenbourg Wissenschaftsverlag).

[20] D. Spisák and J. Hafner, Physical Review B - Condensed Matter and Materials Physics **55**, 8304 (1997).

[21] D. Spišák and J. Hafner, Journal of Magnetism and Magnetic Materials **168**, 257 (1997).

[22] D. Pinna, *Spin-Torque Driven Macrospin Dynamics subject to Thermal Noise*, Ph.D. thesis, New York University.

[23] R. Hertel, Zeitschrift fuer Metallkunde/Materials Research and Advanced Techniques **93**, 957 (2002).

[24] E. Condon and G. Shortley, *The Theory of Atomic Spectra* (Cambridge University Press).

[25] M. Getzlaff, *Fundamentals of Magnetism* (Springer).

[26] M. Levitt, *Spin Dynamics* (Wiley).

[27] J. H. Jensen, A. Strømberg, O. R. Lykkebø, A. Penty, M. Själander, E. Folven, and G. Tufte, (2020), arXiv:2002.11401 .

[28] S. E. C. Wohlfarth E. P., Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences **240**, 599 (1948).

[29] W. Scholz, T. Schrefl, and J. Fidler, Journal of Magnetism and Magnetic Materials **233**, 296 (2001).

[30] L. LANDAU and E. LIFSHITZ, in *Perspectives in Theoretical Physics* (Elsevier, 1992) pp. 51–65.

[31] T. L. Gilbert, IEEE Transactions on Magnetics **40**, 3443 (2004).

[32] V. Kamberský, Czechoslovak Journal of Physics **26**, 1366 (1976).

[33] J. A. Katine, F. J. Albert, R. A. Buhrman, E. B. Myers, and D. C. Ralph, Physical Review Letters **84**, 3149 (2000), arXiv:9908231 [cond-mat] .

[34] E. B. Myers, D. C. Ralph, J. A. Katine, R. N. Louie, and R. A. Buhrman, Science **285**, 867 (1999).

[35] J. C. Slonczewski, Journal of Magnetism and Magnetic Materials **159**, L1 (1996).

[36] D. C. Ralph and M. D. Stiles, "Erratum to "Spin Transfer Torques" [J. Magn. Magn. Mater. 320 (2008) 1190-1216] (DOI:10.1016/j.jmmm.2007.12.019)," (2009).

[37] S. Zhang, P. M. Levy, and A. Fert, Physical Review Letters **88**, 2366011 (2002).

[38] M. A. Zimmler, B. Özyilmaz, W. Chen, A. D. Kent, J. Z. Sun, M. J. Rooks, and R. H. Koch, Physical Review B - Condensed Matter and Materials Physics **70**, 1 (2004).

[39] R. H. Koch, J. A. Katine, and J. Z. Sun, Physical Review Letters **92**, 088302 (2004).

[40] W. F. Brown, Physical Review **130**, 1677 (1963).

[41] J. L. García-Palacios and F. J. Lázaro, Physical Review B **58**, 14937 (1998).

[42] H. Nyquist, Physical Review **32**, 110 (1928).

[43] H. B. Callen and T. A. Welton, Physical Review **83**, 34 (1951).

[44] E. Wong and M. Zakai, The Annals of Mathematical Statistics **36**, 1560 (1965).

[45] J. Honerkamp, *Stochastische Dynamische Systeme* (VCH, 1990).

[46] C. W. Gardiner, *Handbook of Stochastic Methods* (Springer).

[47] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations* (Springer).

[48] P. E. Kloeden and E. Platen, Stochastic Analysis and Applications **9**, 311 (1991).

[49] B.Øksendal, *Stochastic Differential Equations* (Springer).

[50] H. K. Leung, Physical Review A **41**, 1862 (1990).

[51] G. I. I. and S. A. V., *Stochastic Differential Equations* (Springer, 1972).

[52] J. Awrejcewicz, *Numerical Simulations of Physical and Engineering Processes* (InTech, 2012).

[53] P. Gypens, J. Leliaert, and B. Van Waeyenberge, Physical Review Applied **9**, 034004 (2018).

