NTNU
Norwegian University of
Science and Technology

# Multiple Power Domains

**Ivar Håkon Lysfjord**

Master of Science in Electronics
Submission date: June 2008
Supervisor: Einar Johan Aas, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

# Problem Description

Assignment
The assignment is divided into three tasks:

- System design consideration
- Synchronization mechanisms between power domains
- Specification of a Power Management unit

System Design Consideration
A multiple power domain technology introduces new innovative system design possibilities. As always you cannot get everything so weighing pros against cons and come up with the best possible solution with today's technology is necessary.

The first task is to come up with a system design for a new AVR microcontroller using multiple power domains.

Synchronization mechanisms between power domains
Dealing with multiple power domains where each domain has its own clock source, results in a requirement of a synchronization mechanism, to make it possible to exchange data between the domains. Level shifters can easily adapt the physical difference in voltage and transfer signals over the domain boundary, but data buses that operate at different speed cannot be directly connected to each other via level shifters.

The second task is to design an efficient synchronization mechanism between two data buses operating at different clock speeds.

Specification of a Power Management unit
The advantage with multiple power domains is that you can turn off one or more domains to save power. Very useful and even necessary in low power applications, such as battery powered products where low power consumption is one of the most important parameters. Turning on and off power domains must however be done in a controlled way, thus no failures or unwanted data losses occur. A Power Management Unit (PMU) must be designed to handle all different sleep modes and how the power to the different domains shall be controlled.

The final task is to make a specification of a Power Management Unit.

Assignment given: 15. January 2008
Supervisor: Einar Johan Aas, IET

## Preface

To get a good insight in this report, it is necessarily to have good understanding of electronics and physics of integrated circuits.

This master assignment has been laborious and educating, and the 20 weeks has passed very quickly. The work has given me great knowledge about how much work that has to be done just to make the specification of a large digital circuit like a microcontroller.

To start designing a new project is the dream of an engineer. The possibility to create a system from my ideas ensures me that I have chosen the right education. When born with a creative brain, I need challenges that make me really use all the knowledge I have gained through the years at school, this assignment is such a task.

The assignment is just a specification, or the start of a specification of a microcontroller system that uses multiple power domains. There is still work to be done before the microcontroller can be made. To get a complete system up and running, moths, maybe years of research and coding is needed. This is though a good start, here some of the pitfalls are mentioned, and some of the advantages are explained. The system does have potential, now the design needs to go into phase two.

The assignment has some formulas with different variable, see Appendix A - List of variables to see the variables used in the equations.

I would like to thank my advisor Einar J. Aas, a professor at NTNU, for all the help he has given me through the assignment, and in the years before.

My advisor at Atmel Norway, Jonas Ehn has been outstanding, and has given me all the help I have requested. Without him, I could not have gotten this far in this assignment.

I would also like to thank my co students for the help they have given me. They have been helpful both in answering my questions, and also to been a great help during the entire education.

## Summary

When new transistor technology is used in a microcontroller design, the transistors become smaller. They cannot withstand the same voltages as older technology, because of their size. The automotive industry still uses 5V as a standard voltage, and the automotive industry is a major costumer for microcontroller companies. The microcontroller must therefore be able to use5V. This must be done without the need of external voltage regulator. To still be a supplier to the automotive industry, the AVR needs to be able to withstand voltages up to 5.5V.

The main problem with the new transistor technology is the leakage currents. Traditionally, the CMOS devices have used power only when during switching of logical levels. This is no longer true, since the leakage currents have become so large. When using new transistor technology, the dynamic power usage will be reduced, but the total power usage will be increased, if nothing is done to prevent it.

One solution to this is to make a multiple power domain microcontroller. The idea is that one power domain can withstand voltages up to 5.5V. The microcontroller then uses an internal voltage regulator to scale down the voltage to a suitable level. The low voltage area will then have a suitable voltage level, which reduces both the dynamic- and leakage power usage. The different voltage domains uses different clock sources, so communicating between them requires both level shifters to deal with the different voltage levels, and synchronization logic to prevent metastability.

This assignment uses two voltage domains, $V_{IO}$ and $V_{CORE}$. Since voltage regulators are quite inefficient, it is most efficient to use only two domains. The $V_{CORE}$ domain contains most of the digital logic of the microcontroller, such as the CPU, SRAM and timers. This domain uses a high-speed clock source, and a $V_{CORE}$ data bus to communicate between each other. To communicate with the $V_{IO}$ domain, the data bus is connected to the $V_{IO}$ data bus through an asynchronous communication scheme block. This is because the $V_{IO}$ domain uses a low speed clock source. The usage of individual clock sources prevents clock skew problems that may occur when passing level shifters, and there is power saving by using only a low speed clock source on the $V_{IO}$ domain.

The VIO domain contains the Power Management Unit (PMU). The PMU shall control the power usage of the microcontroller. During active mode, the PMU can set unused modules in sleep mode, or shut them completely off. Most of the power savings are during sleep mode though. This is because a microcontroller such as the AVR spends most of the time in sleep mode. To reduce the power usage in sleep mode, the leakage currents needs to be reduced. The best way of doing so is to disconnect the power from the circuits. If the voltage regulator is disconnected, and all the inputs are set to high impedance, the $V_{CORE}$ domain is completely disconnected from the power, and uses absolutely no power. An asynchronous wake up circuit is designed to make it possible to wake up the microcontroller from a sleep mode without the usage of synchronized digital logic. Then the low frequency oscillator can be turned off, and even more power is saved.

The major disadvantage of the multiple power domain solution is the start up time from a sleep mode. If the power to the low voltage area is disconnected, the start up requires that all the capacitors become charged before the chip can start running again. The oscillator is shut off, and it takes time to stabilize the oscillator. Especially since the oscillator requires some stability in the voltage, and the voltage may not stable until the capacitors are charged.

Simulations shows that the multiple power domain solution has great potential of power saving. The proposed asynchronous wake up circuit uses only 1.2275nA. This is significantly smaller than the AVR uses in the deepest sleep mode today. To get a secure microcontroller, a reset circuit has to be on to be able to reset the AVR if necessary. The power usage of the reset circuit used today is confidential Atmel information, and cannot be published in this assignment. By looking at the data sheet of a pico power circuit of the AVR, the ATmega329p, one can see that in the deepest sleep mode, the microcontroller uses 40nA at 1.8V. By assuming that the reset circuit does not use more that half of this current, the total amount of power that saved during a sleep mode by using the multiple power domain solution is about 47%.

# Table of content

# 1 - Introduction

As technology evolves, more and more functionality are implemented on each chip. This functionality is given through more and more transistors, and as the number of transistors increase, the power consumption increases.

This has now become a real issue for digital designers, because the increase in power consumption means that if the chips are used in handheld devices, the battery time decreases, and hence the quality of the product. The heating must also be considered, because heat increases as the power consumption increases, and if the chips are heated too much, they will be destroyed. One cannot depend on heat sinks if the devices are meant to be small. The best way of solving the problem is to make power efficient hardware. This can be done at several stages, but this assignment will look into the digital designer's way of solving the problem.

Formula 1.1 shows the power usage in digital CMOS circuits. The dominating part of the total power consumption has been the dynamic power usage, but as technology has evolved, the leakage currents cannot be ignored anymore. To be able to reduce the power usage of digital CMOS circuits, one cannot only look into how to decrease the dynamic power; one must also decrease the leakage.

$$\boxed{P_{avg} = P_{leakage} + P_{SC} + P_{dynamic}}$$

**Formula 1.1 - CMOS Power consumption [2]. $P_{leakage}$ is due to leakage currents in the transistors. $P_{SC}$ is due to the direct-path short-circuit current occurring if both the PMOS and NMOS devices are conducting simultaneously. $P_{dynamic}$ is the dynamic power usage, which is due to charging load capacitances.**

$$\boxed{P_{SC} = V_{DD} \bullet I_{SC}}$$

**Formula 1.2 - The short circuit power is proportional to the supply voltage $V_{DD}$.**

There are several approaches to be able to decrease the power, and this assignment will look deeper into voltage scaling. The methods researched in this assignment is meant to be used on the AVR microcontroller, and since the microcontroller is commonly used in designs where the operating voltage is 5V or 3.3V, the microcontroller must be able to operate within these external voltages without adding extra components for voltage control and level shifting. One solution to this is to use multiple power domains on the chip. This way one core domain can operate at a desired low voltage, and an I/O domain, which operates at higher voltage, can be used to communicate with the peripherals. By using lower voltage at the core, one can reduce the power usage. By scaling down the voltage, one reduces all three contributions to the total power consumption. By looking at Formula 1.2, Formula 2.1 and Formula 2.2, one can see that $V_{DD}$ is contributing to all three parts of the total power consumption.

As the transistors become smaller, the voltage must be scaled down, because the smaller transistors cannot withstand high voltage. The voltage scaling is then not only desirable, but also imperative.

Besides looking at the advantages of scaling down the voltage internally on a chip, there are other advantages of using multiple power domains. As Figure 5.1 shows, most of the power on a microcontroller is generated in sleep mode. The use of multiple power domains makes it possible to reduce the power in sleep mode to a minimum (5.1.8 - Asynchronous Wake Up).

By choosing a multiple power domain design with possibility of turning off parts, one need to implement a power management unit (PMU) to control the sleep and power off modes (see 5.1 - Sleep- and Power Off –Modes), and to activate/wake up modules/parts when desired. Waking up from a sleep/power off mode takes time. To get an efficient microcontroller, this time must be minimized (see 5.1.6 - Start Up Time).

Designing a multiple power domain system is a big task. Since power consumption is one of the most important parameters when designing circuits today, this is the main concern of this assignment. This assignment will concentrate on reducing the sleep mode power, and try to accomplish a circuit to wake up the microcontroller without the use of a clock source. This should decrease the power usage considerably, since there will be no need of an oscillator running.

# 2 - Background

This chapter contains background information needed for the assignment. To understand why this multiple power domains are used, this chapter is useful.

## *2.1 - Low power*

Lowering power consumption in electronic devices is one of the key challenges of today. This is especially important when designing embedded system consisting of one or more microcontrollers. In batteries drive devices, it is very important that the lifetime of the battery is as long as possible. In order to reduce the power consumption one need to know what causes it.

### 2.1.1 - Leakage currents in CMOS

For a long time CMOS devices were considered to have very high impedance, and the static power consumption were close to zero, at least so much smaller than the dynamic power consumption that it could be ignored. This is no longer real, because as the transistor sizes have decreased, the leakage currents have increased. As a result, the leakage currents are a significant contributor to the total power consumption. Figure 2.1 shows the leakage currents in a NMOS transistor.

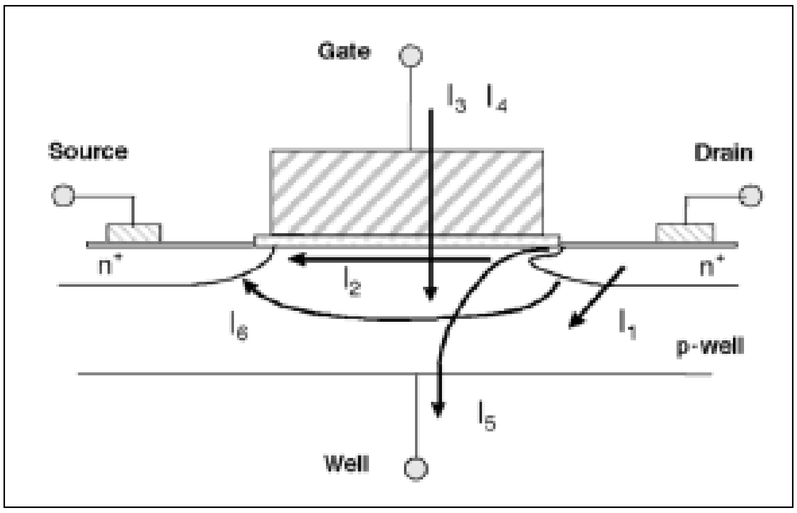


**Figure 2.1 - Leakage currents [1]**

**$I_1$** is the reverse-bias pn junction leakage current.
**$I_2$** is the subthreshold leakage current. This current is the leakage between the drain and source when the transistor is in off mode ($V_{GS} < V_{th}$ for NMOS). The current can be reduced by having a negative voltage at the gate, but as the current is quite small, this may not be worthwhile. This is the most significant leakage current [14]. The current reduces linearly with $V_{DD}$.
**$I_3$** is the tunneling current into and through the gate oxide. This current has increased as the gate oxide has decreased. The electric field results in electrons from the substrate tunneling through to the gate and also from the gate to the substrate.
**$I_4$** is injection of hot carriers from substrate to gate oxide. This is due to the high electric field near the Si–SiO interface in short channel MOS transistors.
**$I_5$** is the gate-induced drain leakage current, and is a result of the high electric field in the gate/drain overlap region. This current occurs at high $V_D$ and low $V_G$. Thinner oxide layer increases this current.

**$I_6$** is the punchtrough current. This occurs as the channel is getting smaller, and the depletion regions of the drain and source as so close that there is almost no channel between, and as a result majority carriers can cross the energy barrier, and create the leakage current. This current may be a major contributor to the leakage currents in new CMOS technology, as the channel lengths are reduced. The punchtrough can be reduced by doping the surface higher than the bulk, thus the punchtrough occurs at the surface.

The leakage power is given by adding all the currents into one, $I_{leakage}$, and multiplying by $V_{DD}$.

$$\boxed{P_{leakage} = V_{DD} \bullet (I_1 + I_2 + I_3 + I_4 + I_5 + I_6)}$$

**Formula 2.1 - Leakage power [1]**

## 2.1.2 - Clock gating

In clocked CMOS devices, the data change occur at clock flanks (usually only at the rising edge). The dynamic power usage is data change, and only present at clock flanks. This power usage is unnecessary if the module is not used. In order to stop this one has two solutions, one can stop the clock from reaching the module, or one can turn off the power of the module. These solutions are called clock and power gating, and are a very effective way of decreasing power usage in CMOS devices. The clock gating is commonly used, and stops the dynamic power usage of the devices. Figure 2.2 shows the clock gating principle applied to a D flip flop.

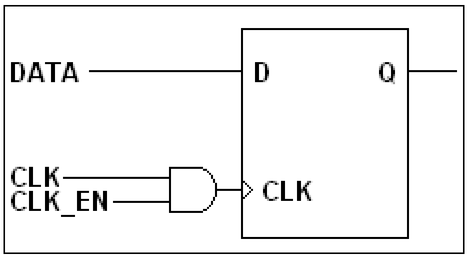


**Figure 2.2 - Simple clock gating by adding an AND gate. This illustrates the clock gating principle.**

## 2.1.3 - Power gating

As technology evolves, the transistors become smaller and smaller, and the leakage currents become larger. This can be stopped by gating the power of the modules, as a consequence, one cannot store any data in the module. This shows that power gating must be used carefully, so unnecessary loss of data does not occur. Power gating can be applied to both the combinatorial and clocked parts of CMOS circuits.

The easiest way of disconnecting the power to gates is by using sleep transistors. The most common way of using sleep transistors is to disconnect the logic from the supply ($V_{DD}$) and GND ($V_{SS}$). This is illustrated in Figure 2.3 a).

