

FPGAにおけるSAS認証回路の設計と実装

指導教員

高橋 寛 教授
王森レイ 講師

令和4年2月10日提出

愛媛大学工学部情報工学科
情報システム工学分野
計算機/ソフトウェアシステム研究室

岡本 悠

目次

第 1 章	まえがき	1
1.1	研究背景および目標・目的	1
1.2	論文の構成	2
第 2 章	JTAG のセキュリティ脆弱性	3
2.1	JTAG の構造	3
2.2	JTAG アクセスポートの脆弱性と対策	4
第 3 章	FPGA	5
第 4 章	SAS-L2	7
4.1	ワンタイムパスワード認証方式	7
4.2	SAS-L2 の概要	7
4.3	SAS-L2 初期登録処理	8
4.4	SAS-L2 認証処理	9
第 5 章	SAS-L2 を用いた JTAG 認証プロトコル	11
第 6 章	SAS 認証回路の設計	13
6.1	SAS 認証回路の設計手順	13
6.2	SAS 認証回路の概要	14
6.3	状態遷移	15
6.4	SAS 認証回路に必要なモジュールの設計	17
6.4.1	データ転送モジュール	17
6.4.2	ステートマシン	19
6.4.3	メモリ・アドレスレジスタ	21
6.5	モジュール間の接続	23

第 7 章 実装	24
7.1 検証方法・実装対象	24
7.2 シミュレーション結果	25
7.3 FPGA における実装結果	29
第 8 章 考察	31
第 9 章 あとがき	33
謝辞	34
参考文献	35

第1章 まえがき

1.1 研究背景および目標・目的

近年、IoT 技術を含む情報通信技術の発展に伴い、高性能・多機能化により IC の構造は複雑になっている。IC のテストや評価を容易に行うためにはテスト・デバックインフラである JTAG が必要である。

しかしながら、IC のハードウェアセキュリティ問題に JTAG が原因となるものがある。このセキュリティ問題には、外部から権限のないユーザがターゲット IC の JTAG 機構に対してリバースエンジニアリングを行い、JTAG 機構へのアクセス権限を取得したうえ、JTAG 機能のデバック機能を悪用することでターゲット IC に対して攻撃を仕掛けるという脆弱性がある。このような攻撃を防ぐ方法として、JTAG ポートに認証機構を用いることで正しく認証されたユーザのみに対してアクセスを許可するという方法が挙げられている。認証機構には共通鍵暗号方式を用いた認証方式や、チャレンジレスポンス認証に基づいた認証方式が提案されている。^[1]

一方で、IoT デバイスは低スペックのマイコンやセンサ、低消費電力の通信モジュールがほとんどであり、一般的に非常に低いコストで製造されているため、複雑な暗号回路を含む認証機構などのコストのかかるセキュリティハードウェアを設けることは困難である。

IoT において、セキュリティによる保護のないデバイスがインターネットに接続されてしまうと、IoT システム全体の安全性を損なうことになる。

そこで、本研究では IoT デバイスにおいて極めて小さい処理負荷で暗号鍵の配送が実現できるワンタイムパスワード認証方式である SAS-L2 を用いて、JTAG の認証機構の軽量化手法を提案する。目標としては、SAS-L2 をハードウェア化した SAS 認証回路を設計し、FPGA に実装する。

1.2 論文の構成

本論文の構成は以下の通りである。第2章ではJTAGのセキュリティ脆弱性についての説明を述べる。第3章ではFPGAの説明を述べる。第4章ではSAS-L2の説明を述べる。第5章ではSAS-L2認証を用いたJTAG認証プロトコルの説明を述べる。第6章ではSAS認証回路の設計の概要を示す。第7章ではSAS認証回路のFPGAへ実装の概要について示す。第8章では実装したSAS認証回路についての考察を示す。第9章では本研究のまとめを行う。

第2章 JTAG のセキュリティ脆弱性

本章では JTAG と JTAG アクセスポートのセキュリティ脆弱性についての説明を述べていく。

2.1 JTAG の構造

JTAG とは 4 本または 5 本の外部端子のみで IC 内の回路にアクセスするためのシリアルアクセスポートの標準規格であり、IC のテスト・評価を行うためや FPGA に回路情報を書き込むために利用されている。^[1]