C. Code

The CUDA code is based on the code that Dr. Daniele Pinna used in his dissertation. In the course of this work, the dipole interaction and the control mechanisms for logic circuits were added.
Furthermore, a framework was created that allows the code to control the numerical evaluation dynamics in Python.

```

import pycuda.autoinit
import pycuda.driver as cuda
from pycuda import gpuarray
from pycuda.compiler import SourceModule

import sys
import timeit
import numpy as np


def mod_pos(posi,j_fix,scaling_factor):
    '''
    Modificies the M_S value of the not fixed magenets.

    Args:
        posi (double): Coupling matrix.
        j_fix (float): Array containing the current information.
        scaling_factor (float): Scaling factor for the M_S value.

    Returns:
        Rescaled coupling matrix.
    '''
    length_of_mag = len(j_fix)
    for i in range(length_of_mag):
        for j in range(1,length_of_mag/3+1):
            if j_fix[3*j-1]!=0:
                posi[i*length_of_mag+3*j-3]*=scaling_factor
                posi[i*length_of_mag+3*j-2]*=scaling_factor
                posi[i*length_of_mag+3*j-1]*=scaling_factor
    return posi


## Set GPU to calculate on.
pycuda.tools.get_default_device(0)

#Calculates coupling matrix.
def pos_array(pos,V,lengthscale):
    '''
```

```
    Calculates the coupling matrix based on the postions the length scale and the volume of the magnets..

    Args:
        pos (float): Array containting position array.
        V (float): Volume of the magnets.
        lengthscale (float): Lengthscale of the system.

    Returns:
       Coupling matrix.
    '''
    r =[]
    for i in (pos):
        r+=list(i)
    ret_mat = []
    for i in range(len(r))[::3]:
        row0 = []
        row1 = []
        row2 = []
        for j in range(len(r))[::3]:
            r_x = (r[j+0] - r[i+0])
            r_y = (r[j+1] - r[i+1])
            r_z = (r[j+2] - r[i+2])
            r_dist = (np.sqrt(r_x**2+r_y**2+r_z**2*lengthscale))**3
            print(r_dist)
            if i==j:
                row0+=[0,0,0]
                row1+=[0,0,0]
                row2+=[0,0,0]
            else:

                row0+=[(3*r_x**2-1)/r_dist,3*r_x*r_y/r_dist,3*r_x*r_z/r_dist]
                row1+=[3*r_y*r_x/r_dist,(3*r_y**2-1)/r_dist,3*r_y*r_z/r_dist]
                row2+=[3*r_z*r_x/r_dist,3*r_z*r_y/r_dist,(3*r_z**2-1)/r_dist]
        ret_mat+=list(row0)
        ret_mat+=list(row1)
        ret_mat+=list(row2)
    ret_mat = -np.array(ret_mat)*V/(4*np.pi)

    return ret_mat

##### number to binary number encoder.
    '''
    Calculates the binary number based on the number of digits and the natural number.

    Args:
        X (int): Number.
        n (int): Number of digits.

    Returns:
```

```
89          Binary number.
90      '''
91  get_bin = lambda x, n: format(x, 'b').zfill(n)
92
93  #### Initilize C++ Code.
94  mod =SourceModule(open("dipole.cu", "r").read(),no_extern_c=True)
95
96  #### Number of gates to simulate.
97  numberofgates = 1
98
99
100 ##### Coupling information of a certain magnet to another one.
101 coupling_information = [0 for i in range(numberofgates*9+3)]
102
103 ### Set biasing strength.
104 coupling_j = 1e-3
105
106 #### Geometry of the gate.
107 pos = [[0,0,0],[1,1.1,0],[1,-1.1,0],[2,0,0],[3,0,0],[4,1.1,0]
108         ,[4,-1.1,0],[4,0,0],[5,0,0],[6,0,0],[7,1.1,0],[7,-1.1,0],
109         [7,0,0],[8,0,0],[9,1.1,0],[9,-1.1,0],[9,0,0],[10.1,0,0]]
110
111
112 #### Total number of spins.
113 numberofspins = len(pos)
114
115 #### Number of spins per gate.