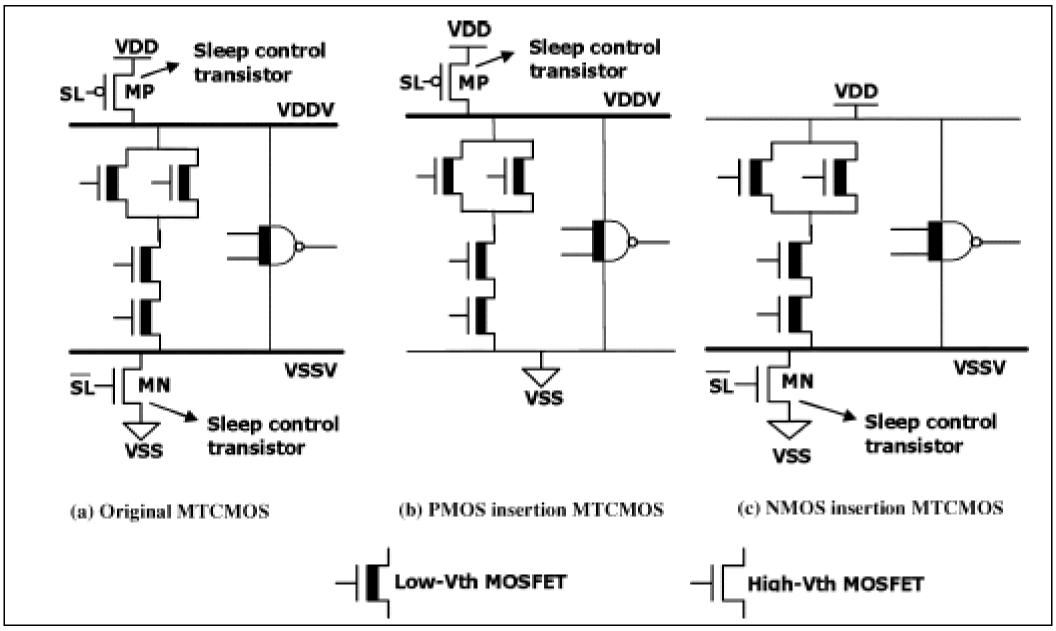


**Figure 2.3 - Sleep transistors [1]. Sleep transistors isolate the digital circuit from the supply, resulting in lower leakage currents. a) shows the most common use, where one NMOS is used to disconnect the logic form $V_{SS}$, and one PMOS is used to disconnect the logic from $V_{DD}$. b) and c) shows the use of only one sleep transistor.**

(MTCMOS = Multithreshold-voltage CMOS, see chapter Threshold voltage scaling)

Figure 2.3 b) and c) illustrates the use of only one sleep transistor per gate. This might be enough for leakage control since the sleep transistors are scaled to have low leakage. This scaling result in slower transistors, but this is unproblematic since the transistors are used in sleep mode, and does not require fast shifting. If only one sleep transistor is used (Figure 2.3 b) or c)), it is preferable to use c), since the NMOS transistors have lower resistance at the same width [1].

To be able to use the sleep transistors, one needs a unit to control the sleep signal. It is obvious that one cannot have individual sleep signal to every gate in the design, as this will make it impossible to control, and the wiring will be close to impossible. To make the design most useful, one partitions the logic belonging together and applies the same sleep signal to all gates in the partition. In microcontrollers, the design is partitioned into modules such as timers and communication modules.

Another way of gating the power is to shut off the power completely to the logic. This can be done by using different voltage islands (see 3.1 - Voltage ). If one shuts down the power to a whole voltage island, and keep all the signal connections and buses at high impedance, we have a *complete off mode* with no leakage. This use of power gating can be done when the chip is set in a sleep (5.1.1 - Sleep Modes) or *power off mode* (5.1.2 - Power Off Mode). The power can be disconnected from the power source by using several parallel gating transistors, the number of transistors must be chosen according to the amount of current flowing through.

### 2.1.4 - Voltage scaling

$$\boxed{P_{dynamic} = \frac{1}{2} C_{eff} \bullet V_{DD}{}^{2} \bullet f_{clk} \bullet \alpha}$$

**Formula 2.2 - Dynamic power [2]. The dynamic power in a MOS transistor is dependent on the frequency, the supply voltage and the effective capacitance. The α is the activity factor, which tells us about how often the flip flops or gates will change value (0 → 1 or 1 → 0).**

One can see from Formula 2.2 that the dynamic power usage of CMOS devices is proportional to the square of the supply voltage. Hence, the easiest way to decrease the power usage in CMOS devices is to scale down the supply voltage. This is not without consequences though, as the delay through the CMOS devices increases as the voltage decreases. As a result, there must be a choice where the middle path is found, where the power usage is low, and the delay is acceptable.

$$\boxed{I_D \approx K(V_{GS} - V_{TH})^2}$$

**Formula 2.3 - Drain current in CMOS [3, p25]. $V_{GS}$ can be replaced by $V_{DD}$ in digital circuits when calculating the current. $V_{GS}$ is $V_{DD}$ when the transistor is on and zero when the transistor is off.**

$$\boxed{K \approx \frac{\mu_n \bullet C_{ox}}{2} \bullet \frac{W}{L}}$$

**Formula 2.4 - $I_D$ constant [3, p25]. This constant is transistor parameter dependent.**

The delay through CMOS devices is shown in Formula 2.5. In digital circuits, the $V_{GS}$ is either $V_{DD}$ when on, or zero when off. When inserting Formula 2.3 in Formula 2.5, we get Formula 2.6, and we can see that the delay is depended on the supply voltage, the threshold voltage and a constant K, and all these parameters are positive. The constant K is process parameters, and will not be changed by changing the supply voltage, neither will the threshold voltage. When $V_{DD}$ decreases, the denominator will decrease more than the numerator, and the result is an increasing $t_d$.

$$\boxed{t_d = \frac{Q}{I_D} = \frac{C_L \bullet V_{DD}}{I_D} = \frac{C_L \bullet V_{DD}}{K(V_{GS} - V_{TH})^2}}$$

**Formula 2.5 - Delay in CMOS 1 [2]**

$$\boxed{td \approx \frac{CL \bullet VDD}{K(V_{DD}{}^{2} + 2 \bullet V_{DD} \bullet V_{TH} + V_{TH}{}^{2})}}$$

**Formula 2.6 - Delay in CMOS 2. When inserting $V_{DD}$ for $V_{GS}$, this is the delay in a MOS transistor when you know the $V_{DD}$, the $V_{TH}$, the load capacitance $C_L$ and the process parameters.**

The power consumption caused by the leakage currents will drop linearly with the voltage drop, as the static power consumption is the supply voltage, $V_{DD}$, multiplied by the static current, $I_{LEAKAGE}$ (See Formula 2.1).

## 2.1.5 - Threshold voltage scaling

The threshold voltage of a MOS transistor is referred to as the amount of voltage needed to open the channel from drain to source so that the transistor is conducting. The threshold voltage is in fact the voltage gap between the gate and the source that makes the concentration of electrons under the gate equal to the number of holes in the substrate (in NMOS) [3]. This voltage is defined by process parameters such as doping and size.
In CMOS devices, the bilk node is usually connected to the source ($V_{DD}$ at PMOS devices, and $V_{SS}$ at NMOS devises). Formula 2.8 shows how the threshold voltage will change when applying a voltage at the bulk node that is different from this. This is called the body effect, and can be used to change the threshold voltage either up or down.

$$\boxed{V_{tn0} = V_{fb} + 2\psi_B + \frac{\sqrt{4\varepsilon_{si} q N_A \psi_B}}{C_{OX}}}$$

**Formula 2.7 - Threshold voltage of NMOS transistors [1]. The threshold voltage is dependent on several process parameters, and cannot be adjusted by the digital designer.**

$$\boxed{V_{tn} = V_{tn0} + \gamma(\sqrt{V_{SB} + |2\Phi_F|} - \sqrt{|2\Phi_F|)}}$$

**Formula 2.8 – Body-effect [2]. When the bulk node of the MOS transistor is connected to a voltage different from the source node, one get a effect called body effect. This effect is used to change the threshold voltage. The $V_{SB}$ is the difference in voltage from source to bulk node.**

$$\boxed{\gamma = \frac{\sqrt{2qN_A K_S \varepsilon_0}}{C_{OX}}}$$

**Formula 2.9 - Body-effect constant [2]**

MTCMOS is a definition where we have MOS transistors with different threshold voltages. This requires either different doped transistors, or the usage of the body effect. To be able to use the body effect, triple well technology must be used. As we can see from Formula 2.8, a lower voltage at the bulk (at NMOS) will increase the threshold voltage $V_{th}$. This results in lower leakage current, but a slower transistor. To be able to increase the body voltage we need a negative power supply.

Another way of changing the threshold voltage is different doping. Two well-known doping methods are “Halo doping” and “Retrograde doping” [1].

The transistors we want to change the threshold voltage for are usually sleep transistors. These transistors do not need to be fast, and shall have as low leakage current as possible. This is done by raising the threshold voltage.

## 2.2 - Level shifting

When two, or more, voltage levels are present in digital circuits, we need to implement a device that can convert the voltage level differences. This has to be done with as little delay as possible, and the shifting should use small amounts of power.

### 2.2.1 - Up-Down (Bidirectional) Level shifter

A simple level shifter can be made by one single NMOS transistor. Figure 2.4 shows a level shifter implemented on an $I^2C$ bus. This bus system has two lines, but the principle can easily be converted to any bus standard, or single transmission lines.

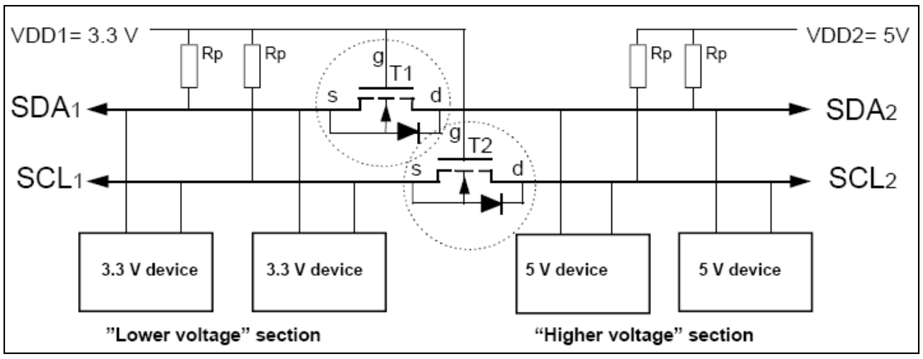


**Figure 2.4 - Simple bidirectional level shifter [6]. This principle shows how to use MOS transistors to compensate for different levels of voltage. This principle uses resistors, which take a lot of area when designing an integrated circuit, and is therefore not the best solution.**

The level shifter will work as long as $V_{DD1}$ is somewhat higher than the threshold voltage for the transistors $T_1$ and $T_2$. $V_{DD2}$ can be as high as the CMOS devices can withstand, but with higher $V_{DD2}$ we must expect slower *falling edge* on the bus lines, as the discharging time is higher.

"The level shifter for each bus line is identical and consists of one discrete N-channel enhancement MOSFET, $T_1$ for the serial data line SDA and $T_2$ for the serial clock line SCL. The gates (g) have to be connected to the lowest supply voltage $V_{DD1}$, the sources (s) to the bus lines of the "Lower voltage" section, and the drains (d) to the bus lines of the "Higher voltage" section. The diode between the drain (d) and substrate is inside the MOSFET present as n-p junction of drain and substrate." [6]

Figure 2.4 shows a level shifter using discrete components, and in discrete MOSFET transistors, the bulk node is coupled to the source node. When integrating MOSFET transistors, we normally couple the bulk node to the lowest possible voltage, $V_{SS}$/GND.

For the level shift operation three states have to be considered, this is described in [6]:

1. No device is pulling down the bus line and the bus line of the "Lower voltage" section is pulled up by its pull-up resistors Rp to 3.3 V. The gate and the source of the MOSFET are both at 3.3 V, so its $V_{GS}$ is below the threshold voltage and the MOSFET is not conducting. This allows that the bus line at the "Higher voltage" section is pulled up by its pull-up resistor Rp to 5V. So the bus lines of both sections are HIGH, but at a different voltage level.

2. A 3.3 V device pulls down the bus line to a LOW level. The source of the MOSFET becomes also LOW, while the gate stay at 3.3 V. The $V_{GS}$ rises above the threshold and the MOSFET becomes conducting. Now the bus line of the “Higher voltage” section is also pulled down to a LOW level by the 3.3 V devices via the conducting MOSFET. So the bus lines of both sections become LOW at the same voltage level.

3. A 5 V device pulls down the bus line to a LOW level. Via the drain-substrate diode of the MOSFET the “Lower voltage” section is in first instance pulled down until $V_{GS}$ passes the threshold and the MOSFET becomes conducting. Now the bus line of the “Lower voltage” section is further pulled down to a LOW level by the 5 V devices via the conducting MOSFET. So the bus lines of both sections become LOW at the same voltage level.

There are some disadvantages to this solution. First the pull-up resistors will take up a lot of area on an integrated chip. This could be solved using PMOS transistors. If we consider Rp(high_v) as the pull up at the high voltage area and Rp(low_v) as the pull ups at the low voltage area, the Rp(high_v) needs to be active only when the signal on the low voltage side is high. If then the pull up is realized by a PMOS, the signal on the low voltage side can be inverted, and control the PMOS. This way the pull-up is only active when needed. The problem is that the signal on the low voltage side is not large enough to open the pull up, so we need level shifting to this signal too. The Rp(low_v) pull-up can be replaced by a PMOS and drive by the inverted high voltage signal.
So the level shifter is not as simple and small as it seems, but the level shifter is quite fast. The static power consumption is low, as the modules connected to the bus are at high Z. As the lines are kept at $V_{DD}$ in the static case, the leakage currents in the level shifting transistor are the only currents flowing. And because there are a limited amount of these transistors in the design, the power consumption should be quite low.

This level shifter is not commonly used in integrated circuit design, but illustrates the purpose of a level shifter.

### 2.2.2 - Level Up shifter 1

A conventional integrated CMOS level shifter (CLS) is shown in Figure 2.5. The level shifter shifts from low voltage to high voltage. $V_{DDH}$ is the high supply voltage and $V_{DDL}$ is the low supply voltage. The node “in” is the low voltage signal, and the node “out” is the high voltage signal.

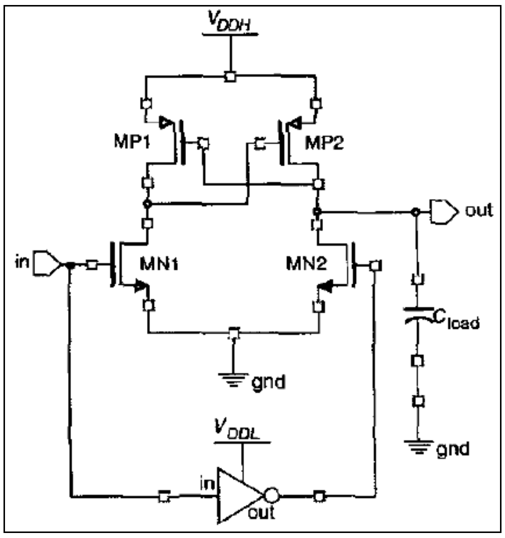


**Figure 2.5 - Conventional CMOS level shifter [9]. This level shifter shows how to create a simple level shifter using MOS transistors. When the in node is high, the MN1 is open, and the gate at MP2 is grounded, and opens MP2. The gate of MN2 is low, and closes MN2, this result in a high at the gate of MP1, and MP1 is closed. The inverse happens if the in node is low. The output follows the input.**

The pull-down NMOS (MN1) has to overcome the PMOS latch (MP1 and MP2) action before the output changes state. In addition, the PMOS gate experiences full voltage swing from 0V to $V_{DDH}$. As a result, high short circuit current flows during the switching, leading to excessive power dissipation [9].