図 2.1 に JTAG 対応デバイスの例を示す。

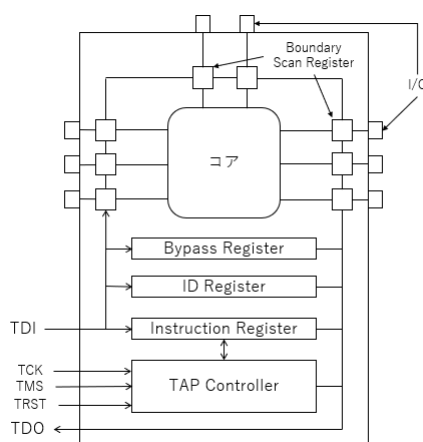


図 2.1 JTAG 対応デバイスの例

図 2.1 を用いて JTAG の構造について述べていく。

まず、JTAG のインタフェース信号には TCK、TDI、TDO、TMS の JTAG において必須な 4 本の信号と TRST というオプションの信号が存在し、これらをまとめて TAP(Test Access Port)と呼ぶ。TCK はクロック、TDI はデータ入力、TDO はデータ出力、TMS は TAP コントローラの状態遷移に用いられる。

次に、JTAG のレジスタにはインストラクションレジスタとデータレジスタの 2 種類が存

在する。

インストラクションレジスタでは、現在の命令を保持しており、TAP コントローラは保持されている命令を用いて受信した信号の処理を決定する。

データレジスタではバウンダリスキャンレジスタ (BSR) がデバイスの I/O ピンとの間に配置され、これらをチェーン状に接続してシフトレジスタを構成する。このシフトレジスタを利用してデータの読み書きを行っている。BSR のほかに必須となる Bypass レジスタやデバイスの部品番号や部品のバージョンコードなどを格納している ID レジスタの3つが存在している。

最後に、TAP コントローラでは TMS 信号によって状態遷移を制御しているステートマシンであり、JTAG の動作を制御している。^[2]

2.2 JTAG アクセスポートの脆弱性と対策

JTAG への不正アクセスを防ぐために JTAG インフラストラクチャーにロッキング回路を追加することで TAP コントローラを制御するアクセス認証機構が提案されている。

この認証機構では通常は TAP をロッキング回路によってロックしている状態である。そして、クライアントは JTAG ポートをアクセスする際にパスワードの入力が求められ、デバイスに格納されている認証データと照合し、一致した場合に TAP を解除し、全ての JTAG インフラストラクチャーへのアクセスを許可する。

また、盗聴などによるパスワードの漏洩を防ぐために共通鍵暗号や公開鍵暗号などの暗号回路を導入した認証機構も提案されている。

一方で、IoT システムにおいて、従来の JTAG 認証方法が以下の課題に直面している。

まず、ハードウェアコスト面に課題が存在する。デバイス側に JTAG アクセス認証を実現するには暗号回路を含む専用のハードウェアリソースを追加する必要がある。IoT システムにおいては低コストで生産されていることから低スペックである製品がほとんどであり、コストのかかる認証専用のハードウェアを実装することは困難である。

そして、セキュリティ面にも課題が存在する。IoT システムではデバイスが現場に設置され、デバイス間で頻繁なデータのやり取りが行われる。そのため盗聴により大量のデータを容易に得ることが可能であるため、総当たり攻撃が容易に行われる。従来法では認証データはデバイス側に格納し、固定されている。そのため暗号化された認証データが盗聴された場合、総当たり攻撃により認証データが割り出されてしまうリスクがある。^[3]

第3章 FPGA

本章では本研究で使用するデバイスである FPGA の説明を述べていく。

FPGA は Field Programmable Gate Array の略で、現場でプログラム可能な論理回路の多数配列という意味であり、ハードウェア言語 (VHDL、Verilog HDL など) を用いることで内部の回路を書き換えることが可能なデバイスである。

FPGA はロジックセル (LC) と呼ばれる回路ブロックを格子状に並べ、それぞれのブロックをハードウェア言語によるプログラムで配線することで回路を実現している。