116 number_of_bits_per_gate = len(pos)/(len(connections))
117
118
119 #### Specifies GPU memory and function to calculate.
120 func_first = mod.get_function("advance_system_gpu")
121 gridsize = int(numberofspins*3/512)+1
122
123
124
125 #### Simulation parameter for the sLLGS.
126
127
128
129 alpha = 0.01                ### The Gilbert damping parameter.
130 t_step = 0.01               ### Natural timestep.
131 tmax = 550*t_step           ### Number of natural to evolve the system.
132 M_s = 1e6                   ### Saturation magnetization.
133 mu_0 = 1.2566*1e-6          ### Vacuum permeability.
134 K_ani = 3e5#                ### Easy-axis anisotropy.
135 r_ = 1e-9                   ### Lengthscale of the system.
136 V = 60*90*5*(1e-9)**3       ### Volume of a magnets.
137 kb = 1.38*1e-23             ### Boltzmann constant.
138 j_bias = 10                 ### Strength of the biasing current.
139 H_k = 2*K_ani/(M_s)         ### STW field.
140 Xi = (K_ani*V)/(kb*300)     ### Chi height of the energy barrier.
141
142
```

```
143
144 ### Copying positions, coupling, current strength and initial
    magnetization from main memory to GPU after defining their type for
    C++ and allocationg memeory on the GPU.
145 pos = np.array(pos)
146 new_positions =  np.array(dt.pos_array(pos,V,33.5*1e-9))/(H_k)
147 positons_gpu = cuda.mem_alloc(new_positions.nbytes)
148 cuda.memcpy_htod(positons_gpu, new_positions)
149
150
151 coupling_information = np.array(coupling_information).astype(np.int32)
152 coupling_information_gpu = cuda.mem_alloc(coupling_information.nbytes)
153 cuda.memcpy_htod(coupling_information_gpu, coupling_information)
154
155 j_fix_original = np.zeros(numberofspins*3)
156 j_fix  = (np.array(j_fix)).astype(np.double)*jbias
157 j_fix_gpu = cuda.mem_alloc(j_fix.nbytes)
158 cuda.memcpy_htod(j_fix_gpu, j_fix)
159
160 for i in range(numberofspins):
161     random_number = np.random.rand()*np.pi*2
162     magnetization[3*i] = np.sin(random_number)
163     magnetization[3*i+1] = np.cos(random_number)
164 magnetization  = (np.array(magnetization)).astype(np.double)#1e
165 initial_mag_gpu = cuda.mem_alloc(magnetization.nbytes)
166 cuda.memcpy_htod(initial_mag_gpu, magnetization)
167
168
169 ##### Allocates memory for the copy from GPU to main memory on the main
     memory.
170 test_out = np.empty_like(magnetization)
171
172
173 ##### Loop the numerical evolve the dynamics.
174 for i in range(400):
175     ####### Runs Dynamics on GPU.
176     ####### It takes the positons, magnetization, couupling_information
    ,  total number of spins, number of gates, number of bits per gate,
177     ####### terminal positions within a gate, time delay to check the
    logic, time step to check to logic, angles (Theta, Phi) for the STT,
178     ####### Xi, alpha, natural timestep and the total time to run the
    simulation.
179     func_first(positons_gpu, initial_mag_gpu ,coupling_information_gpu,
     j_array_gpu, np.int32(3*numberofspins),
180     np.int32(numberofgates), np.int32(number_of_bits_per_gate), np.
    int32(1),np.int32(2),np.int32(3), np.int32(time_delay), np.int32((i
    *6250)%time_delay),
181      np.float32(0), np.float32(0), np.float32(Xi),np.float32(alpha),
182      np.float32(t_step), np.float32(tmax), block=(256,1,1),grid=(
    gridsize,1,1))
183
184      #### Copy z-components back from GPU to main memory and save in
    formation.
185     cuda.memcpy_dtoh(test_out, initial_mag_gpu)
```

```
186     save_magnetic_state+= list(test_out[2::3])
187 
188 #### Saves z-components.
189 np.savetxt("mag_z.dat",save_magnetic_state)
```