### 2.2.3 - Level Up shifter 2

Because of the high power consumption of level shifter 1, there has been developed a low power level shifter using bootstrapping technique [9]. This level shifter is shown in Figure 2.6.

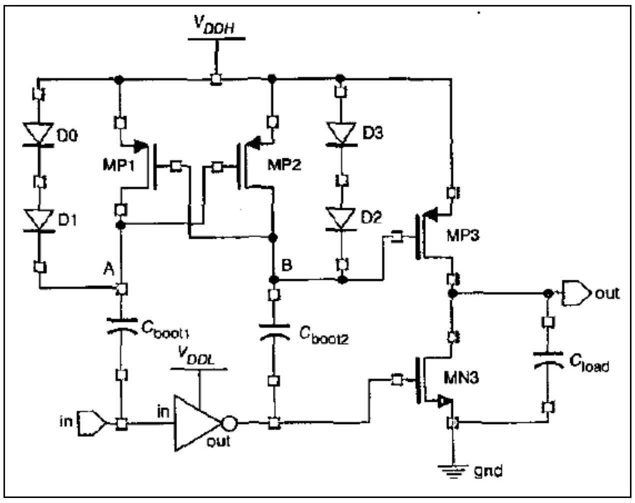


**Figure 2.6 - Low power level shifter using bootstrap technique [9]. The bootstrap technique reduces the power consumption of the level shifter by reducing the short current between $V_{DD}$ and GND during switching.**

The bootstrap technique reduces the power consumption, and slightly decreases the delay. For a completely operation of level shifter 2, see [9].

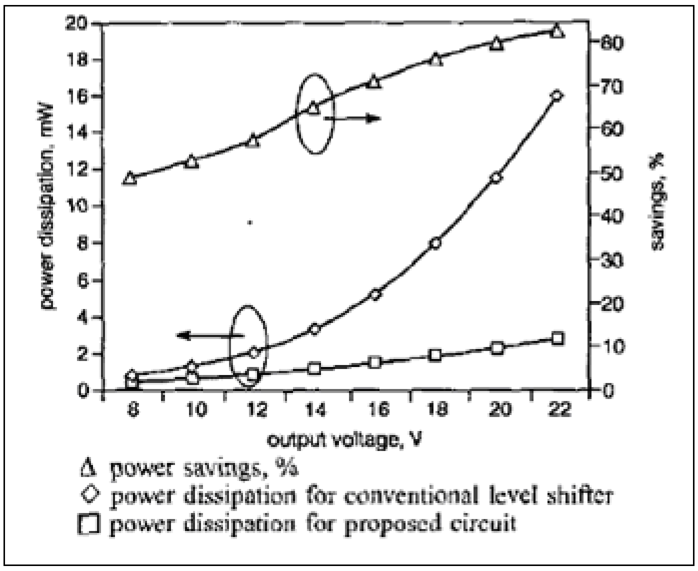


**Figure 2.7 - Power saving conventional vs. bootstrapped level shifter [9]. The proposed circuit is shown in Figure 2.6.**

Figure 2.7 shows the power saving of the bootstrapped level shifter implemented in 1.5µm technology, the low voltage is held constant at 5V. We see that the amount of power saving is most when the voltage gap between the two voltage levels is high. The voltage levels shown in Figure 2.7 are much higher than the ones used today, but the principles are the same.

Figure 2.8 shows a measured waveform of the two level shifters implemented in 1.5µm technology.

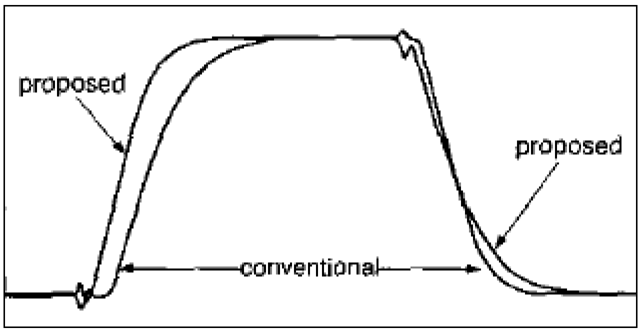


**Figure 2.8 - Waveform of conventional vs. bootstrapped level shifter [9]. The bootstrapped circuit has much faster response.**

### 2.2.4 - Level Up shifter 3

The frequency has increased; the level shifters need to be faster. Power consumption is another critical parameter. Several level shifters have been developed to meet these demands. To decrease both delay and power consumption is conflicting, since fast shifting requires much power. To deal with this problem, the complexity of the level shifters has increased. One such level shifter is shown in Figure 2.9.

In order to achieve high performance in a chip consisting of logic blocks having different $V_{DD}$ voltages, the proposed circuit uses a circuit technique to reduce the capacitive loading of input signals and to minimize the contention between pull-up and pulldown transistors through positive feedback loop [10].

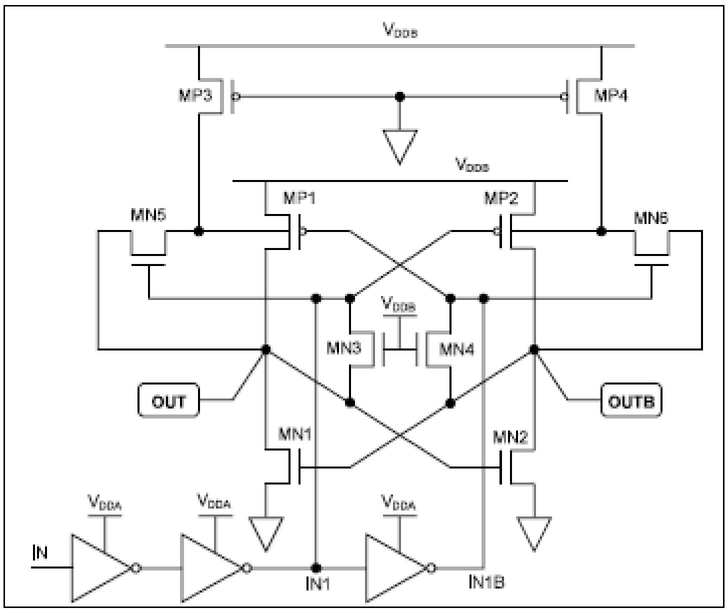


**Figure 2.9 – High speed level shifter [10]. This level shifter reduces power consumption and delay when using ultra low voltages.**

In the figure, $V_{DDA}$ is the low voltage, and $V_{DDB}$ is the high voltage. Simulation of delay is shown in Figure 2.10, and simulation of power consumption is shown in Figure 2.11.

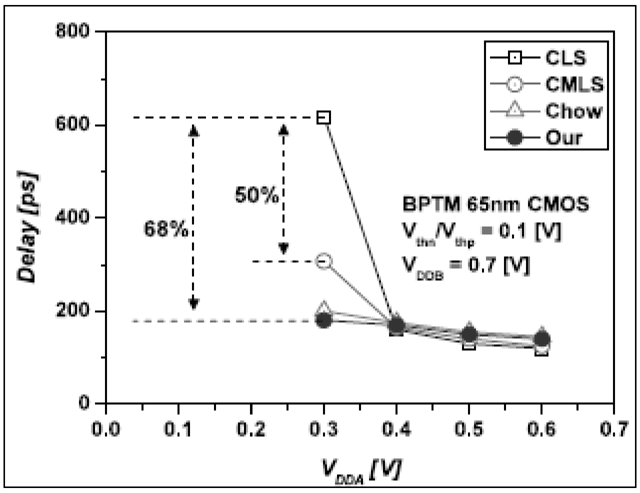


**Figure 2.10 - High speed level shifter, simulation of delay [10]**

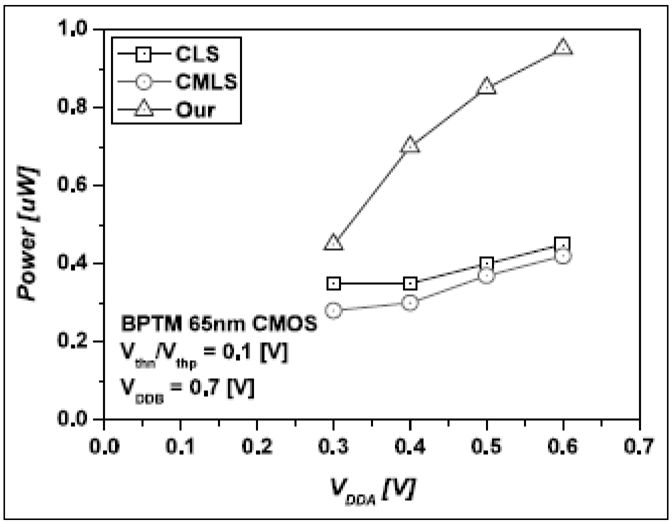


**Figure 2.11 - High speed level shifter, simulation of power consumption [10]**

In these simulations, there are several level shifters being simulated. The level shifter described above is named “our” in the figures. The “CMLS” and “Chow” level shifters are other improved (vs. CLS) level shifters.

As one can see from the simulations, the delay is much improved at ultra low voltages, but at higher voltages, the CLS is just as good. The high speed level shifter consumes more dynamic power, and the cost of high speed seems to be power. Since this assignment is to reduce the power, the level shifter might not be appropriate. If the low/core voltage of the microcontroller is scaled down to the levels shown in the simulation ($V_{CORE}$ < 0.5V), the level shifter may be chosen.

### 2.2.5 - Level Down shifter

The three proposed level shifters above are up shifters only. The down shifting is much simpler, and can be realized using only inverters. The conventional down level shifter is shown in Figure 2.12. The circuit is realized consisting of two inverters, where the first inverter use MOSFETs designed for the $V_{IO}$ domain (high oxide MOSFETS), and the last uses MOSFETs for the $V_{CORE}$ domain.

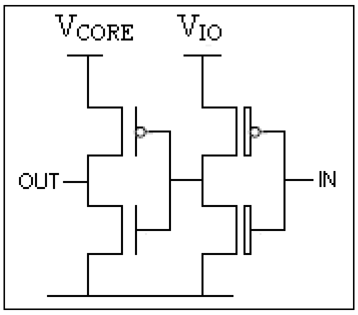


**Figure 2.12 - Conventional level-down shifter. This is an easy and often used level shifter to shift from high to low voltage. The inverter to the right uses thich oxide layer MOS transistors to withstand the high voltage.**

# 3 - System Design Consideration

When designing a multiple power domain system, there are some considerations. In this assignment, I will look into some of the basic considerations, such as different voltages and different clock sources. I am going to show how to solve these problems. One must also take into consideration that the digital world is just a moody occurrence of the analog world, and that the gates in a digital design can have analog values for some time, this phenomena is called metastability.

## *3.1 - Voltage Difference*

When designing a system using several voltage domains using different voltages, one must take into consideration that communicating between these domains cannot be done straight forward. The circuit then requires at least level shifters to compensate for the voltage differences. If the communication between the domains is bidirectional, the level shifters must be designed according to this. Level shifters can easily convert the voltage difference, but they always add a time delay. The delay can be modeled by a RC network, where the capacitance is the load, usually the gate capacitance at a node, and R is usually the resistance in the MOS device. If the domains use different clocks, synchronization logic is also required.

## *3.2 - Different Clock*

Two domains may use different clock sources, or the clocks may be out of phase. This requires synchronization between the domains. There are a number of synchronization methods for this, choosing the right one is dependent on the communication speed and the relationship between the clock sources. The traditional asynchronous communications is the usage of a full handshake scheme.

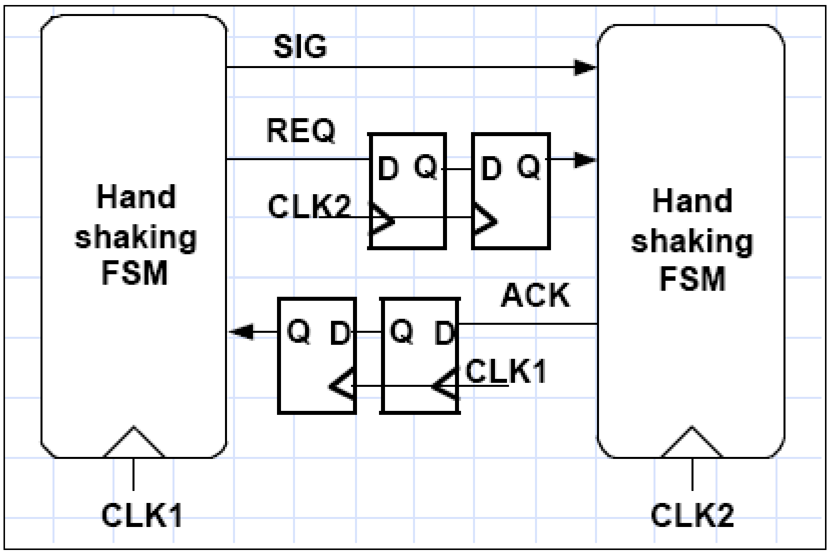


**Figure 3.1 - Traditional handshake communication [7]. This show one way communication, and uses D flip flops to reduce the chance of metastability.**

Figure 3.1 shows the traditional handshake communication with synchronization flip flops. The figure shows a one-way communication, but expanding to half duplex (two-way, but one at a time) communication is quite easy. The traditional 4-phase handshake scheme works like this [7]:

1. Sender outputs data, then asserts REQ
2. Receiver latches data, then asserts ACK
3. Sender deasserts REQ, and will not reassert it until ACK is deasserted
4. Receiver sees REQ deasserted, and deasserts ACK when ready to continue

Appendix C contains a timing diagram of the handshake scheme.

To get a half duplex communication system, one need REQ and ACK signal from each side, and one need to be able to transmit the data both ways. With the traditional handshake protocol, it is not possible to use the same data ports, as the signal is outputted before the REQ signal, this might cause crash between data. The solution is either a renewal of the scheme, or two data ports on each side. Two data ports on each side give the highest bandwidth, but require most area. By using two data ports, it is also possible to have full duplex.

To reduce the switching of signals, the handshake protocol can be adjusted to an edge based handshake scheme, often known as 2-phase. The edge based scheme works like this [7]:
1. Sender outputs data, then changes state (inverts previous value) of REQ. Sender will not change state of REQ until after ACK has changed state.
2. Receiver latches data. Once receiver is ready to accept more data, it changes the state of ACK.

If the amount of data transferred across the channel is high, the implementation of a FIFO (First In First Out) might be necessary. A FIFO is a buffer between two modules communicating through a channel. The FIFO stores the data and sends it when the receiver is ready for receiving. The FIFO is practical when the amount of data is large, but not continually. The size of the buffer/FIFO can be adjusted to the needed amount. The FIFO is more complex and requires more area and consumes more power than just a traditional handshake scheme, but the FIFO increases the bandwidth of the communication.

## *3.3 - Metastability*

If one look at the transistors used in CMOS, they are analog devices. In digital design, one wants the gates to have only two states, '1' and '0', but because the transistors are analog, the gates can have any value between $V_{DD}$ and GND, this is called metastability.

### 3.3.1 - What is metastability?

Data is clocked into register and through latches and flip flops at clock flanks. The input data needs to be stabile for a certain time before ($t_{setup}$) and a certain time after ($t_{hold}$) the clock flank. If this requirement is not met, the register, latch or flip flop may enter an unknown state, this is called metastability.

When designing digital logic using clock signals, the synthesis tool usually calculate the clock speed to meet this demand. When asynchronous communication is used, the data arrival is unknown. It is not possible to avoid metastability when using asynchronous communication, but there are a number of methods to reduce the possibility of metastability. The most common method is to have a number of synchronizing flip flops between the asynchronous signal and the synchronous module; this is illustrated in Figure 3.2. The probability of metastability decreases by each flip flop we implement, but the delay also increases. Each flip flop increases the delay by one period of the synchronous clock (clk).