以下の図 3.1 に FPGA の構造を示す。

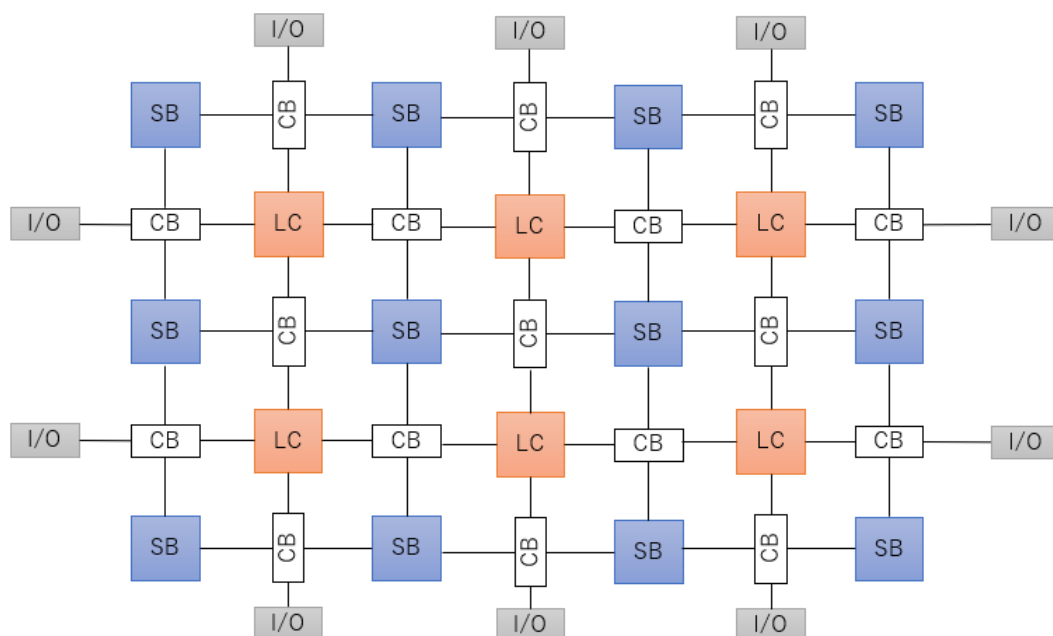


図 3.1 FPGA の構造

図 3.1 が示すように FPGA は LC、I/O、SB、CB から構成されている。

まず、LC はロジックセルを示している。ロジックセルでは入力に対して出力を割り当てるテーブルを小規模な RAM に書き込んでおくことで論理回路を実現しており、この RAM のことを LUT(Look Up Table) と呼ぶ。

次に、I/O は外部からの入出力を示しており、CB はコネクションブロックと呼び、多数のロジックセルや I/O を接続するための配線を行っている。そして、SB はスイッチブロックと呼び、信号をどこに送るかを切り替えており、CB だけでは実現できない柔軟性を補っている。^[4]

また、FPGA は RAM によって構成されているため、電源を切ると書き込まれている回路情報は消えてしまう。このため、FPGA を使用する際は外部に不揮発性メモリを用意し、電源投入時にこのメモリから回路情報を読み込んで動作を始める。

最後に、FPGA にはその場で回路情報を書き換え、回路をダウンロードさせてすぐに動作させることが可能であるという特徴があるため、FPGA を用いて開発を行う際にエラーが判明した場合でもその場で修正可能であるというメリットがある。^[5]

第4章 SAS-L2

本章では、本研究で使用するワンタイムパスワード認証方式である SAS-L2 の説明を述べていく。

4.1 ワンタイムパスワード認証方式

ワンタイムパスワード認証方式とは、認証の際に利用されるパスワードの盗聴を防ぐために、認証を行うごとにパスワードを使い捨てる認証方式である。この認証方式は認証(サーバ)側が被認証(クライアント)側の資格認証に用いられるだけでなく、認証ごとに変化する認証情報を元に暗号鍵を生成し、サーバ側とクライアント側で共有することで暗号通信を実現することにも用いることが可能である。

4.2 SAS-L2 の概要

SAS-L2 は Simple And Secure password authentication protocol Light processing type 2 の略で、高知工科大学の清水明宏教授が提案したワンタイムパスワード認証方式である。この認証方式の特徴はクライアント側での演算は排他的論理和 2 回と加算 2 回 (更新を除くと 1 回) のみで認証処理が実行できる点である。

4.3 SAS-L2 初期登録処理

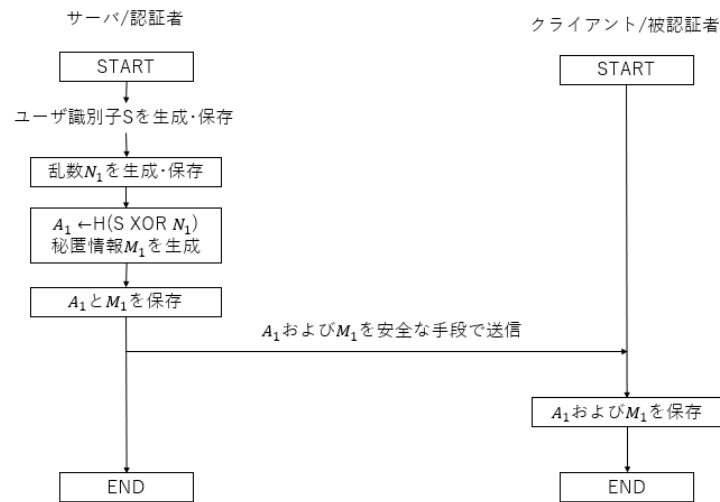


図 4.1 SAS-L2 初期登録処理

図 4.1 は SAS-L2 の初期登録手順についてのフローチャートを示しており、初期登録処理は以下のような処理を行う。

- (i) サーバ側で乱数を用いて初回秘匿情報 M_1 と、ユーザ識別子 S (ID とパスワード) を生成し保存を行い、乱数 N_1 と一方向性関数 H (ハッシュ関数など) を用いて暗号化し、初回認証情報 A_1 を生成する。
- (ii) サーバ側で生成された初回秘匿情報 M_1 と初回認証情報 A_1 を安全な手段でクライアント側と共有する。