Listing C.1: Python Code to run the Half adder dynamics

```
1  
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <assert.h>
5  #include <cuda.h>
6  #include <getopt.h>
7  #include <unistd.h>
8  #include <time.h>
9  #include <cmath>                  // for sqrt, emp
10 #include <fstream>                // for std::ofstream
11 #include <valarray>               // for std::valarray
12 #include <string>
13 #include <sstream>
14 #include <math.h>
15 #include <iostream>
16 #include <cstdlib>
17 #include <cmath>
18 #include <curand_kernel.h>
19 
20 
21 
22 extern "C"{
23 
24 // Simulation parameter.
25     __shared__ int numberofspins;
26     //__shared__ double dipole_m[30];
27 
28     __constant__ double alpha_d = 0.04;
29     __constant__ float D_d = 1.0f;
30     __constant__ float pi = 3.1415926536;
31 
32     __constant__ float h = 0;
33     __constant__ double dt = 0.0;
34     __constant__ double sigma = 0.0f;
35 
36     __constant__ float theta = 0.0f;
37     __constant__ float lambda = 0.5f;
38     __constant__ float phi = 0.0f;
39 
40   __shared__ float ang, lam, alpha;
41   __shared__ double sigma_b, t_step, B;
42 
43 
44     __device__ inline void range(double m[])
45     {
46     /* Normalize the magnetization.
47 
```

```
48          Args:
49              m (double): Magnetization of single macrospin.
50
51          Returns:
52              Normilized magnetization vector.
53      */
54      double temp, temp1;
55      temp1 = sqrtf(m[0]*m[0]+m[1]*m[1]+m[2]*m[2]);
56      temp = m[0];
57      m[0] = temp/temp1;
58      temp = m[1];
59      m[1] = temp/temp1;
60      temp = m[2];
61      m[2] = temp/temp1;
62      }
63
64
65      __device__ inline void dipole(double dip_ret[],double pos[], double
         cm[], int numberofspins, int row, int numberofgates, int
        numspinspergate)
66      {
67         /* Calculates dipole energy.
68
69          Args:
70              dip_ret (double): Dipole energy of action on a magnet.
71              pos (double): Coupling matrix.
72              cm (double): Magnetization vecotr.
73              numberofspins (int): Number of spins.
74              row (int): Row in the coupling matrix.
75              numberofgates (int):  Number of gates.
76              numspinspergate (int): Number of magnets per gate.
77
78          Returns:
79              Dipole interaction acting on a macrospin.
80      */
81      double temp;
82
83      int N = 3*numspinspergate;
84      int startingpoint = row/numspinspergate;
85      for(int i=0;i<3;i++){
86        temp = 0;
87        for (int j=0;j<N;j++){
88            temp += pos[j+(startingpoint)*N+numberofspins*(i+row*3)]*cm[j
       +(startingpoint)*N];
89          }
90        dip_ret[i] += temp;
91        __syncthreads();
92        }
93  }
94
95  __device__ inline void drift(double nx[], double m[], double J, float
       ang, float lam, float alpha, double b_dip[],float D_d)
96          /*
97           Calculates the determentistic drift term.
```

```
98
99          Args:
100             nx (double): Return array.
101             m (double): Magnetization vecotr.
102             numberofspins (int): Number of spins.
103             J (double): Currents strength.
104             ang (float): Angle in between z- and x-axis.
105             lam (float): Angle in between x- and y-axis.
106             alpha (float): The Gilbert damping parameter.
107             b_dip (double): Dipole field.
108             D_d (float): Hard- to easy-axis ratio.
109
110         Returns:
111             Deterministic drift term.
112     */
113 {
114
115   double omega=atan(tan(ang)*(1-lam*m[2]));
116   double any=sin(ang);