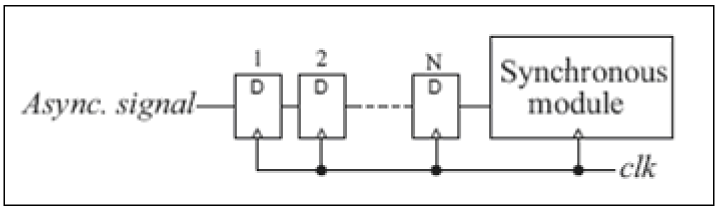


**Figure 3.2 - Synchronizer of asynchronous signals [5]. By adding several D flip flops in series, the asynchronous signal is synchronized, and the more flip flop used, the more the chance of metastability is reduced.**

The operation of a two state version of the synchronizer in Figure 3.2 is shown in Figure 3.3. As one can see, if the asynchronous signal causes the first flip flop to enter a metastable state, this is resolved after the second flip flop. The time the first flip flop exits the metastable state is not certain, so the metastability can still be unresolved after the second flip flop. The probability that the flip flop will exit the metastable state increases exponentially with the time, so the more flip flops one cascade couples, the less is the probability that the output is metastable. Calculations for this are done in chapter 3.3.2 - Mean Time Between Failure. The type of flip flop used is essential, dynamic flip flops cannot be used, as they have no regeneration [7]. If available, there should be used special synchronization flip flops.

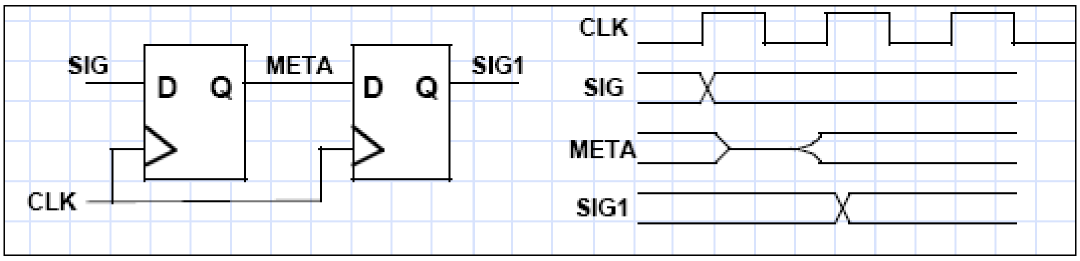


**Figure 3.3 - Synchronizer operation [7]. This shows how the signal is synchronized, and the delay is the price to pay for reducing the chance of metastability.**

### 3.3.2 - Mean Time Between Failure

MTBF is "Mean Time Between Failure", and is an estimate of the time between failure in a flip flop (i.e., when the flip flop does not resolve the metastable state in time). The MTBF for a flip flop, register etc is given by Formula 3.1.

$$\boxed{MTBF = \frac{e^{\frac{t_r}{\tau}}}{T_W \bullet f_c \bullet f_i}}$$

**Formula 3.1 - MTBF [5]. This formula is used to calculate the mean time between failure, where failure is reading wrong logic value of the signal, caused by metastability.**

$t_r$ is the allowed metastability resolution time, $f_c$ is the clock frequency, $f_i$ is the frequency of the asynchronous input data signal. $T_W$ is the likelihood that the flip flop will enter a metastable state and ! is the time the device is expected to stay in the metastable state. $T_W$ and ! are depended on the characteristic of the flip flop [5].
One want MTBF to be as large as possible, as it is an estimation on the time between two failures.

Some typical parameters for a 0.25um ASIC library [7] are:
$T_W$ = 9.6as
! = 0.31ns
$t_r$ = 2.3ns

If we use $f_c$ = 100 MHz and $f_i$ = 1 MHz, and we put these parameters into Formula 3.1, we get MTBF for a single flip flop:

$$\boxed{MTBF = \frac{e^{\frac{2.3ns}{0.31ns}}}{9.6as \bullet 100MHz \bullet 1MHz} \approx \frac{1668}{9.6 \times 10^{-4}} = 1737500s \approx 20.1 \quad days}$$

**Formula 3.2 - MTBF without synchronization [7]**

A digital designer wants to increase this time, if possible. As one see from Formula 3.1, the numerator is dependent on the flip flop, so the digital designer cannot do anything about that. The denominator contains the frequency of the asynchronous signal and the frequency of the clock signal to the digital circuit. To increase the MTBF we can simply decrease $f_c$ and/or $f_i$, but this reduces the performance of the circuit.

The traditional and easy approach to increase the MTBF is to cascade couple several flip flops. The new MTBF for the system is MTBF(N), where N is the number of flip flops. The formula is still the same, but the $f_i$ is not the asynchronous frequency, but 1/MTBF(N-1) [8]. With some algebra we can derive the MTBF(N) as:

$$\boxed{MTBF(N) = \frac{1}{f_i} \bullet \left(\frac{e^{\frac{t_r}{\tau}}}{T_W \bullet f_c}\right)^N}$$

**Formula 3.3 - MTBW for N size synchronizing circuit**

If we cascade couple two D flip flops as shown in Figure 3.3, the MTBF(2) with the same parameters as in Formula 3.2 we get a $MTBF(2) \approx 9{,}57 \times 10^{10} \quad years$.

# 4 - System Solutions

There are many different design options when designing a multiple power domain system. Different systems have different advantages and disadvantages. The key is to find the solution best suited for the specific design, and cost has to be taken into consideration.
The AVR is a RISC microcontroller, aiming to use as little power as possible, and still maintain functionality. This can be achieved by setting the microcontroller in sleep or power off modes when there is no activity. With new process technology, we cannot only disconnect the clock; we also need to disconnect the voltage to the modules to prevent leakage currents to get a real power off mode.

It is desirable to use as low voltage as possible to power the digital circuit, this reduces both the dynamic and static power usage. The external voltage used is dependent on the devices the microcontroller communicates with, but the internal voltage can be controlled and set to a desired value. It is desirable to use the microcontroller in systems using up to 5.5V without using an external voltage regulator and level shifters. This then requires an internal voltage regulator if the internal voltage is set to be lower. This regulator must be efficient, since the idea of using low voltage is to use less power. The voltage level used is dependent on the technology available. To get as low power consumption as possible, the voltage needs to be as low as possible. To have the performance (low delay) that is required, one may be scaling the voltage higher than the lowest possible (see 2.1.4 - Voltage scaling).

The digital logic uses less power when the voltage is lowered. This is actually not the case for the analog modules; which use more power when the voltage is lowered. The analog modules are then most efficient to implement in the $V_{IO}$ domain.

In this assignment, I limit the number of voltage domains to two. If more domains are needed, the principles are still the same. In addition, there might not be need for more than two domains, since the AVR is a relatively small design. For each power/voltage domain there must be voltage regulators and synchronization logic, and may then lead to using more power, rather than less.

## 4.1 - Principle 1: Two Separate Power Domains

One can implement the system with two separate power domains, where the first domain uses the external voltage, for example 5V. This power domain is always on and can communicate with the peripherals. The second power domain is a low voltage digital domain; this domain can be either on or off. During a *power off mode* (5.1.2 - Power Off Mode), the voltage regulator is turned off, and the $V_{CORE}$ domain is in real power off mode. The $V_{IO}$ domain still uses power, so the controller will not be in power off mode, but most of the power is used in the $V_{CORE}$ domain, since most of the logic is there.

This is a quite easy approach to the task, and it is possible to implement. Figure 4.1 shows the principle of such a design, where the $V_{IO}$ domain is directly connected to the external voltage, and we implement a voltage regulator to generate the $V_{CORE}$. We have one data bus on each voltage domain, and these are connected together through level shifters and synchronization logic.

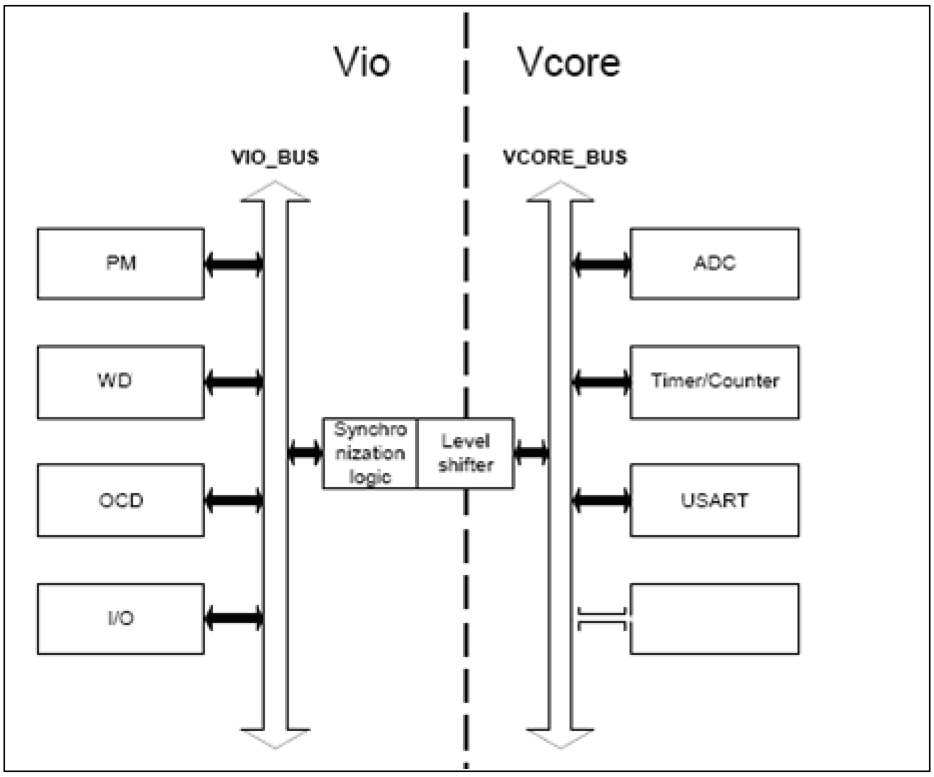


**Figure 4.1 - Two separate power domains. When using aggressive process technologies, the transistors are getting smaller and smaller, but the voltage needs to be adjusted because the transistors cannot withstand high voltages such as 5V or 3.3V, which is commonly used today and, and is still used by the automotive industry. Therefore most of the logic is implemented in a domain using lower voltage.**

## *4.2 - Principle 2: One Digital Low Voltage Power Domain*

Another solution is to implement one low voltage digital domain. The domain will connect to the peripherals using internal level shifters. The analog modules will use the external voltage. Then all the modules can be turned on or off using a power management unit (PMU). This requires a voltage regulator from the external voltage and down to the core voltage. This voltage regulator needs to be very efficient, because if the regulator is leaking, the gain of reducing the power is less. This is because the PMU is powered from the voltage regulator, so the voltage regulator needs to be active at all times. If one needs a *power off mode,* the leakage in the regulator will dominate the power usage, and that is not a good power off mode. This is the main problem with this design vs. principle 1, because the voltage regulator can be turned off in solution 1. Principle 2 has higher potential of power saving than principle 1 during active mode because all the digital logic is in the low voltage area. In sleep modes, the voltage regulator leakage might be so large that the total power consumption for principle 2 is greater than for principle 1. Figure 4.2 shows the principle 2.

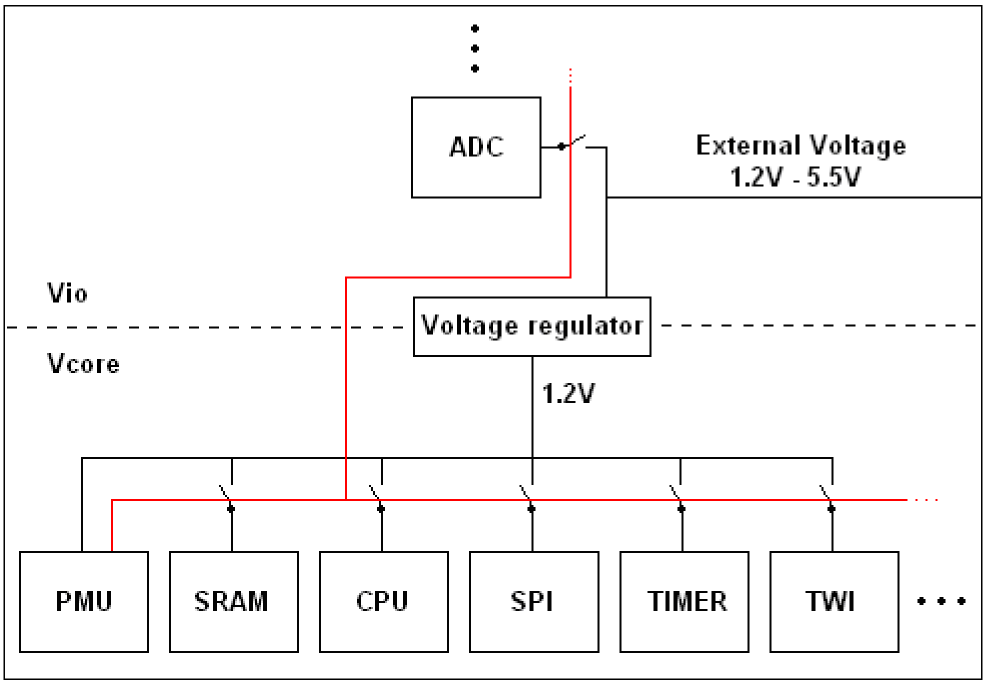


**Figure 4.2 - One digital low power solution. All logic is implemented in a domain using low voltage.**

### *4.3 - My solution*

There are advantages and disadvantages of the two principles described above. If one take the advantages of both, and create a middle path, one get a solution which is practical and power aware. To be able to save as much power as possible, both during active mode and power off mode, as much as possible of the logic is implemented on the low voltage domain.

#### 4.3.1 - Overview

Figure 4.3 shows my proposed solution. The PMU is placed on the I/O domain because the voltage regulator must be able to be turned off during sleep modes, and we need the PMU to turn the voltage regulator on again. A low speed clock clocks the I/O domain; this is the watchdog (WD) clock that has been used to drive the watchdog timer in the AVR. The WD clock on the AVR today is 128kHz.

There are two voltage domains; these are divided into several voltage islands. The voltage islands make it possible to turn off one or more islands when modules are not used, and minimize the power consumption caused by the leakage currents. Communicating between islands on the same voltage domain is unproblematic, as they all use the same voltage level and clock source. The communication between the low voltage ($V_{CORE}$) and high voltage ($V_{IO}$) voltage domain is done by a synchronization module that consist of level shifters and an asynchronous communication module (See 3 - System Design Consideration). The IO pins are reached through level shifters, and as today, through synchronization flip flops.