4.4 SAS-L2 認証処理

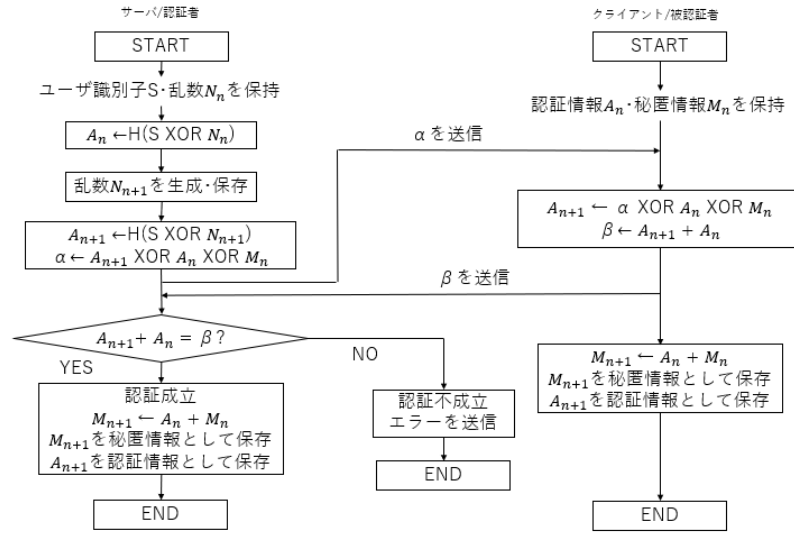


図 4.2 SAS-L2 認証処理

図 4.2 認証手順を示したものであり、認証処理では以下のような処理を行う。

- (i) サーバ側で新たに乱数 N_{n+1} を生成し、これと一方向性関数 H を用いて保存していたユーザ識別子 S を暗号化し次回認証情報 A_{n+1} を生成する。
- (ii) サーバ側で以下の演算を行い送信データ α を生成し、クライアント側へ送信する。

$$\alpha = A_{n+1} \oplus A_n \oplus M_n \quad (4.1)$$

- (iii) クライアント側では受信した α を保存していた認証情報と秘匿情報である A_n と M_n と以下のように排他的論理和を取ることで、サーバ側で生成された A_{n+1} を復号し、一時的に保存する。

$$A_{n+1} = \alpha \oplus A_n \oplus M_n \quad (4.2)$$

- (iv) クライアント側で以下の演算を行い送信データ β を生成し、サーバ側へ送信する。

$$\beta = A_{n+1} + A_n \quad (4.3)$$

- (v) サーバ側でも $A_{n+1} + A_n$ の演算を行い、演算結果が受信した β と等しい場合は認証成功となる。

- (vi) 認証成功の場合はクライアント側に認証成功を送信し、サーバ側とクライアント側の秘匿情報 M_n を以下の演算で更新し、認証情報 A_n を A_{n+1} に更新する。

$$M_{n+1} = A_n + M_n \quad (4.4)$$

$$M_n = M_{n+1} \quad (4.5)$$

- (vii) 認証失敗の場合はクライアント側に認証失敗を送信し、秘匿情報と認証情報の更新を行わない。

第5章 SAS-L2 を用いた JTAG 認証プロトコル

本章では SAS-L2 を用いた JTAG 認証プロトコルについての説明を述べていく。

JTAG における SAS-L2 認証方式の実現には、IC 製品の出荷前にメーカーによるデバイスとユーザの識別情報（デバイス ID とパスワードなど）を暗号化し、初期認証情報としてデバイスのメモリに書き込む必要がある。また、IC 製品ごとのデバイス識別情報とユーザ情報を管理するために、メーカー側はデバイス管理サーバ（DMS）を用意することが必要である。

