Hoofdstuk 1

Implementatie

In dit hoofdstuk wordt de implementatie van een schakeling voor de berekening van Tate pairings uit de doeken gedaan. Zo zal onderzocht worden welke basisbewerkingen nodig zijn en hoe deze verwezenlijkt kunnen worden in hardware. Vervolgens wordt een schakeling ontworpen die aan de hand hiervan alle nodige berekeningen kan uitvoeren in het veld $\mathbb{F}_{2^m}$. Ten slotte is er dan nog de schakeling die alle berekeningen voor de Tate pairing in goede banen leidt. Allereerst wordt echter gekeken welke beperkingen aan de implementatie opgelegd moeten worden.

1.1 Beperkingen

Het doel is de uiteindelijke schakeling zo klein mogelijk te maken, zodat ze gebruikt kan worden in bv. netwerken van sensoren of smartcards. Stroomverbruik is een tweede belangrijke factor, maar is helaas zeer moeilijk te berekenen. Het verbruik hangt echter samen met de oppervlakte, dus het beperken daarvan zal ook het verbruik ten goede komen. Het verbruik kan ook verlaagd worden door een lagere kloksnelheid voor de schakeling te gebruiken, wat uiteraard de rekensnelheid niet bevordert. De uiteindelijk maximale kloksnelheid zal sowieso echter hoger zijn dan gewenst, dus kan dit aspect bij het ontwerp van de schakelingen genegeerd worden. Hier zal verder op ingegaan worden in Hoofdstuk **??**

Algemeen kan gesteld worden dat hoe kleiner het uiteindelijke resultaat, hoe beter. Het is dus cruciaal de elementen te identificeren die het meeste plaats innemen in een ASIC schakeling. In Tabel 1.1 is de grootte van de belangrijkste elementen te vinden. Deze cijfers gelden enkel bij gebruik van een $0.13nm$ technologie. De relatieve volgorde van de elementen blijft echter behouden voor andere technologiën. Uit de tabel blijkt dat het gebruik van zowel flip-flops (registers) als MUXs zoveel mogelijk beperkt moet worden.

1.2 Modular Arithmetic Logical Unit

De kern van de hardware implementatie wordt gevormd door de Modular Arithmetic Logical Unit (MALU)[**?**]. Dit circuit laat toe basis bewerkingen uit te voeren op getallen. Gezien de beperking die is opgelegd aan de oppervlakte

Tabel 1.1: Grootte van elementen in een ASIC schakeling (0.13nm technologie)

Element	Gates/bit
flip-flop	6-7
MUX	??
AND	??
XOR	2.75
OR	??
NOT	??

van de schakeling, wordt enkel de optelling geïmplementeerd. Later worden met behulp daarvan elke andere nodige berekening verwezenlijkt.

Aangezien er in het veld $\mathbb{F}_{2^m}$ gewerkt wordt, is een optelling equivalent aan een XOR bewerking. De bewerking die moet uitgevoerd kunnen worden is:

$$\begin{aligned} T + B &= T \otimes B \\ &= R \mod P \end{aligned}$$

Merk op dat bij een optelling de graad van R enkel kleiner of gelijk kan zijn aan die van T en B. Indien B van graad $\leq m$ is en T van graad $\leq m + 1$, is de modulo bewerking te implementeren als in Algoritme 1.1.

Algoritme 1.1: Modulo optelling in $\mathbb{F}_{2^m}$

Input: $B \in \mathbb{F}_{2^m}$, $T \in \mathbb{F}_{2^{m+1}}$
Output: $R \mod P \in \mathbb{F}_{2^m}$
$R \leftarrow T \otimes B$
if $degree(T) = m$ **then**
  $R \leftarrow R \otimes P$

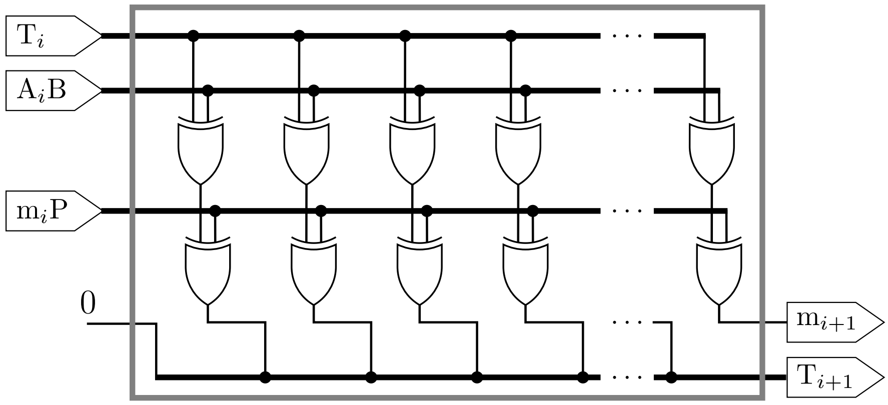
Een voor de hand liggende schakeling die dit alles implementeert, is te zien in Figuur 1.1. Ingang P_{in} dient afhankelijk van T_{MSB} ingesteld te worden op 0 of P. P_{MSB} (graad m) kan genegeerd worden, aangezien in het resultaat de graad $\leq m$ is.