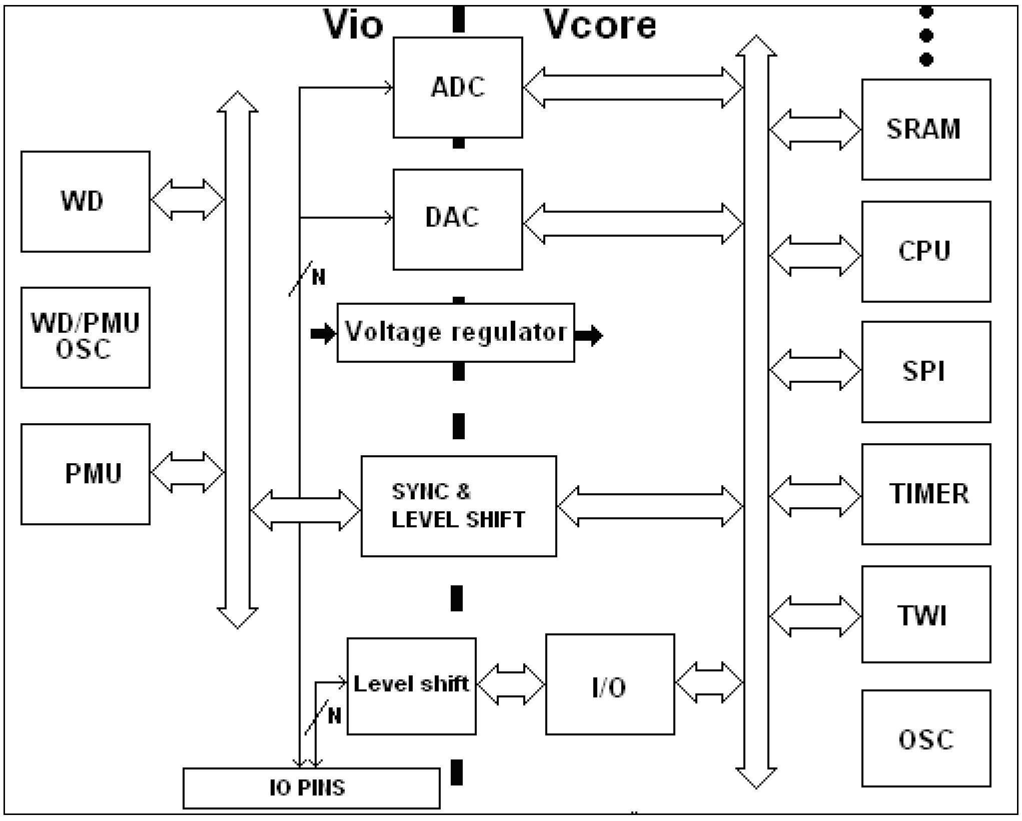


**Figure 4.3 - My solution. Most of the logic is on the $V_{CORE}$ side. The $V_{IO}$ domain contains the PMU and a low speed oscillator. The analog modules use both the $V_{IO}$ and the $V_{CORE}$.**

### 4.3.2 - Analog Modules

This assignment is about optimizing the digital part of the microcontroller. So the analog modules here are only mentioned, because they are important. The design and characterization of these modules are not a part of this assignment. The analog modules must be able to be turned on and off by the PMU.

The analog modules such as the ADC (Analog to digital converter) and DAC (Digital to Analog converter) use both voltage domains. The analog part of the modules uses the I/O voltage, and the digital part uses the $V_{CORE}$. By doing so, the digital part of the modules can use the system clock, and by using level shifters, the physical voltage gap is resolved.

A general feedback ADC is shown in Figure 4.4. The “Up/Down counter” is the digital part, and uses the $V_{CORE}$ voltage and system clock. The output from the comparator and the counter is digital, so applying level shifters at this point will make the circuit suitable for implementing at both voltage domains.

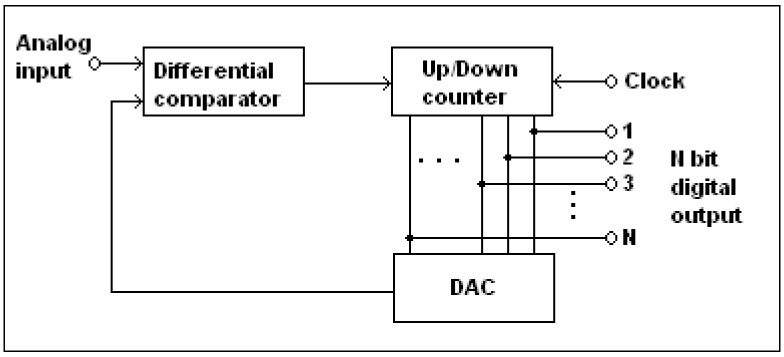


**Figure 4.4 - Feedback ADC [4]. A feedback ADC is a commonly used ADC. The comparator and DAC is analog and uses $V_{IO}$, the counter is digital and uses $V_{CORE}$. Level shifters on the input of the DAC solves the voltage difference.**

To implement the DAC on the I/O voltage domain one need level shifters on the digital inputs. The DAC has N digital inputs, where N is the resolution, and one analog output.

The oscillators generate the clock needed for the digital circuits. The AVR contains one crystal oscillator, and several RC oscillators. The RC oscillators make it possible to run the AVR without the usage of external crystals. Only one oscillator is used, and is selected by setting fuses when programming the AVR. The high frequency oscillators are implemented in the low voltage domain. By doing this, the oscillator can be turned off during power off mode, and starting the microcontroller after a power off mode requires starting the oscillator first. This takes some time, see 5.1.5 - Start Up Time.

To scale the voltage down from the IO voltage to the core voltage, one needs a voltage regulator. The voltage regulator must have a steady output voltage, and have low leakage. This is quite difficult to achieve, but is essential to ensure that the microcontroller uses as little power as possible. As the low voltage area is shut off during “power off mode”, the voltage regulator can also be turned off. The voltage regulator must be turned on and have a stable voltage before the rest of the low voltage area can be turned on. So choosing the right voltage regulator is essential to the start up time (see 5.1.5 - Start Up Time).

### 4.3.3 - Digital Circuits

Most of the digital circuits of the microcontroller are placed on the low voltage area. This results in reduced power consumption in active mode. The digital circuits operate at the same voltage and use the same clock source, so there is no need for synchronization when communicating between the low voltage digital circuits.

The WD and the PMU are digital circuits placed on the high voltage area. The PMU must always be on, and can therefore not be placed on the low voltage area. The WD is just a timer running on a low speed clock source, to save the need of two low speed oscillators, the PMU and WD share clock. Because of this, we place the WD on the high voltage area.

Communication between the low voltage area and the high voltage area is slow, since the high voltage area uses a low frequency clock. This is unproblematic since there are a small amount of dataflow between the domains, compared to the dataflow on the low voltage bus. The data going from low voltage area to high voltage area are resetting of the WD timer, and instruction to the PMU to initiate a sleep or power off mode.

### 4.3.4 - Voltage Islands

The usage of new transistor technology with lower voltage results in less area, because the transistors are smaller. Smaller transistors are faster, but cannot withstand high voltages. Smaller transistors results in higher leakage currents. Therefore we need to be able to turn the power off to reduce these currents when the circuit is not active. Therefore the low voltage area is divided into several voltage islands. They use the same voltage level, but are able to be turned off individually. The amount of voltage islands and what to place together has no exact answer, as there are many possible solutions. A way of dividing may be to place communication modules, timers or other similar modules together.

Dividing the high voltage area into voltage islands makes it possible to turn off some of these modules too. This can for example be the WD if not used, or the analog modules. In addition, to make a completely power off mode, one might turn off the WD/PMU oscillator and turn off parts of the PMU, and use an asynchronous wake up (see 5.1.8 - Asynchronous Wake Up)

Voltage islands can be turned off by power gating (see 2.1.3 - Power gating).

### 4.3.5 - Clock Distribution

The low voltage side of the chip uses the main clock source. This clock is generated using either some internal RC oscillators, external oscillators or an internal crystal oscillator with an external crystal. Dividing this clock source into several independent clock sources gives the opportunity to shut off one or more of these clocks to either save power, or reduce the noise they generate. Since these clocks are generated from the same source, the modules can communicate synchronous. Since the clock frequency is high, bad wiring may cause clock skew. Clock skew is a phenomenon where the clock flanks arrive at gates as different times. This is caused when the length of two clock wires are not the same. The problem with clock skew is increasing with increased clock frequency. Using new transistor technology with smaller transistors decreases the area, and may decrease the wire lengths, but by adding more transistors per area unity, the wiring is more complex, and may wiring may be problematic. The solution so clock skew is to make sure that every clock net has the same length. This can be done by using a H-pattern to the wiring, this is shown in Figure 4.5. The clock enters the net at the blue dot, and reaches its destination at the red dots. By examining the figure, one can see that the path from the blue dot to each red dot is the same.

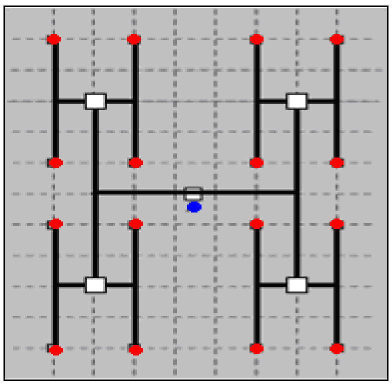


**Figure 4.5 - H-pattern of clock net distribution. By using the H-pattern, the lengths of all clock wires are the same.**

# 5 - Power Management unit (PMU)

To control the power management in the AVR I implement a unit called the “Power Management Unit” (PMU). The PMU will control the sleep- and power off modes of the AVR. The PMU must make sure that the AVR is not going to a sleep- or power off mode when writing to RAM or executing other operations, and the shutting down of modules must be done in the right order.

Waking up/activating the AVR is also an important task for the PMU. All modules cannot be turned on at the same time, as this will cause large rush currents and some modules must be turned on before others.

## *5.1 - Sleep- and Power Off –Modes*

Small microcontrollers like the 8-bit AVR are often used in applications where the microcontroller spends most of the time in some sort of idle state. A perfect example of this is a remote control for a TV or DVD player. In this example, the microcontroller must read the buttons, and send an IR or RF signal when a button is pushed. There is no need to keep the microcontroller on when the buttons are not being pushed. When the remote control is inactive, the microcontroller can be set in a sleep- or power off mode.

During the active mode, the microcontroller consume quite a lot power, but when going to a sleep- or power off mode, the power consumption is down to a minimum. If the application spends most of its time in the sleep- or power off mode, the average power consumption is most depending on the power consumption in the sleep modes. In the remote control example, the time spent in sleep might be over close to 100%, this is illustrated in Figure 5.1

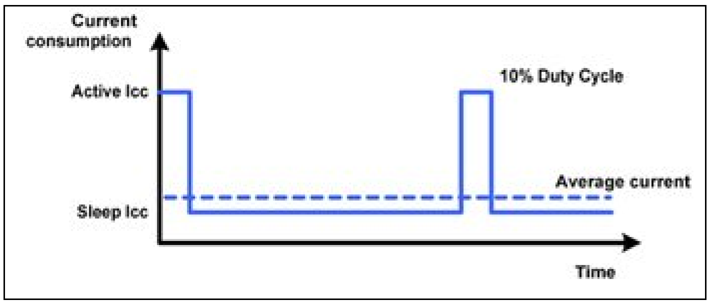


**Figure 5.1 - Power Budget [11]. As the microcontroller is in sleep mode most of the time, the average power consumption is mostly dependent on the power consumption in sleep mode.**

### 5.1.1 - Sleep Modes

Having an infinite number of sleep modes would have been the ideal situation. The right sleep mode would always been initiated depending on the application. Unfortunately, this is not possible. Today there are five sleep modes in the AVR (six for pico power devices). These are IDLE, ADC noise reduction, Power-Down, Power save and Standby (extended standby for pico power) [13]. The choice of sleep mode is done according to the application. The deeper sleep mode, the longer the wake up time is.

During a sleep mode, several modules in the microcontroller are turned off. Turning off clock sources is a functional sleep mode. By turning off the clock, there can be no dynamic power usage. To get a deeper sleep mode, the oscillator can be turned off.

This has usually been enough to initiate sleep modes when the leakage currents have been small. But with new transistor technology, the leakage currents can no longer be ignored, and modules might need to be power gated, or turned off during sleep modes.

To be able to start the microcontroller again, we need some sort of wake up signal. It is standard to use an external wake up signal applied to an interrupt pin on the microcontroller. Another frequently used wake up signal is an internal timer interrupt; this is a good wake up signal when the amount of time spent in sleep mode is known before entering the sleep mode.

### 5.1.2 - Power Off Mode

During the sleep modes, there is still some power consumption. It is preferable to turn the microcontroller in a power off mode where the power consumption is as close to zero as possible. To make such a mode, most of the microcontroller must be turned off completely. Since the microcontroller is divided into two voltage domains, and then divided into several voltage islands, it is possible to turn most of the microcontroller completely off by using the power gating (see 2.1.3 - Power gating). The only part needed active, is the wake up detection part. Normally a digital circuit needs a clock source to drive the logic that detects the wake up signal. By choosing a low speed clock, the active power usage is minimized. But the oscillator used to create the clock uses quite a lot of power. It is then desirable to have a module that can wake up the microcontroller without the need of a clock source (see 5.1.8 - Asynchronous Wake Up).

The microcontroller can start with a cold or a hot start after a sleep or power off mode. The difference is that a cold start restarts the program from the beginning, while a hot start starts where the sleep was initiated. Some applications needs a hot start, and therefore the value of the program counter (PC) and the status register (SREG) must be stored. When the voltage to the $V_{CORE}$ domain is turned off, these values are deleted. Therefore, these values must be stored in some registers inside the PMU that never looses power. This increases the time the microcontroller uses to initiate a power down mode, but is a useful when a hot start is required. This is showed in Pseudo 5-1.

```
Power down with SREG and PC storage

1 : Power down command received
2 : Fetch and Store SREG in PMURAM
3 : Fetch and Store PC in PMURAM
4 : Start power down
```

**Pseudo 5-1 - Microcontroller Power-down with storing SRAM and PC. By storing these values, the microcontroller may start up where it was shut down, called hot start.**

### 5.1.3 - Initiating a sleep/power off mode

When the microcontroller enters sleep or power off mode, the CPU must stop the program counter (PC). Then the PMU takes control of the chip. If a hot start is required, the PMU must fetch the content of the PC and SREG, and store in registers on the high voltage domain. The PMU must then turn off the voltage islands and the voltage regulator, and initiate the wake up source(s) needed to wake up the microcontroller. Then the WD oscillator can be turned off if the mode is a power off mode.

### 5.1.4 - Initiating a wake up

The PMU must take control of a wake up of the microcontroller. When a wake up signal is detected, the PMU must turn on the voltage regulator, if it is turned off, and turn on the voltage islands in the right order. The PMU must also start the high frequency oscillator and restore the SREG and PC. If a hot start, the values are stored, if a cold start, the values must be reset to an initial value.

### 5.1.5 - Start up sequence

When a microcontroller has entered a sleep- or power off mode, it takes some amount of time to wake up the microcontroller again. A general rule is that the deeper sleep mode, the longer the wake up time is. This is because the deeper the sleep mode, the more modules needs to be turned on before the microcontroller is in active mode again.