117   double anx=0;
118   double anz=cos(ang);
119   double bext[3],ene[3];
120   double I = J;
121   int B    = -1;
122   double n = sin(lam);
123   double nsqrt = sqrt(1-n*n);
124
125   bext[0] = B*b_dip[0];
126   bext[1] = B*b_dip[1];
127   bext[2] = B*b_dip[2];
128
129   bext[0] = -D_d*m[0]+bext[0];
130   bext[1] = bext[1];
131   bext[2] = m[2]+bext[2];
132
133   double dot = bext[1]*m[1]+m[2]*bext[2]+m[0]*bext[0];
134   double dot_current = (m[1]*any+m[2]*anz);
135
136   nx[0] = (-bext[2]*m[1]+ bext[1]*m[2])
137   +alpha*I*(-m[0]*dot_current)
138   -alpha*(-bext[0]+m[0]*dot);
139
140   nx[1] = (-bext[0]*m[2] + bext[2]*m[0])
141   +alpha*I*(any-m[1]*dot_current)
142   -alpha*(-bext[1]+m[1]*dot);
143
144   nx[2] = (-bext[1]*m[0] + bext[0]*m[1])
145   +alpha*I*(anz-m[2]*dot_current)
146   -alpha*(-bext[2]+m[2]*dot);
147
148
149   }
150
151   __device__ inline void diff1(double ndm1[], double m[], float alpha)
```

```
152 {
153    /*
154       Calculates the first column of the diffusion matrix.
155       Args:
156           nx (double): First column of the diffusion matrix.
157           m (double): Magnetization vecotr.
158           numberofspins (int): Number of spins.
159           alpha (float): The Gilbert damping parameter.
160
161       Returns:
162           First column of the diffusion matrix.
163   */
164
165   ndm1[0] = 1.0f*(alpha*(1-m[0]*m[0]));
166   ndm1[1] = 1.0f*(-m[2]-alpha*m[1]*m[0]);
167   ndm1[2] = 1.0f*(m[1]-alpha*m[2]*m[0]);
168 }
169
170 __device__ inline void diff2(double ndm2[], double m[], float alpha)
171 {
172   /*
173       Calculates the second column of the diffusion matrix.
174       Args:
175           nx (double): Second column of the diffusion matrix.
176           m (double): Magnetization vecotr.
177           numberofspins (int): Number of spins.
178           alpha (float): The Gilbert damping parameter.
179
180       Returns:
181           Second column of the diffusion matrix.
182   /*
183
184   ndm2[0] = 1.0f*(m[2]-alpha*m[1]*m[0]);
185   ndm2[1] = 1.0f*(alpha*(1-m[1]*m[1]));
186   ndm2[2] = 1.0f*(-m[0]-alpha*m[1]*m[2]);
187 }
188
189 __device__ inline void diff3(double ndm3[], double m[], float alpha)
190 {
191     /*
192       Calculates the third column of the diffusion matrix.
193       Args:
194           nx (double): Third column of the diffusion matrix.
195           m (double): Magnetization vecotr.
196           numberofspins (int): Number of spins.
197           alpha (float): The Gilbert damping parameter.
198
199       Returns:
200           Third column of the diffusion matrix.
201   */
202   ndm3[0] = 1.0f*(-m[1]-alpha*m[0]*m[2]);
203   ndm3[1] = 1.0f*(m[0]-alpha*m[1]*m[2]);
204   ndm3[2] = 1.0f*(alpha*(1-m[2]*m[2]));
205 }
```

```
206
207 __device__ inline double connect_gates(double cm[],double m[], double
    IsC, double biasing, int coup1, int bo)
208   {
209        /*
210       Adjust biasing on the connected magnets.
211       Args:
212           cn (double): Array of magnetization.
213           m (double): Magnetization vecotr.
214           IsC (double): Current acting on the magnets.
215           biasing (double): Increment of the current.
216           coup1 (int): Check to which magnet the magnet is connected.
217           bo (int): Check if there is a connection.
218
219
220       Returns:
221           Adjusted current strenght.
222   */
223   if ((cm[3*(coup1)-1]>0 && m[2]>0) || (cm[3*coup1-1]<0 && m[2]<0)){
224       IsC = 0;
225     }
226   else{
227     if ((m[2]+cm[3*coup1-1])<0){
228       IsC += biasing;
229     }
230     else{
231       IsC -= biasing;
232     }
233   }
234   return IsC;