図 5.1 に SAS-L2 による JTAG アクセス認証のイメージを示す。

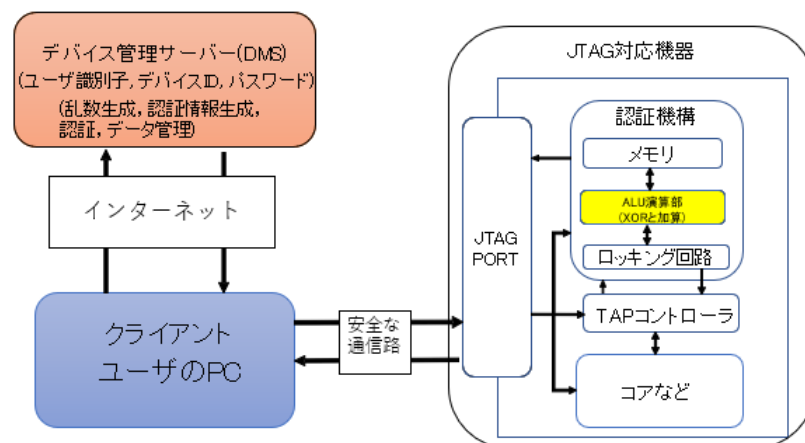


図 5.1 SAS-L2 による JTAG 認証プロトコル

IC 製品の出荷後、クライアントはオンチップデバッグ (OCD) ツールを用いて、以下の手順でデバイスに組込まれる JTAG にアクセスする。

- (i) クライアントが OCD ツールで製品メーカーのデバイス管理サーバにアクセスする。アクセスする際に別途の認証が求められる。
- (ii) クライアントが DMS においてターゲットデバイスへのアクセス要請を送信する。
- (iii) DMS がクライアントからのアクセス要請を受理し、当該デバイスの識別情報に対して、

認証データ A_{n+1} を新規に生成し、認証コード α を作成してクライアントに送信する。

- (iv) クライアントが DMS から受信した認証コード α を OCD ツールを用いて JTAG に送信する。
- (v) JTAG デバイス側では、クライアントが入力した認証コード α に対して、セキュアメモリに格納されている前回認証データ A_n と排他的論理和を取ることで、DMS 側で生成した新規認証データ A_{n+1} を解く。
- (vi) JTAG デバイスで、 A_n と A_{n+1} および α を用いて、認証データ β を生成し、クライアントに送信する。
- (vii) クライアントが β を DMS にアップロードして、認証データの照合処理を行う。
- (viii) 認証成立の場合は $A_n + A_{n+1}$ を、認証失敗の場合は乱数を認証結果としてクライアントに発行する。
- (ix) クライアントが DMS から受信した認証結果を JTAG に送信する。
- (x) JTAG デバイスにおいて、認証結果を $A_n + A_{n+1}$ と比較し、一致した場合はメモリのデータを A_{n+1} に更新し、アクセス権限を開放する。一致しない場合は更新せず、TAP をロックしたままにする。

(viii)~(x) において、データ改ざんによる DMS と JTAG 機器間の認証情報の不整合が起きる恐れがあるため、DMS とクライアントおよび JTAG デバイスの間で安全な通信ルートを確保することが必要である。

以上のプロトコルは JTAG 対応機器に強力なセキュリティ対策をより低コストで実装することが可能であると考えられている。その理由を以下に述べる。

まず、コスト面について述べていく。デバイス側では排他的論理和と加算などといった簡単な演算回路のみが必要となるため、認証用ハードウェアが少ない。また、乱数や一方向性関数といったワンタイムパスワード認証で必要とされる演算は DMS に集中するので少ないソフトウェアで大量のデバイスに対して認証の実行が可能である。

次に、セキュリティ面について述べていく。暗号化された認証情報は認証が行われるごとに更新されるため、総当たり攻撃に強く、データのやり取りが頻繁に行われる IoT システムに適している。また、ワンタイムパスワード認証に必要な認証情報の生成は DMS で行われているため、必要に応じて生成アルゴリズムの強化が可能であり、セキュリティが強いと言える。^[3]

第6章 SAS 認証回路の設計

本章では SAS 認証回路の設計について述べる。

6.1 SAS 認証回路の設計手順

今回 FPGA に実装する SAS 認証回路は SAS-L2 認証処理のクライアント側をハードウェア化したものであり、その設計方法を以下に示す。

- (1) ハードウェア全体の入力と出力を示した構成図を作成する。
- (2) ハードウェアを制御するための状態を把握するために状態遷移図を作成する。
- (3) 必要なモジュールの設計図を作成する。
- (4) モジュール間の接続を示した図を作成する。