In Sectie 1.3 zal blijken dat het vaak nodig zal zijn om het resultaat R te vermenigvuldigen met z, maw. alle bits 1 plaats naar links te verschuiven. Indien die bewerking wordt toegevoegd, bekomt men de schakeling uit Figuur 1.2. Net als de vorige implementatie), bestaat deze uit $2m$ XOR poorten.

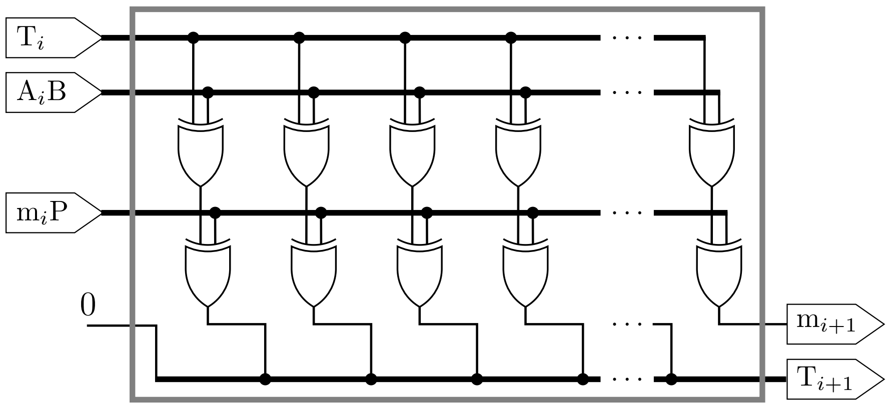
Aangezien voor het ontwerp het veld en de modulo veelterm op voorhand bepaald zijn, is het mogelijk een zeer groot aantal XOR poorten uit het ontwerp te verwijderen. Het is immers niet nodig dat de ingang P_{in} m bits breed is, aangezien P gekend is. De ingang en de bijhorende m XOR poorten kunnen vervangen worden door een 1 bit 'modulo enable' ingang *mod* en enkel XOR porten voor de bits i waarvoor $P_i = 1$. Hierdoor worden

$$\Delta = m - (\text{hamm}(P) - 1)$$

XOR poorten uitgespaard, met hamm(P) gelijk aan het Hamming gewicht van de binaire represenatie van P.



Figuur 1.1: MALU - Basis ontwerp



Figuur 1.2: MALU - Basis ontwerp met shift

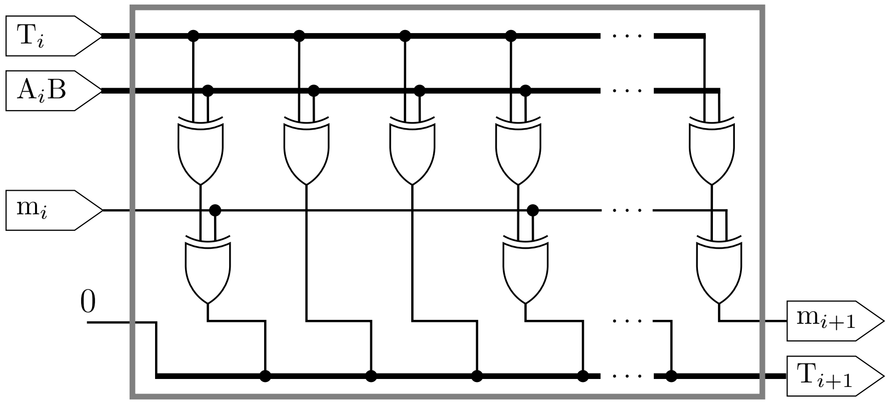
In dit geval is $m = 163$ en $P = z^{163} + z^7 + z^6 + z^3 + 1$. Er zijn dus $\mathrm{hamm}(P) - 1 = 4$ XOR poorten nodig, wat een besparing van $163 - 4 = 159$ XOR poorten oplevert (51% kleiner dan de oorsponkelijke grootte).