235 }
236
237
238 __device__ inline void logic_check_nand(double m[],double jm[], int
   numberofspins,
239   int numberofgates, int number_of_bits_per_gate, int inputA,int
   inputB, int output, int bo)
240 {
241       /*
242       This function is doing the BIASING of the islands according to
   NAND gate logic.
243       Args:
244           m (double): Array of magnetization.
245           jm (double): Array of current action on the magnets.
246           numberofspins (int): Number of spins.
247           numberofgates (int):  Number of gates.
248           numspinspergate (int): Number of magnets per gate.
249           inputA (int): Position of input A in a gate.
250           inputB (int): Position of input B in a gate.
251           output (int): Position of output in a gate.
252           bo (int): Check if there is a connection.
253       Returns:
254           Adjusted current strenght.
255   */
```

```
256     int InA = 0;
257     int InB = 0;
258     int Out = 0;
259     int temp;
260     double j_bias;
261     /// This function is doing the BIASING of the islands according to
        NAND gate logic
262     for(int i=0;i<numberofgates;i+=1){
263       temp = i*number_of_bits_per_gate;
264       j_bias = jm[3*(temp+inputA)-3];
265       if ((m[3*(temp+inputA)-1]<0.) && (bo==1)){
266         InA =  0;
267       }
268       else if (m[3*(temp+inputA)-1]>0.){
269         InA = 1;
270       }
271       if (m[3*(temp+inputB)-1]<0.){
272         InB =  0;
273       }
274       else if (m[3*(temp+inputB)-1]>0.){
275         InB = 1;
276       }
277       if (m[3*(temp+output)-1]<0.){
278         Out = 0;
279       }
280       else if (m[3*(temp+output)-1]>0.){
281         Out =  1;
282       }
283       if ((Out==0) && (bo==1)) {
284         if (InA==0) {
285           jm[3*(temp+inputA)-1] += j_bias;
286           jm[3*(temp+output)-1] += j_bias;
287         }
288         if (InB==0){
289           jm[3*(temp+inputB)-1] += j_bias;
290           jm[3*(temp+output)-1] += j_bias;
291         }
292       }
293       if ((InA==1 && InB==1 && Out==1) && (bo==1)){
294         jm[3*(temp+inputA)-1] -= j_bias;
295         jm[3*(temp+inputB)-1] -= j_bias;
296         jm[3*(temp+output)-1] -= j_bias;
297       }
298       if (((InA==1  && InB==1 && Out==0) || (InA==1  && InB==0 && Out
        ==1)  || (InA==0  && InB==1 && Out==1)  || (InA==0  && InB==0 && Out
        ==1))){
299         jm[3*(temp+inputA)-1] =0.;
300         jm[3*(temp+inputB)-1] =0.;
301         jm[3*(temp+output)-1] =0.;
302
303       }
304
305     }
306   }
```

```
307
308
309
310
311
312 __global__ void advance_system_gpu(double *pos, double *cm, int *
      coupling, double *j, int coupling_switch, int numberofspins,
313     int numberofgates, int number_of_bits_per_gate, int inputA, int
      inputB, int output, int counter, int counter2,
314     float ang, float lam,float epsilon, float alpha, float t_step,
      float t_max)
315 {
316
317   /*
318       This function is numerical evolving the system based on the
      sLLGS.
319       Args:
320           pos (double): Coupling matrix.
321           cm (double): Array of agnetization vecotr.
322           couling (double): Array of couling information vecotr.
323           j (double): Array of current action on the magnets.
324           coupling_switch (int): Enables couling.
325           numberofspins (int): Number of spins.
326           numspinspergate (int): Number of magnets per gate.
327           inputA (int): Position of input A in a gate.
328           inputB (int): Position of input B in a gate.
329           output (int): Position of output in a gate.