以上の順に設計を進めていく。

6.2 SAS 認証回路の概要

はじめに SAS-L2 のクライアント側の認証処理から SAS 認証回路全体の構成図を以下の図 6.1 に示す。

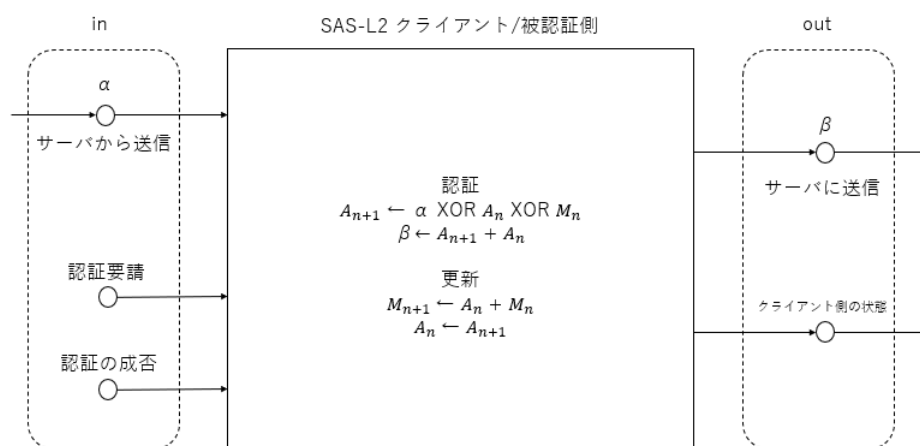


図 6.1 ハードウェアのイメージ図

図 6.1 が示すように今回実装するハードウェアは認証要請を受け取り、サーバで算出した α を受信し、サーバで認証を行うために必要な β を生成したのちサーバに送信する。このとき、クライアント側の状態をサーバへ送信することで、 α の受信と β の送信のタイミングを合わせる。

そして、サーバから認証の成否を受信し、認証成功であれば認証情報 A_n と秘匿情報 M_n をサーバと合致させるための更新作業を行い、認証失敗であればそのまま次の認証要請を受信するまで待機するように設計する。

また、今回サーバとクライアントでやり取りするデータである α と β は一般的に最適な暗号鍵のサイズといわれている 256bit に設定する。

6.3 状態遷移

次に SAS-L2 のクライアント側の認証処理を実装するために必要なハードウェアの状態について整理し、以下の表 6.1 に示す。表 6.1 中の Q2～Q0 は状態制御信号を表している。

表 6.1 SAS 認証回路の状態

Q2	Q1	Q0	状態
0	0	0	S0 : 待機
0	0	1	S1 : α 受信
0	1	0	S2 : A_{n+1} 生成 1
0	1	1	S3 : A_{n+1} 生成 2
1	0	0	S4 : β 生成
1	0	1	S5 : β 送信
1	1	0	S6 : M_n 更新
1	1	1	S7 : A_n 更新

状態 S2、状態 S3、状態 S4、状態 S6 および状態 S7 は以下のような演算を行っている。

$$S2 : N = \alpha \oplus A_n \quad (6.1)$$

$$S3 : A_{n+1} = N \oplus M_n \quad (6.2)$$

$$S4 : \beta = A_{n+1} + A_n \quad (6.3)$$

$$S6 : M_n = A_n + M_n \quad (6.4)$$

$$S7 : A_n = A_{n+1} \quad (6.5)$$

次に図 6.2 に状態遷移を示した状態遷移図を示す。

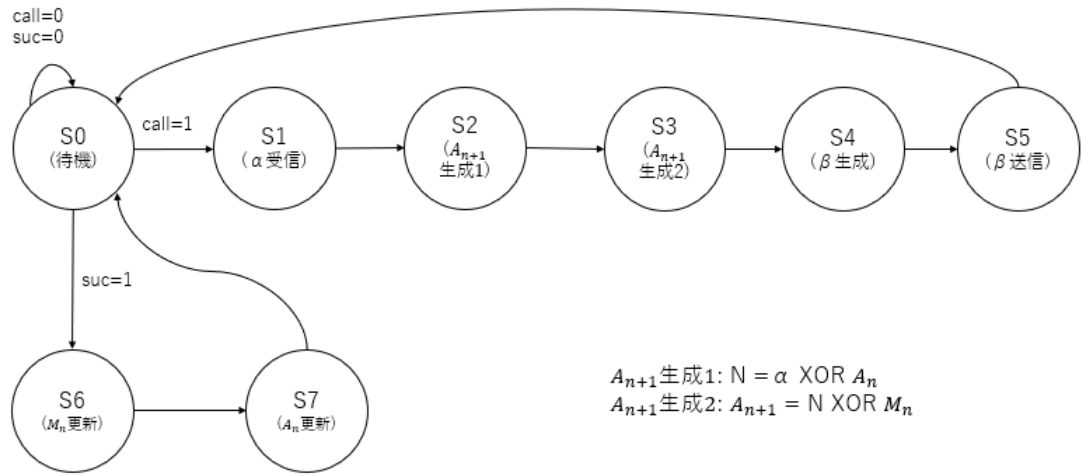


図 6.2 状態遷移図

ハードウェアの状態は待機 (S0) と認証 (S1～S5)、更新 (S6、S7) の3つに分けられる。待機から認証へ遷移する条件は、認証要請 $call$ が $call = 1$ となるときであり、 β の送信が終わると認証から待機に遷移する。待機から更新へ遷移する条件は、サーバから認証成功を表す $suc = 1$ を受信したときであり、 M_n の更新、 A_n の更新の順番で更新を行ったのちに待機へ遷移する。