The oscillator must be turned on first and become stable before any of the digital circuits can be turned on. And if the voltage is turned off at some modules, the voltage must be stable. The start up sequence is showed in Pseudo 5-2.

```
Start up sequence

1 : If(WDosc = off)
        Start up low frequncy oscillator
        Wait until oscillator is stable
        Start low frequency clock
2 : If(Voltage regulator = off)
        Start up voltage regulator
        Wait until voltage is stable
3 : Turn on Voltage to voltage islands
4 : if(SREG and PC is stored)
        Load SREG and PC
5 : If(CPU clk = off)
        Start CPU clock
```

**Pseudo 5-2 - Start up sequence, step 3 may include many islands to be turned on in sequence to keep the current from the supply to a acceptable value.**

To ensure that the start up time is kept as low as possible, and the current dawned from the power supply is kept under a certain value, $I_{MAX}$, the all modules cannot be started at the same time. This is illustrated in Figure 5.4, where modules are started as soon as possible, one can see that the $I_{MAX}$ is exceeded. By turning on the different voltage islands in a fixed order, the start up time can me minimized if the right sequence is selected. The PMU needs to know which voltage islands are on and which are off at every time. A good way of arranging this is to have a register (VIR, Voltage Island Register) that keeps track of this. If there is for example sixteen different voltage islands, there would be need of one sixteen bit register to keep track of the status. Since this assignment is based on the eight bit AVR microcontroller, the register could be divided into two eight bit registers. One voltage islands is to be turned on at any time, so by simply shifting logic ones into the register from one side, all the voltage islands would be turned on after sixteen shifts. Since there are voltage islands that may need longer time to be ready, and voltage islands that needs to be turned on before others, the selections of which voltage island belonging to which bit in the register is important. The arrangement of the VIR is illustrated in Figure 5.2, where “Bit 0” is MSB and shifted into first, and is connected to “voltage island 0”. Arranging the voltage islands must de done according to which needs to be turned on before others, and according to the size, which again leads to time to charge the capacitances. To optimize the start up time and keep the current drained from the power supply below $I_{MAX}$, the voltage islands needs to be turned on in sequence, and the time delay between them must be optimized. If they are turned on to quick the $I_{MAX}$ is exceeded, if they are turned on to slow, the start up time is extended. If an optimal sequence and time delay is chosen, the current drained is illustrated in Figure 5.5. The logical one shifted into MSB can be shifted in by implementing the register as a shift register, and use a clock source to the shift register that is optimized so that the start up sequence results in an $I_{DD}$ illustrated in Figure 5.5.

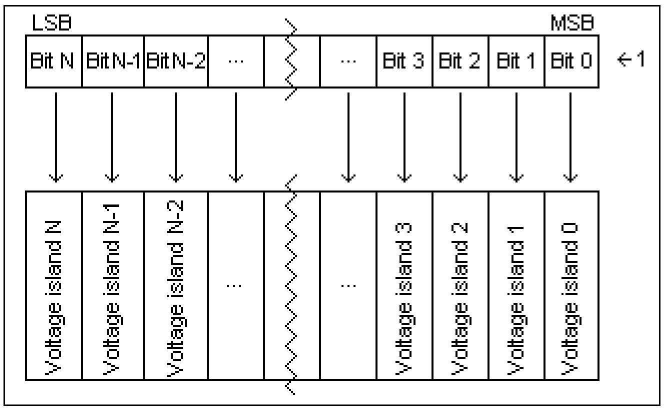


**Figure 5.2 - Voltage islands turn on sequence. By setting the bits in the VIR, the voltage is turned on at the appurtenant voltage island. When the logic 1 is shifted in from MSB to LSB, the voltages is turned on to the voltage islands, one by one. This reduces the rush currents, but increases the start up time compared to turning all voltage islands on at the same time.**

### 5.1.6 - Start Up Time

The time needed to get the microcontroller running after a sleep mode is essential. If the start up time is too slow, the programmer may not use the sleep mode. It is therefore preferable to have several sleep- and power off modes. If quick start up is required, a less deep mode is chosen. Since the voltage domains are divided into voltage islands, parts of the microcontroller can still be in sleep mode (off mode) while the CPU wakes up, and starting these islands after the program has started. This can for example be the island containing the communication modules. This makes it impossible to communicate for a short time, but makes it possible to have a quicker start up. Calculations of the start up time can be done using Formula 5.1. The $t_{OSC\_ON}$ is the time to start the oscillator and clock, $t_{LDO}$ is the time to get a stable voltage from the voltage regulator. The $t_{INIT}$ time is the time to set up the microcontroller. This is the time to get all the capacitors charged, and the time to set all the registers to an initial value. As the chip gets larger, the effective capacitance ($C_{EFF}$) will become larger, and the start up time of the microcontroller will increase because it takes longer time to charge a larger capacitor. The time to get an oscillator and voltage regulator stabile can be quite long. The $t_{LDO\_ON}$ in Formula 5.1 is when the voltage is within a required level. To decrease the rush currents, all modules cannot be started at the same time, as it will result in a current spike (Figure 5.3) that exceeds $I_{MAX}$. Otherwise, it would be possible to start the oscillator and voltage regulator at the same time, and only the longest in time would be included in the formula. At this time, the start up time cannot be estimated, because the $t_{INIT}$ time cannot be estimated without any hardware. Most of the low voltage hardware is available in code, but the PMU and the communication blocks between the domains are not. If a power down mode is used, the start up time from this must be estimated to the same as the start up time when first applying the voltage, since almost all registers must be set, and all capacitors charged.

$$\boxed{Start\_up\_time = t_{OSC\_ON} + t_{LDO\_ON} + t_{INIT}}$$

**Formula 5.1 - Start up time for the microcontroller. The $t_{INIT}$ includes startup of all voltage islands and initialization of the CPU.**

To avoid big rush currents, the voltage islands in step 3 (Pseudo 5-2) has to be turned on one by one. If we have ten voltage islands, we might need 10 steps, depending on the size of the voltage islands. When a voltage islands are turned on, there are several networks of transistors, which can be modeled as a number of capacitances. The capacitances needs to be charged, and therefore large currents may occur during start up. All this current is drained through the $V_{DD}$ lines on the PCB and through the $V_{DD}$ pins on the microcontroller. If this current is too large, there might be a voltage drop in the $V_{DD}$ line. The voltage drop is called an IR drop, because the supply line has a certain resistance R and there flows a certain current I. Figure 5.3 illustrates the current drained from the power supply when the microcontroller is turned on. If the $I_{MAX}$ line illustrated in the figure is the maximum current that can be drained from through the VDD, the current illustrated is far too high. If the different voltage islands are turned on one by one, the current spike will be reduced in amplitude, but wider. This is illustrated in Figure 5.4. As shown on the figure, the high spike has been reduced, but is wider. This causes the start up time to increase, but is necessary to keep the current low.

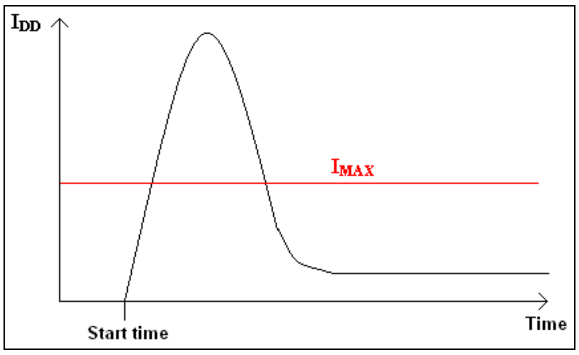


**Figure 5.3 - Current spike on startup of a microcontroller when all modules are started simultaneous. The current exceeds $I_{MAX}$.**

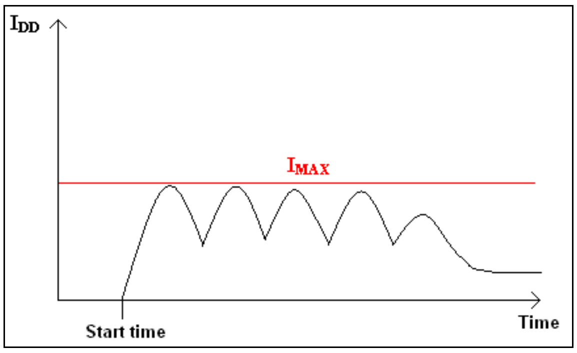


**Figure 5.4 - Several smaller current spikes when starting up modules one by one. We can see that the current is kept under $I_{MAX}$.**

Another problem with large rush currents is the possibility of electro migration in the supply lines. The electro migration is a phenomenon, which happens when there is flowing more current in a line than the capacity of the line ($I_{MAX}$). This results in a failure that is an electrical open circuit due to the loss of conductor metal [12].

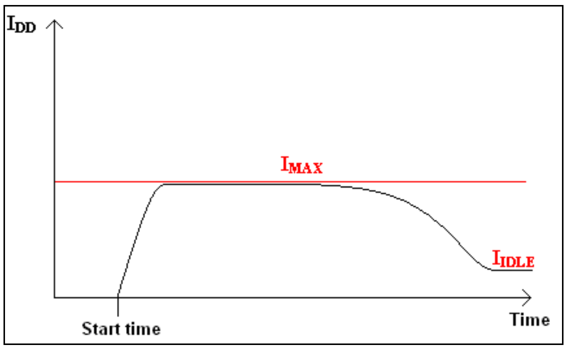


**Figure 5.5 - Optimal start up, minimum time and $I_{DD}$ kept below $I_{MAX}$.**

If the start up sequence is optimized, the current spikes are placed close together and drain close to IMAX until the start up is finished. The start up time is then minimized, and $I_{DD}$ is still kept below $I_{MAX}$. This is illustrated in Figure 5.5.

### 5.1.7 - Synchronous Wake Up

In deep sleep modes and power off mode the microcontroller turns the main clock and oscillator off, because the purpose of the sleep signals is to save power. If the WD clock is still running, this clock makes it possible to have a synchronous wake up. The usage of synchronous wake up is reliable, because it is better at detecting false wake up signals that might be caused by noise. The synchronous wake up is faster than asynchronous, because there are fewer modules that need to be turned off. Therefore, during less deep sleep modes, synchronous wake up might be preferable.

### 5.1.8 - Asynchronous Wake Up

When introducing a power off mode, the idea is to have as close to zero power usage as possible. Even if a low speed clock source is used to detect the wake up in synchronous wake up scheme, the oscillator still use some amount of power. Therefore, there is a need for a reliable asynchronous wake up scheme. Such a scheme will reduce the power usage considerably. If asynchronous wake up is used, and it is to be trigged by a pushed button, the wake up pin should have a schmitt trigger. The schmitt trigger is a comparator with a positive feedback. The positive feedback makes a hysteresis output of the schmitt trigger. The schmitt trigger will remove the bounce noise created by the button when it is being pushed. This will reduce the chance of a false wake up signal.

A proposed circuit for asynchronous wake up is showed in Figure 5.6. The circuits' mode of operation is quite easy. When the microcontroller is entering power off mode, the state of the wake up pad is stored and used as input to the "XOR" gate (X). The capacitor C must be discharged through M1, and the SR latch should be reset. When the state of the wake up pad is changed, the output of the "XOR" gate will be high, and the capacitor C will start charging. When the capacitor has been charged to a level where the "Buf" reads the value to be high, the SR latch sets the output high, and a wake up signal has been generated. The buffer "Buf" helps to ensure that the SR latch does not toggle if noise occurs at the wake up pad. This wake up signal can be used to turn on the voltage to the rest of the PMU, and to start the low frequency oscillator. When the oscillator is stable, the PMU can wake up the rest of the circuit. The wake up circuit is level trigged, and the value of the C and R decides the time the wake up signal needs to be active (i.e. different from what it was before the power off mode).

The only circuits needed to be on during a power off mode are the reset circuit and the wake up circuit. This should reduce the power consumption close to zero during the power off mode, as there are no oscillators and clock sources on, just some simple logic blocks.

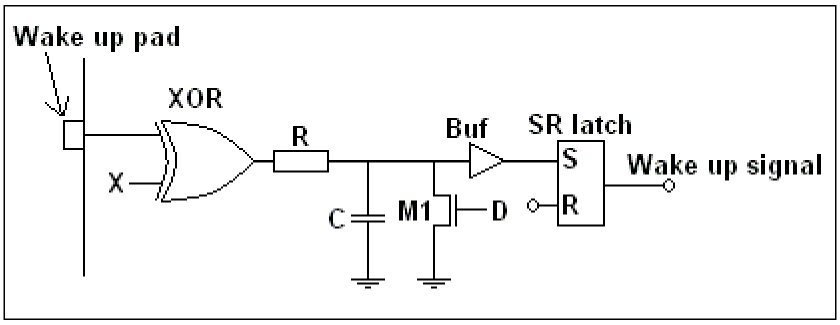


**Figure 5.6 – Easy version of proposed asynchronous wake up circuit.**

If the logic signal at the wake up pad is changed for a short time, the capacitor C gets charged by a bit. To discharge this capacitor if this occurs, the signal from the wake up pad can be routed to the gate of M1 (signal D) through logic. When the logic signal is the same as before the power off were initialized, the transistor (M1) is open, and ensures that the capacitor is discharged. If the pads logic level changes for a shorter time than the required wake up time, the capacitor will be charged, but discharged when the signal changes again. This is ensuring the microcontroller to stay in power off mode until it is supposed to wake up.

The SR latch in this figure is an asynchronous latch. An asynchronous latch can be made by two cross coupled NOR gates (Figure 5.7 a)) or two cross coupled NAND gates (Figure 5.7 b)). The difference between the two latches is the idle state that the inputs may have. The SR latch made of the NOR gates cannot have both inputs at logical ‘1’; this is called a *forbidden state*. At the NAND gate the *forbidden state* is when both inputs are at logical ‘0’.This state is forbidden because both the outputs Q and not_Q are set to the same logical value, this is ‘0’ for the NOR latch, and ‘1’ for the NAND latch. The SR latch should contain the previously state of the outputs when the latch is in idle state, this makes the latch useful when only a pulse at either the set or reset input determents the output.

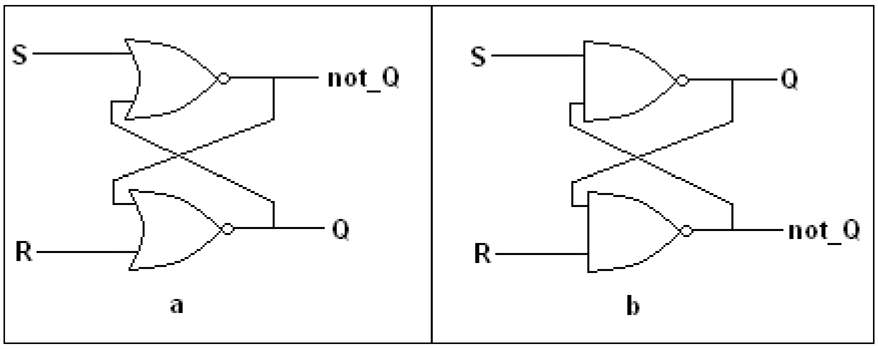
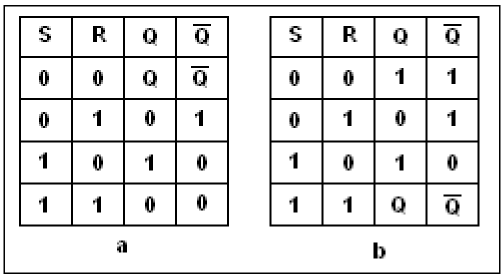


**Figure 5.7 - SR latch. a) shows a SR latch made of two NOR gates, b) shows latch made of two NAND gates.**

The legal idle state for the NOR based SR latch is both inputs at logical ‘0’, and the legal idle state for the NAND based SR latch is both inputs at logical ‘1’. At this idle state the outputs Q and not_Q should retain the previously value, this is shown in Table 5-1, a) shows the truth table of a NOR based SR latch, and b) shows the truth table of a NAND based SR latch. If the inputs enters the forbidden state, the latch must get a set or reset signal to exit this unwanted forbidden state.



| S | R | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | Q | $\overline{Q}$ |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

a

| S | R | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Q | $\overline{Q}$ |

b

**Table 5-1 - Truth table of the SR LATCH. a) is the truth table of a NOR based SR latch. b) is the truth table of a NAND based SR latch.**

Figure 5.6 needs to be expanded to be able to get an initial value, and to be able to discharge the capacitor if needed. A more complete circuit is shown in Figure 5.8.