330           counter (int):  Delay time of time checks.
331           counter2 (int): Time checks.
332           ang (float): Angle in between z- and x-axis.
333           lam (float): Angle in between x- and y-axis.
334           epsilon (float): Height of the energy barrier.
335           alpha (float): The Gilbert damping parameter.
336           b_dip (double): Dipole field.
337           t_step (float): Size of natural time step.
338           t_max (float): End time.
339       Returns:
340           Returns evolved dynamics.
341   */
342   double m[3], mim[3], mt1[3], mdt1[3], mt2[3], mdt2[3],
343          mdt3[3], umdt1[3], umdt2[3], umdt3[3], dipole_m[3], jx[3];
344   float  n1, n2, n3, sigma_b;
345   float t;
346   double Ie, Is,lamz,lamy, IsC  , IeC, jbias;
347   int N = numberofspins/3, coup1, coup2, coup3, boolint, tempgate;
348   float D_d = 0.f;
349   ### Defines each macrospin to a thread.
350   unsigned int idx = blockIdx.x * blockDim.x  + threadIdx.x;
351
352   ### Draw a random number
353   curandState cr_state;
354   curand_init((unsigned long long)clock() + idx, 0, 0, &cr_state);
355   if (epsilon>10000){
356     sigma_b = 0;
```

```
357     }
358     else{
359       sigma_b = sqrt(alpha/((1+alpha*alpha)*epsilon));
360     }
361
362     t = 0.f;
363
364     if ((idx<N)){
365       m[0] = cm[3*idx];
366       m[1] = cm[3*idx+1];
367       m[2] = cm[3*idx+2];
368
369       jbias = j[3*idx];
370
371       Ie = j[3*idx+2];
372       Is = 0;
373       IsC = 0;
374       boolint = 0;
375
376       jx[0] = 0;
377       jx[1] = 0;
378       jx[2] = 0;
379
380       //counter2 =0;
381
382       coup1 = coupling[3*idx];
383       coup2 = coupling[3*idx+1];
384       coup3 = coupling[3*idx+2];
385
386       while (t<t_max && abs(Ie)<1000){
387         n1 = curand_normal(&cr_state);
388         n2 = curand_normal(&cr_state);
389         n3 = curand_normal(&cr_state);
390         dipole_m[0] = 0;
391         dipole_m[1] = 0;
392         dipole_m[2] = 0;
393         dipole(dipole_m, pos, cm, numberofspins, idx, numberofgates,
    number_of_bits_per_gate);
394
395         //
    ////////////////////////////////////////////////////////////////////////////////////////
396         //  Checks for time delay. If TRUE apply biasing, else reset
    logic when in the correct statee
397         //
    ////////////////////////////////////////////////////////////////////////////////////////
398         if (coup2==2){
399           counter2+=1;
400           if (counter2%counter==0){
401             boolint=1;
402             counter2=0;
403           } else{
404             boolint=0;
```

```
405         }
406         logic_check_nand(cm, j, numberofspins, numberofgates,
    number_of_bits_per_gate, inputA, inputB, output, boolint);
407       }
408       //
    ////////////////////////////////////////////////////////////////////////////////

409       //  Checks for I/O to be in the align if not bias them
410       //
    ////////////////////////////////////////////////////////////////////////////////

411
412       tempgate = idx/number_of_bits_per_gate;
413       if (coupling_switch>0){
414
415         if (((cm[3*(coup1)-1]<0 && m[2]>0) || (cm[3*coup1-1]>0 && m
    [2]<0)) && (coup1>0)&& (counter2%counter==0)){
416
417           IsC = connect_gates(cm , m,  IsC, jbias,  coup1,  1);
418
419           }
420         if (((cm[3*(coup2)-1]<0 && m[2]>0) || (cm[3*coup2-1]>0 && m
    [2]<0)) && (coup2>0)&& (counter2%counter==0)){
421           IsC = connect_gates(cm , m,  IsC, jbias,  coup2,  1);
422
423           }
424         if (((cm[3*(coup3)-1]<0 && m[2]>0) || (cm[3*coup3-1]>0 && m