認証では、まず $call = 1$ を受信すると S1 に遷移し、サーバから α を受信し、受信が終わると S2 に遷移する。次に S2～S4 では受信した α を A_n と M_n を用いて排他的論理和と加算を用いることで β を生成する。このときの状態遷移条件は演算の終了である。最後に S5 では生成した β をサーバへ送信し、送信が終わると S0 へ遷移して認証結果を待つ。

更新では、サーバから認証成功を表す $suc = 1$ を受信したときに S6 へ遷移し、S6 では M_n を更新、S7 では A_n を更新する。このとき、式 6.4 からわかるように M_n の更新には更新前の A_n が必要となるので、必ず M_n を更新してから A_n を更新しなければならない。また、遷移条件は、S2～S4 での遷移条件と同様である。

6.4 SAS 認証回路に必要なモジュールの設計

今回実装する SAS 認証回路に必要なモジュールを述べていく。

6.4.1 データ転送モジュール

まず、データの転送を行うモジュールが必要であり、今回は 256bit のデータのやり取りを行う。サーバで α を生成する際に、unsigned int 型で大きさが 8 の配列で設定する。そのため、クライアント側では 32bit ずつ受信し、256bit に変換して演算器に送信するためのモジュールが必要となる。そして、 β も α と同様にサーバでは unsigned int 型で大きさ 8 の配列で扱うため、演算器から 256bit の結果を 32bit ずつに分割してサーバへ送信するためのモジュールが必要である。

以上を踏まえて、設計したデータ受信モジュールを図 6.3 と表 6.2 に、データ送信モジュールを図 6.4 と表 6.3 に示す。

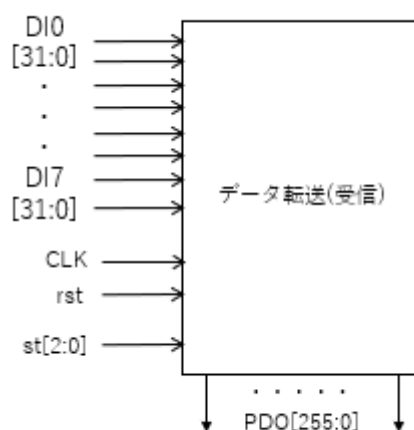


図 6.3 データ受信モジュール

表 6.2 データ受信モジュールの入出力

信号名	I/O	備考
CLK	in	システムクロック
rst	in	リセット信号
DI0	in	受信データ α (32bit)
DI1	in	受信データ α (32bit)
DI2	in	受信データ α (32bit)
DI3	in	受信データ α (32bit)
DI4	in	受信データ α (32bit)
DI5	in	受信データ α (32bit)
DI6	in	受信データ α (32bit)
DI7	in	受信データ α (32bit)
st	in	状態制御信号
PDO	out	受信データ α (256bit)

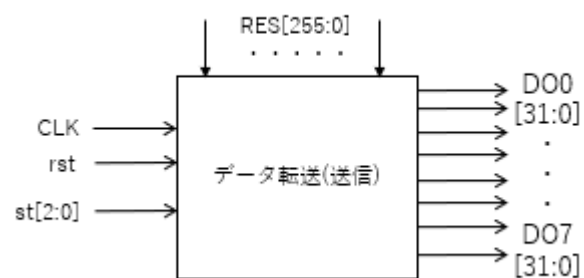


図 6.4 データ送信モジュール

表 6.3 データ送信モジュールの入出力

信号名	I/O	備考
CLK	in	システムクロック
rst	in	外部からのリセット信号
RES	in	送信データ β (256bit)
st	in	状態制御信号
DO0	out	送信データ β (32bit)
DO1	out	送信データ β (32bit)
DO2	out	送信データ β (32bit)
DO3	out	送信データ β (32bit)
DO4	out	送信データ β (32bit)
DO5	out	送信データ β (32bit)
DO6	out	送信データ β (32bit)
DO7	out	送信データ β (32bit)

6.4.2 ステートマシン

次に必要なモジュールは SAS 認証回路の状態遷移を実現するためのステートマシンが必要である。今回は、S0 から S1 への状態遷移と S0 から S6 へ遷移するとき以外は演算が終わると状態遷移を行うため、ステートマシンで演算も行えるように設計する。