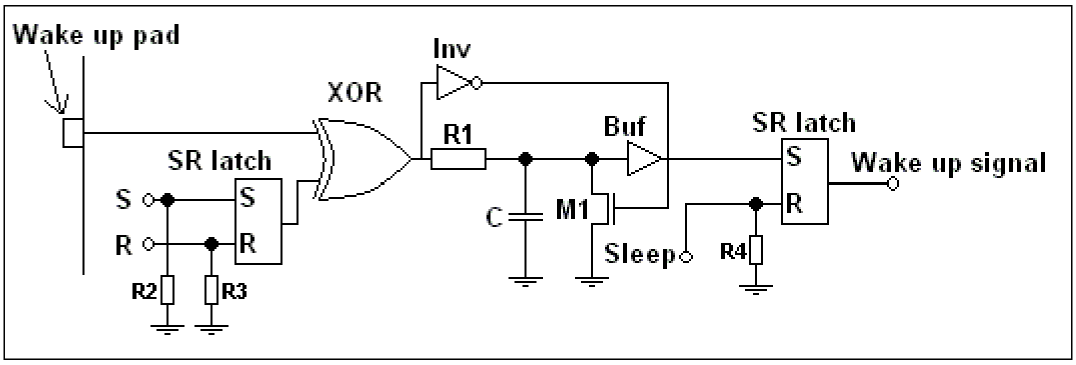


**Figure 5.8 - Complex version of proposed asynchronous wake up circuit.**

This circuit uses a SR latch made of NOR gates, therefore the inputs idle state are logical ‘0’. The “S”, “R” and “Sleep” nodes in the circuits are controlled by the microcontroller, and if a *power off* mode is initiated, these nodes will be floating. Therefore the nodes are “pulled down” using resistors. This ensures that the levels of the nodes are kept to logical ‘0’, and not floating. The values of the resistors are dependent on the transistors used. If the resistors are too small, the transistors cannot override, and cannot change the logical value of the node. If the resistance is too high, the parasitic resistance and capacitance in the circuit can override the resistors, and the value becomes floating again. The size of resistors in integrated circuits increase with the size, and since more area is costly, it is preferable to have small resistance.

### 5.1.9 - Wake Up Sources

There are a number of different sources to wake a sleeping microcontroller. One of the most common is to use an interrupt signal to the microcontroller. The signal source may be an external digital circuit or a push button.

The use of TWI address match is used as a wake up source in every sleep modes on the AVR today. The TWI uses two wires, where one is the clock and the other is the data. This makes it possible to wake up the AVR without the use of internal clock. The wake up circuit gets clock from the TWI. This is allowing the TWI circuit to be on even during sleep modes, as long as the voltage on.

In a lighter sleep mode, the usage of internal timer as a wake up source is commonly used and in use at the AVR today. This requires the main clock oscillator to run. This wake up source is useful when the designer knows the time the microcontroller should be in sleep mode, or if the microcontroller should wake up with certain interval to check a condition. To use this wake up source, the timer needs power and an oscillator. To implement an efficient sleep mode, the voltage regulator powering the low voltage domain must be off. Then implementing a low frequency timer in the high voltage domain gives the opportunity to have a timed wake even if the low voltage domain is off. This timer could use the WD/PMU clock.

If the ADC needs very accurate measuring, a sleep mode to turn off most of the digital circuits is in use at the AVR today. Then the ADC is used as a wake up source. This sleep mode is not used to save power, but to reduce the noise that the digital circuits create.

## *5.2 - Controlling Power Usage*

A PMU is implemented to control the power usage of the microcontroller. To be able to do this, the PMU must have control of the voltage regulator and the clock generator. Then the PMU can shut off the microcontroller with total control. If the PMU stops the clock to the CPU, there will not be executed any instructions. This gives the PMU time to store information, if needed and initiate start up logic before the voltage is shut off. The voltage regulator and clock generator can be turned off using simple gating transistors. The PMU must also be able to control all the power supply lines to the voltage islands, and make sure that the islands are disconnected from the rest of the system to ensure that there cannot be any leakage currents in and out of the module.

The CPU communicates with the PMU through the data bus to initiate sleep modes, but to turn on/off modules, the PMU must have extra wiring to control the sleep transistors, voltage regulator and voltage switches to the voltage islands.

### 5.2.1 - Controlling Active Mode Power

Power usage during sleep modes is the most effective way to reduce the overall power consumption in most designs using an 8-bit microcontroller. This is because the microcontroller often spends most of its time in sleep mode. Sometimes the microcontroller cannot go into sleep mode, and then it is desirable to reduce the active mode power usage as much as possible. Reducing the voltage level is the most effective way (see Formula 2.2), but reducing the effective capacitance ($C_{EFF}$) is also effective. This can be done easily by turning off some unused modules. If all modules on a voltage island are unused, the voltage can be turned off, and the leakage currents are removed too. The voltage islands can be turned off by setting logical zeroes to the VIR (Voltage Islands Register).This then requires good planning when selecting modules to be placed on the voltage islands. There is no optimum solution to this, as different applications uses different modules. It may be a good solution to have different modules placed together at different versions of the AVR controller. Then it is up the designer to choose the right controller based on the application.

## 5.3 - Summary of Key Features of the PMU

The main feature of the PMU is the asynchronous wake up circuit, which allows the microcontroller to go in a power off mode with very little power usage.

Implementing a timer in the PMU that uses the WD/PMU low frequency clock allows the microcontroller to use a timer as a wake up source from a sleep mode, and still turn the power off from the low voltage area.

Register within the PMU to store the PC and SREG. This allows the microcontroller to have a hot start, even if the voltage is turned off at the low voltage area. The PMU must use the data bus to fetch the values of SREG and PC, and store them before turning the power off at the low voltage area.

The PMU have the possibility to turn off voltage islands during active mode, this reduces the total leakage current of the chip, and removes unnecessary switching in unused modules.

# 6 - Simulations

The proposed asynchronous wake up circuit must be simulated to test its functionality, and check the power saving potential. The most efficient way to save power in a microcontroller design is to reduce the sleep mode power consumption. The power analyzes in this assignment is only for the power off mode. The simulation is done using “Eldo” from “Mentor Graphics”. The transistor models are Atmel technology, and they are confidential, and therefore not published with this assignment.

## *6.1 - Functional Simulation*

The proposed circuit (Figure 5.8) gives a logical high out when a wake up signal is present at the input for a certain time. The time can be adjusted by setting the R1 and C values. If the wake up signal is not present for the required time to wake up the circuit, the discharging transistor M1 will discharge the capacitor C. The leakage in the CMOS technology would discharge this capacitor, but the time would be significantly higher.

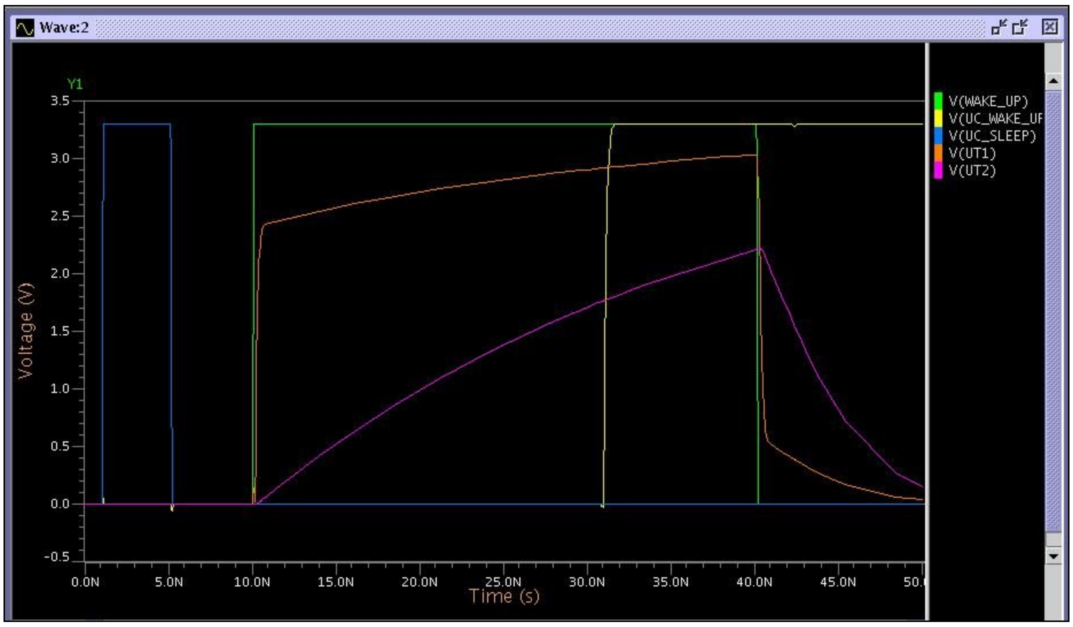


**Figure 6.1 - Functional simulation of the proposed circuit.**

The Green graph (WAKE_UP) shows the external wake up signal. The yellow graph (UC_WAKE_UP) shows the wake up signal applied to the rest of the circuit, which goes high only if the external wake up signal is high long enough. The blue graph (UC_SLEEP) is a signal that resets the internal wake up signal (UC_WAKE_UP). The orange graph (UT1) shows the output of the XOR gate. The purple graph (UT2) shows the output of the RC time delay.

As we can see from Figure 6.1, the external wake up signal needs to be active for a certain time before the internal wake up signal goes high. We can also see that the capacitor is discharged through the discharging transistor M1 when the external wake up signal goes low.

The SPICE files used for the simulations are available in Appendix B.

## 6.2 - Power Saving Analyzes

The circuit that is simulated is the asynchronous wake up circuit in Figure 5.6. The reset circuit of the AVR must also be on during the power off mode, as the circuit must always be able to have a power off reset and possibility for external reset. The power usage for the reset circuit of the AVR today is confidential, and can therefore not be published. The ATmega329P microcontroller uses 40nA in sleep mode at 1.8V [13]. By assuming that the reset circuit uses half of this current, the total current drained from a microcontroller using the proposed system can be assumed to be 20nA plus the current drained from the proposed circuit.

Simulations show that the average current drained by the proposed circuit is 1.2275nA at 3.3V. When adding the current drained from the reset circuit, the total current drained from the microcontroller during a power off mode is 21.2nA. This is clearly much less than the circuits today. The current drained from ATmega329P, which is a "pico power" microcontroller, is 40nA at 1.8V [13]. The estimated power usage of the proposed solution, compared to an existing microcontroller is found by taking the total current drained from the proposed solution, $I_{PR}$ divided by the current drained from an existing circuit, $I_{EXISTING}$. Since the power usage is promotional to the current drained, the power saving ($P_{SAVING}$) in percent is shown in Formula 6.1.

$$\boxed{P_{SAVING}(\%) = 100\% - \frac{I_{PR}}{I_{EXISTING}} \bullet 100\% = \frac{21.2275nA}{40nA} \bullet 100\% \approx 47\%}$$

**Formula 6.1 - Power saving in power off mode using the proposed solution.**

The proposed circuit has been compared to ATmega329p, which uses a transistor technology with low leakage, and the proposed circuit is simulated with relatively high leakage. Therefore the comparison between these two is not completely right. I have chosen to compare the proposed circuit to ATmega329p because I want to show that it is possible to reduce the power consumption, even if high leakage transistors are used.

Simulations of power consumption in other lighter sleep modes are more complex, and needs more understanding and research of the AVR structure. Therefore, this assignment only simulates the power consumption of the power off mode.

# 7 - Discussion

When using aggressive transistor technology, the transistors become so small that they cannot withstand high voltages, such as 5V or 3.3V. Therefore, there is a need for using multiple power domains, so the microcontroller can be used with external voltages up to 5.5V, which is the automotive standard. The aggressive transistor technology also introduces high leakage currents.

Reducing the power consumption of a circuit with transistor technology with high leakage currents is quite a challenge. In convention CMOS technology, the static power consumption is so little, that power is mostly reduced by optimizing the analog circuits. With the new transistor technology, the challenge of the digital designer is getting larger as power saving techniques must be taken much more into consideration.

Reducing the active mode power consumption is positive, but for a microcontroller such as the AVR, the sleep mode power consumption is much more important to reduce. This is because a microcontroller spends so much time in sleep mode. During active mode, the transistors need to be active, and then reducing the leakage currents can be done by different types of doping, which is not a task for the digital designer. What can be done by the digital designer during active mode is to set different non active modules in sleep. By using sleep transistors the leakage currents are reduced, but not eliminated. By turning the power off at the module, and disconnect the signal lines, the leakage currents are eliminated. This requires more digital logic, and the modules need a start up time after getting the power back.

Reducing the sleep mode power basically means turning off the power. This seems easy, but to get the microcontroller to wake up again, some modules must be on at all times. If a slow wake up is allowed, the proposed asynchronous circuit (Figure 5.6) can be used. This allows most of the microcontroller to be shut off. This saves a lot of power, but the start up time is relatively high. By using the asynchronous wake up, the wake up sources are limited. If a timed wake up is needed, the WD/PMU oscillator must be on. This reduces the efficiency of the sleep mode.

The multiple power domain solution is used because of the reduced area, but introduces higher leakage and more complexity. When designing the multiple power domain, there is a tradeoff between area, power and performance. If very low power usage and a good performance are needed, there must be done much more research to reduce the leakage currents before the goal is accomplished. This assignment has shown that reducing the power is possible, but introduces a time delay when waking up from a sleep mode. There is also a little delay when entering the sleep mode, but this has usually no consequences.

Using one high voltage and one low voltage area, as in chapter 4.3 - My solution, can be effective when considering power usage, but the delay introduced when waking the microcontroller from sleep may be too much. The microcontroller should therefore be designed with several sleep modes, so that time critical applications can use the microcontroller. The main problem is still the voltage regulator. If the voltage regulator efficiency is close to 100%, there is no need to shut it off, and almost the entire chip can be implemented in the low voltage area (4.2 - Principle 2: One Digital Low Voltage Power Domain). This will reduce the time delay when waking the microcontroller up from a sleep mode.

The simulations of the proposed circuit show that it is possible to reduce the power consumption considerably to an AVR microcontroller by using the suggested solution of multiple power domains. The start up time may be high, but it is up to the users of the microcontroller to decide if they want to use the power off mode, or use a less deep sleep mode.

In a multiple power domain, the different power domains must be able to communicate. This communication is set to be asynchronous in this assignment. The high voltage modules do not need a fast clock source, since there is no large processing power needed here. This saves a lot of power. In addition, if the clock is shared between the power domains, the level shifting might introduce a major clock skew.

# 8 - Conclusion

Reducing the power when transistor technology introduces large leakage currents is a great challenge. When transistors become smaller, they cannot withstand the high voltages used in microcontroller systems today. To be able to continue using these high voltages with new transistor technology, a multiple power domain is one of the solutions.

The multiple power domain solution in this assignment is two domains, using a voltage regulator to control the second voltage. The high voltage domain uses the external voltage, which can be up to 5.5V. By creating two domains, the new transistor technology can be used. Since the AVR 8-bit microcontroller is a quite small device, and low power is essential, introducing more than two power domains using different voltages can be ineffective, mainly because of the voltage regulators, which are ineffective.

The leakage currents are the biggest challenge, since they are a major part of the total power consumption with new transistor technology. They can be reduced by special doping techniques, but not eliminated. To reduce the power consumption effectively, modules must be kept in sleep, or turned off when inactive. To secure zero leakage current, the power must be disconnected from the circuit. This gives the circuit a certain start up time, which is a disadvantage. By grouping modules together on several voltage islands, the PMU can switch off several voltage islands when they are not in use. This gives the opportunity to run the parts of the chip that is required for the application.