    [2]<0)) && (coup3>0)&& (counter2%counter==0)){
425
426           IsC = connect_gates(cm , m,  IsC, jbias,  coup3,  1);
427
428           }
429       }
430
431       counter2+=1;
432       __syncthreads();
433
434
435       ////////////////////////////////////////
436       ///////// HEUN SCHEME /////////////////
437       ////////////////////////////////////////
438       Is = j[3*idx+2]+IsC;
439       __syncthreads();
440
441       drift(mt1, m, Is, ang, lam, alpha, dipole_m, D_d);
442
443       diff1(mdt1, m, alpha);
444       diff2(mdt2, m, alpha);
445       diff3(mdt3, m, alpha);
446
447       mim[0] = m[0] + mt1[0] * t_step + sqrtf(t_step) * sigma_b * (
    mdt1[0]*n1+mdt2[0]*n2+mdt3[0]*n3);
448       mim[1] = m[1] + mt1[1] * t_step + sqrtf(t_step) * sigma_b * (
    mdt1[1]*n1+mdt2[1]*n2+mdt3[1]*n3);
```

```
449         mim[2] = m[2] + mt1[2] * t_step + sqrtf(t_step) * sigma_b * (
    mdt1[2]*n1+mdt2[2]*n2+mdt3[2]*n3);
450
451         range(mim);
452
453         cm[3*idx]   = mim[0];
454         cm[3*idx+1] = mim[1];
455         cm[3*idx+2] = mim[2];
456
457         dipole_m[0]=0;
458         dipole_m[1]=0;
459         dipole_m[2]=0;
460
461         dipole(dipole_m, pos, cm, numberofspins, idx, numberofgates,
    number_of_bits_per_gate);
462
463         __syncthreads();
464         drift(mt2, mim, Is, ang, lam, alpha, dipole_m, D_d);
465         diff1(umdt1, mim, alpha);
466         diff2(umdt2, mim, alpha);
467         diff3(umdt3, mim, alpha);
468
469
470         m[0] += 0.5f*t_step*(mt1[0]+mt2[0])+0.5f*sqrt(t_step)*sigma_b
    *((mdt1[0]+umdt1[0])*n1+(mdt2[0]+umdt2[0])*n2+(mdt3[0]+umdt3[0])*n3)
    ;
471         m[1] += 0.5f*t_step*(mt1[1]+mt2[1])+0.5f*sqrt(t_step)*sigma_b
    *((mdt1[1]+umdt1[1])*n1+(mdt2[1]+umdt2[1])*n2+(mdt3[1]+umdt3[1])*n3)
    ;
472         m[2] += 0.5f*t_step*(mt1[2]+mt2[2])+0.5f*sqrt(t_step)*sigma_b
    *((mdt1[2]+umdt1[2])*n1+(mdt2[2]+umdt2[2])*n2+(mdt3[2]+umdt3[2])*n3)
    ;
473
474         range(m);
475
476         cm[3*idx]   = m[0];
477         cm[3*idx+1] = m[1];
478         cm[3*idx+2] = m[2];
479         if (abs(j[3*idx+2])>1000){
480           j[3*idx+2] =0 ;
481         }
482         if (abs(j[3*(tempgate+inputB)-1])>1000){
483           j[3*(tempgate+inputB)-1] =0;
484         }
485         if (abs(j[3*(tempgate+output)-1])>1000){
486           j[3*(tempgate+output)-1] =0;
487         }
488
489         __syncthreads();
490         t+=t_step;
491
492     }
493   }
494 }
```

```
495 }
```

Listing C.2: CUDA

D. Acknowledgment

I would like to thank my supervisor Dr. Karin Everschor-Sitte for providing me this interesting topic, the even more interesting conversations, and finally, for the support during the year of my master's research. Other special thanks go to Dr. Daniele Pinna, who was able to give a reliable answer to every question and supported me at all times. During this time, they opened up unimagined possibilities for me within physics, regarding my topic, but also outside of science.
Furthermore, I like to thank Dr. Ricardo Zarzuela for the lesson and hints on scientific writing.
I thank Benjamin McKeever and Venkata Krishna Bharadwaj for their help on problems regarding CUDA and coding. I thank my dear office colleague Nils Sommer for his suggestions and the discussions about my topic.
I would like to extend my thanks to the entire TWIST Group for the support and for providing a computer to run my simulations.
Furthermore, I would like to thank Prof. Dr. Mathias Kläui second reviewer of this thesis.
Lastly, I would like to thank my friends and family for their continuous support over the years of my studies.