De resulterende schakeling is te zien in Figuur 1.3.

1.3 Berekeningen in $\mathbb{F}_{2^m}$

De eerder ontworpen MALU schakeling laat toe optellingen te doen, maar het Miller algoritme vereist dat er ook vermenigvuldigingen worden uitgerekend. Delingen en machtsverheffingen kunnen met behulp van vermenigvuldiging berekend worden en dienen dus niet rechtstreeks geïmplementeerd te worden. Indien dus zowel optellingen als vermenigvuldigingen berekend kunnen worden, is alles voorhanden om de Tate pairing te berekenen.

Door toepassing van een “shift and add” algoritme, kan de waarde van $A{\cdot}B = R$ berekend worden met behulp van de MALU schakeling. In Algoritme 1.2 is te



Figuur 1.3: MALU - Geoptimaliseerd ontwerp met shift

zien hoe dit juist in z'n werk gaat. Op het einde van elke iteratie is de graad van T gelijk aan $m + mod$. Door de modulo operatie telkens op het tussenresultaat uit te voeren, is het maximum van graad m en kan het steeds opgeslagen worden in T. Op het einde moet het resultaat door z gedeeld te worden, wat neerkomt op een verschuiving van alle bits met 1 plaats naar rechts.

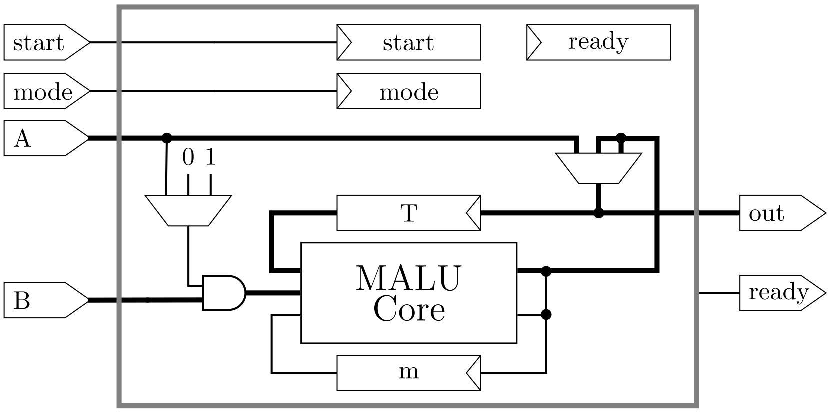
Algoritme 1.2: "Shift and add" vermenigvuldiging

Input: $A, B \in \mathbb{F}_{2^m}/[P]$
Output: $R \in \mathbb{F}_{2^m}/[P]$
Data: $T \in \mathbb{F}_{2^{m+1}}$
$T \leftarrow 0$
for $i \leftarrow m-1$ **to** 0 **do**
  if $A_i = 1$ **then**
    $b \leftarrow B$
  else
    $b \leftarrow 0$
  $T \leftarrow T \otimes b$
  if $T_m = 1$ **then**
    $T \leftarrow T \otimes P$
  $T \leftarrow T \ll 1$
$R \leftarrow T \gg 1$

Indien dit algoritme in een schakeling gegoten wordt, dienen enkele toevoegingen te gebeuren. Omdat een vermenigvuldiging langer dan één klokslag duurt, is het noodzakelijk een *start* ingang en *ready* uitgang te hebben. Ook moet worden bijgehouden of A en B opgeteld of vermenigvuldigd dienen te worden. Deze functie wordt vervuld door het register *mode*. Ten slotte moet het mogelijk zijn om R in T op te slaan, zodat het juiste resultaat niet verloren gaat.

Bij het hoog gaan van *start*, wordt nagegaan wat de waarde van *mode* is. Indien het om een optelling gaat, wordt A opgeslagen in T, anders wordt $T \leftarrow 0$ zoals in Algoritme 1.2. Bij een optelling is het resultaat na één klokslag beschikbaar aan de uitgang van het MALU blok. Het dient dan enkel nog 1 bit naar rechts verschoven te worden, net zoals moet gebeuren op het einde van een vermenigvuldiging.

Wanneer dit alles in rekening gebracht wordt, verkrijgt men uiteindelijk de schakeling in Figuur 1.4. Om het geheel overzichtelijker te houden, bevat register T een getal in $\mathbb{F}_{2^m}$ en wordt de waarde van T_m bijgehouden in het register *mod*.



Figuur 1.4: Schakeling voor berekeningen in $\mathbb{F}_{2^m}$