Since most of the applications using a microcontroller such as the 8-bit AVR spend a lot of time in sleep mode, the sleep mode power consumption will be the most contributing to the total power consumption. Introducing the asynchronous wake up circuit (Figure 5.8), almost the entire microcontroller can be turned off, and such a mode would be called a power off mode. The disadvantage is the start up time, since this mode switches off the voltage to almost the entire chip, and all oscillators. There will be almost no power consumption though. The microcontroller would then be perfect for battery driven applications, since the low power consumption will contribute to a long battery lifetime.

This assignment has pointed out some of the major concerns and difficulties of creating a multiple power domain system. There might still be several pitfalls to such a design, and therefore further research must be done to make sure that the design will work properly.

The simulations shows that the proposed asynchronous wake up circuit uses very little power, and if a low power reset circuit is made, the power usage could be considerably reduced compared to circuits today.

Future work on a multiple power domain should contain research on an effective voltage regulator, this way the voltage of the low voltage domain can be held on during a power off mode, and this will probably reduce the start up time. This will also reduce the power consumption during a start up, because the capacitors do not need to be charged again. The leakage current can be reduced by doping and use of sleep transistors. Reducing the core voltage to a minimum would reduce the power consumption, but not eliminate the problem. Grouping modules together on the voltage islands is not easy, as different applications uses different modules. This requires some attentions though in future work.

# 9 - References

[1] : Kaushik Roy, Saibal Mukhopadhyay, Hamid Mahmoodi-Meimand
“Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits”
PROCEEDINGS OF THE IEEE, VOL. 91, NO. 2 (FEBRUARY 2003)
http://ieeexplore.ieee.org/iel5/5/26532/01182065.pdf
(7.feb 2008)

[2] : Tormod Njølstad
“Introduction to SIE40AA Low Power Digital Design”, Introduction to course “Low Power Design” at NTNU (August 2002)
http://www.iet.ntnu.no/courses/tfe01/pdfs/sie40aa_A4.pdf
(8.feb 2008)

[3] : David A. Johns, Ken Martin
“Analog Integrated Circuit Design” (1997)
John Wiley & Sons, Inc
ISBN: 0-471-14448-7

[4] : Adel S. Sedra, Kenneth C. Smith
“Microelectronic CIRCUITS”, fifth edition. (2004)
Oxford University Press
ISBN: 0-19-514252-7

[5] : Jonas Carlsson
“Contribution to Asynchronous Communication Ports of GALS Systems” (2006)
Lindköping studies in science and technology
ISBN: 91-85643-30-0

[6] : Herman Schutte
“Bi-directional level shifter for I$^2$C-bus and other systems” (August 1997)
http://wwwasic.kip.uni-heidelberg.de/lhcb/Publications/external/AN97055.pdf
(13.feb 2008)

[7] : Ryan Donohue
“Synchronization in Digital Logic Circuit” (2003)
http://www.stanford.edu/class/ee183/handouts_spr2003/synchronization_pres.pdf
(10.mar 2008)

[8] : Deepak Kumar Tala
“Asic-World” Website about digital design.
http://www.asic-world.com/tidbits/metastablity.html
(5.mar 2008)

[9] : S. C. Tan, X. W. Sun
“Low power CMOS level shifters by bootstrapping technique”
Electronic letters, Vol. 38, No. 16
(August 2002)
http://ieeexplore.ieee.org/iel5/2220/22201/01033821.pdf
(12.mar 2008)

[10] : Ji-Hoon Lim, Jong-Chan Ha, Won-Young Jung, Yong-Ju Kim, Jae-Kyung Wee
“A Novel High-Speed and Low-Voltage CMOS Level-Up/Down Shifter Design for Multiple-Power and Multiple-Clock Domain Chips” IEICE TRANS. ELECTRON., VOL.E90–C, NO.3 (March 2007)
http://ietele.oxfordjournals.org/cgi/reprint/E90-C/3/644.pdf
(12.mar 2008)

[11] : Arne Martin Holberg, Åsmund Sætre
“Innovative Techniques for Extremely Low Power Consumption with 8-bit Microcontrollers” (Feb 2006)
http://www.atmel.com/dyn/resources/prod_documents/doc7903.pdf
(7.april 2008)

| | | |
|---|---|---|
| [12] : | James R. Black, | “Electromigration – A Brief Survey and Some Recent Results” IEEE Transactions On Electron Devices, vol ED-16, NO-4, (April 1969) http://ieeexplore.ieee.org/iel5/16/31658/01475796.pdf (29.apr 2008) |
| [13] : | Atmel AVR ATmega329P datasheet | http://www.atmel.com/dyn/resources/prod_documents/doc8021.pdf (4.jun 2008) |
| [14] : | Farzan Fallah, Massoud Pedram | "Power Aware Design Methodologies" p 398-412, Springer US 2002 ISBN: 978-1-4020-7152-2 (Print) 978-0-306-48139-0 (Online) http://www.springerlink.com/content/n46592k7w3713rn1/?p=aaad45a535eb4e66825612ae049f495a&pi=12 (5.jun 2008) |

# Appendix A - List of Variables Used In Formulas

| Variable | Description | Unit |
|---|---|---|
| $V_{Supply}$ | Supply voltage for electronic circuits | Voltage [V] |
| $V_{DD}$ | Supply voltage for CMOS circuits | Voltage [V] |
| $\alpha$ | Activity factor, a number of how frequently the signals will change. $\alpha$ is in range 0 to 1. | |
| $V_{th}$ | Threshold voltage for MOS transistors, the amount of voltage needed from gate to source to make the transistor conduct | Voltage [V] |
| $V_{tn0}$ | The threshold voltage when bulk is connected to source | Voltage [V] |
| $V_{fb}$ | The flat band voltage, corresponds to the voltage which when applied to the gate electrode yields a flat energy band in the semiconductor. | Voltage [V] |
| ! | The body-effect constant | Root of voltage [ $\sqrt{V}$ ] |
| $V_{GS}$ | The voltage difference from the gate of the MOS transistor to the source | Voltage [V] |
| $V_{SB}$ | The voltage from source to bulk | Voltage [V] |
| $I_D$ | Drain current in CMOS devices, current flowing from drain to source | Current [I] |
| $I_{SC}$ | The current flowing from VDD to ground when CMOS gates are switching from logic H to L or L to H | Current [I] |
| $P_{avg}$ | Average power usage in circuits | Power [W] |
| $P_{leakage}$ | Power usage due to leakage currents in circuits | Power [W] |
| $P_{SC}$ | Power usage due to short circuit currents in circuits | Power [W] |
| $P_{dynamic}$ | The dynamic power usage, due to logic shifting from H to L or L to H in circuits | Power [W] |
| $C_{eff}$ | The effective capacitance in a circuit | Capacitance [C] |
| $C_L$ | The load capacitance of a CMOS gate | Capacitance [C] |
| $f_{clk}$ | The clock signal frequency | Hertz [Hz] |
| $\mu_n$ | ??? (process parameter) | |
| $C_{ox}$ | The capacitance of the gate oxide layer in the transistor (process parameter) | Capacitance [C] |
| W | The width of the transistors gate (process parameter) | Meters [M] |
| L | The length of the transistors gate (process parameter) | Meters [M] |
| Q | The amount of electrical charge the capacitors have stored | Joule [J] |
| ! $_F$ | Surface potential : $\phi_F = \frac{kT}{q} \bullet \ln(\frac{N_A}{N_I})$ [1] | Voltage [V] |
| $N_A$ | Acceptor concentration | Holes/m$^3$ [H/ m$^{3]}$ |
| $K_S$ | Relative permittivity of silicon | Farad/meter [F/m] |
| ! $_0$ | Permittivity of free space, $8.854 \times 10^{-12}$ | Farad/meter [F/m] |

# Appendix B – SPICE Simulation Files

## *Asynchronous_wake_up.cir*

```
* Aynchronous wake up curcuit

***************************
* Include the transitor models **
***************************
.lib 'eldo.mod' process_tolerances
.lib 'eldo.mod' mos_nom
.lib 'eldo.mod' techno
.lib 'eldo.mod' nonmc
.lib 'eldo.mod' nonmatching
.lib 'eldo.mod' model_58k85
.lib 'eldo.mod' rnom
.lib 'eldo.mod' cnom


***************************
* Include the xor gate, ********
* inverter and the SR latch ****
***************************
.include xor.cir
.include inverter.cir
.include nor.cir
.include srlatch.cir

***************************
* Supply, Vdd and Vss(GND)   **
***************************
Vdd Vdd Vss dc 3.3
Vss Vss 0 dc 0

***************************
* Signals ******************
***************************
*The signal vInn is connected to the wake_up pad
*vInn wake_up Vss pulse(0 3.3 10ns 0.1ns 0.1ns 30ns 100ns)
vInn wake_up Vss dc 0

*Sets the first SR latch so that the inputs to the XOR is
equal
vA uC_set Vss dc 0
vB uC_rset Vss dc 3.3

*Initiate a sleep with a pulse to the sleep node
vC uC_sleep Vss pulse(0 3.3 1ns 0.1ns 0.1ns 4ns 100ns)
*vD uC_sleep Vss dc 3.3
```

```
***************************
* Netlist of the proposed circuit *
***************************

*The XOR GATE
x1 Vdd Vss wake_up wake_up_def ut1 XOR

*The RC time delay
r1 ut1 ut2 10k
c1 ut2 0 2pF

*The discharging transistor
M1 ut2 discharge Vss Vss nmosox3 L=0.36u W=1u


*The buffer, mad of two inverters
x2 Vdd Vss ut2 ut2inv inverter
x3 Vdd Vss ut2inv ut3 inverter

*The inverter to control the discharging transistor
x6 Vdd Vss ut1 discharge inverter

*The SR latches
*The first SR latch sets the value of input to XOR
x4 Vdd Vss uC_set uC_rset wake_up_def SRLATCH

*The secound SR latch is reset by uC, and set by wake up this
wake_up circuit
x5 Vdd Vss ut3 uC_sleep uC_wake_up SRLATCH

*Need pull down resistors to the SR latches
r2 uC_set Vss 10k
r3 uC_rset Vss 10k
r4 uC_sleep Vss 10k

*A capacitive load for second SRLATCH
c2 uC_wake_up 0 12fF

***************************
*Simulating commands *******
***************************

*Simulate for 100ns, with 1ns resolution
.tran 1n 100n

*Plot the voltage of the following nodes:
.plot tran v(wake_up) v(uC_wake_up) v(uC_sleep) v(ut1) v(ut2)
.plot tran i(Vdd)

.extract average( i(Vdd), 50ns, 100ns)
```

## NOR.cir

```
*NOR

.subckt NOR Vdd Vss A B OUT

m1 Vdd B n1 Vdd pmosox3 L=0.36u W=3u
m2 n1 A OUT Vdd pmosox3 L=0.36u W=3u
m3 OUT A Vss Vss nmosox3 L=0.36u W=1u
m4 OUT B Vss Vss nmosox3 L=0.36u W=1u

.ends
```

## *INVERTER.cir*

```
*Inverter

.subckt inverter Vdd Vss Vinn Vut

*M1 Vdd Vinn Vut Vdd pm L=0.35u W=3u
*M2 Vut Vinn Vss Vss nm L=0.35u W=1u

*Test eldo
M1 Vdd Vinn Vut Vdd pmosox3 L=0.36u W = 3u
M2 Vut Vinn Vss Vss nmosox3 L=0.36u W = 1u
.ends
```

## XOR.cir

```
*XOR
.subckt XOR Vdd Vss A B OUT

xIA Vdd Vss A Ainv inverter
xIB Vdd Vss B Binv inverter

m1 Vdd Ainv x1 Vdd pmosox3 L=0.36u W=3u
m2 Vdd Binv x1 Vdd pmosox3 L=0.36u W=3u
m3 x1 A OUT Vdd pmosox3 L=0.36u W=3u
m4 x1 B OUT Vdd pmosox3 L=0.36u W=3u
m5 OUT A x2 Vss nmosox3 L=0.36u W=1u
m6 OUT Ainv x3 Vss nmosox3 L=0.36u W=1u
m7 x2 B Vss Vss nmosox3 L=0.36u W=1u
m8 x3 Binv Vss Vss nmosox3 L=0.36u W=1u

.ends
```

## SRLATCH.cir

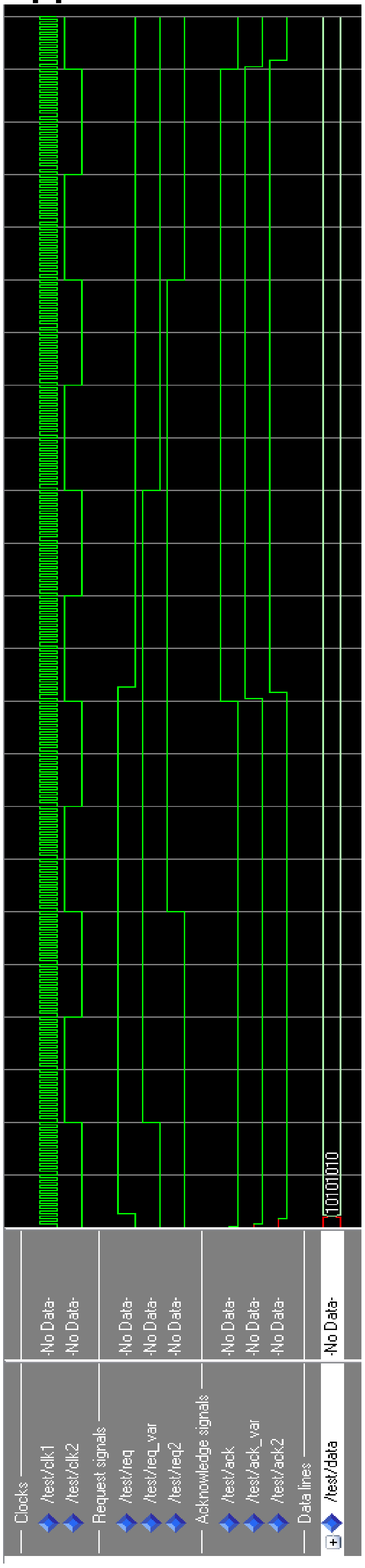
```
* Asynchronous SR latch based on two cross coupled NOR gates

.subckt SRLATCH Vdd Vss S R out

xNOR1 Vdd Vss S out n_out NOR
xNOR2 Vdd Vss R n_out out NOR


.ends
```

# Appendix C – Handshake Scheme Timing Diagram



The figure to the left shows the timing diagram of the handshake scheme with synchronization logic (Figure 3.1). The request is put by the module using the high speed clock (clk1). Then the request signal is synchronized to match the clock of the other module. We can see that the request signal (req) is set high at a random time, i.e. not at a clock flank of clk2. The signal enters two consecutive D flip flops. The output signal from the first D flip flop is req_var, and the output signal reaching the second module is req2. We can see that these two request signals are synchronized with clk2.

The data lines do not have synchronization blocks, and are outputted at the same time as the request signal. As soon as the recipient block gets the request signal, the data lines are latched into the data register. At the next positive clock flank, the recipient block sets the acknowledgement signal. The acknowledgement signal also enters two consecutive D flip flops, and ack_var and ack2 is synchronized with clk1.

When the acknowledgement signal has reached the sending block, the request signal is set low again. The recipient block will set the acknowledgement signal low when the low request signal has reached the block. When this signal has reached the sending block again, a new sending may occur.

The minimum time needed for a transfer is six clock periods of the low speed clock. From the figure to the left, we can see that at the sixed positive clock flank of clk2, the acknowledgement signal is reset. Two positive clock flanks of the clk1 after that, the sending block can see that the acknowledgement signal is reset, and another sending may occur.