さらに、S2～S4 ではメモリからデータを読み出し、S6 と S7 では演算結果をメモリに書き込まなければいけないので、メモリの読み出しモードと書き込みモードを変更させなければいけない。よって、ハードウェアの状態によってメモリのモード変更の制御が行えるように設計する。

また、今回はリセット信号を外部から直接各モジュールへ送信するのではなくこのステートマシンを通して各モジュールへ送信するように設計する。

これらを踏まえて設計したステートマシンを図 6.5 と表 6.4 に示す。

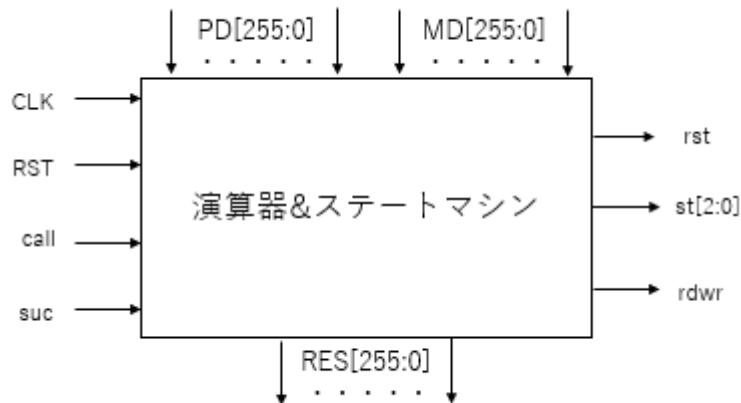


図 6.5 ステートマシン

表 6.4 ステートマシンの入出力

信号名	I/O	備考
CLK	in	システムクロック
RST	in	RST = 1 : リセット
call	in	認証要請
suc	in	認証の成否
PD	in	受信データ (α)
MD	in	メモリからの入力 (256bit)
st	out	状態制御信号 (3bit)
rst	out	リセット信号
rdwr	out	メモリの読み書き有効化信号 (read : 0、write : 1)
RES	out	演算結果 (256bit)

ステートマシン内部で行われている演算の変化について状態と演算の対応を以下の表 6.5 に示す。

表 6.5 ハードウェアの状態とステートマシン内部の演算の対応

ハードウェアの状態	演算内容
010(S2)	$N = \alpha \oplus A_n$
011(S3)	$A_{n+1} = N \oplus M_n$
100(S4)	$\beta = A_{n+1} + A_n$
110(S6)	$M_n = A_n + M_n$
111(S7)	$A_n = A_{n+1}$

6.4.3 メモリ・アドレスレジスタ

最後に必要になるモジュールは演算で利用する認証情報 A_n と秘匿情報 M_n を格納しておくためのメモリ、そしてメモリのデータにアクセスするために必要なアドレスを格納しておくためのアドレスレジスタである。設計したメモリを図 6.6 と表 6.6 に、アドレスレジスタを図 6.7 と表 6.7 に示す。

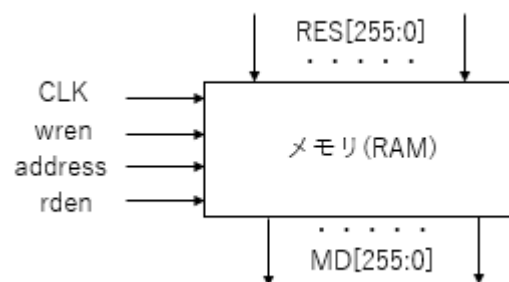


図 6.6 メモリ

表 6.6 メモリの入出力

信号名	I/O	備考
CLK	in	システムクロック
wren	in	書き込み有効化信号
address	in	アドレス ($A_n : 0$ 、 $M_n : 1$)
rden	in	読み出し有効化信号
RES	in	更新データ (256bit)
MD	out	演算器への出力 (256bit)

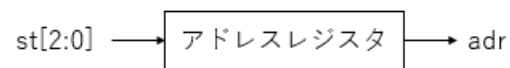


図 6.7 アドレスレジスタ

表 6.7 アドレスレジスタの入出力

信号名	I/O	備考
st	in	状態制御信号 (3bit)
adr	out	アドレス ($A_n : 0$ 、 $M_n : 1$)

アドレスレジスタは状態制御信号を受け取り、それに対応したアドレスを出力する。以下の表 6.8 に状態制御信号とアドレスの対応を示す。

表 6.8 ハードウェアの状態とアドレスの対応

状態制御信号	出力するアドレス
000(S0)	1
001(S1)	0
010(S2)	0
011(S3)	0
100(S4)	1
101(S5)	0
110(S6)	0
111(S7)	0

6.5 モジュール間の接続

各モジュールを実装するために接続し、完成させた SAS 認証回路のモジュール図を図 6.8 に示す。

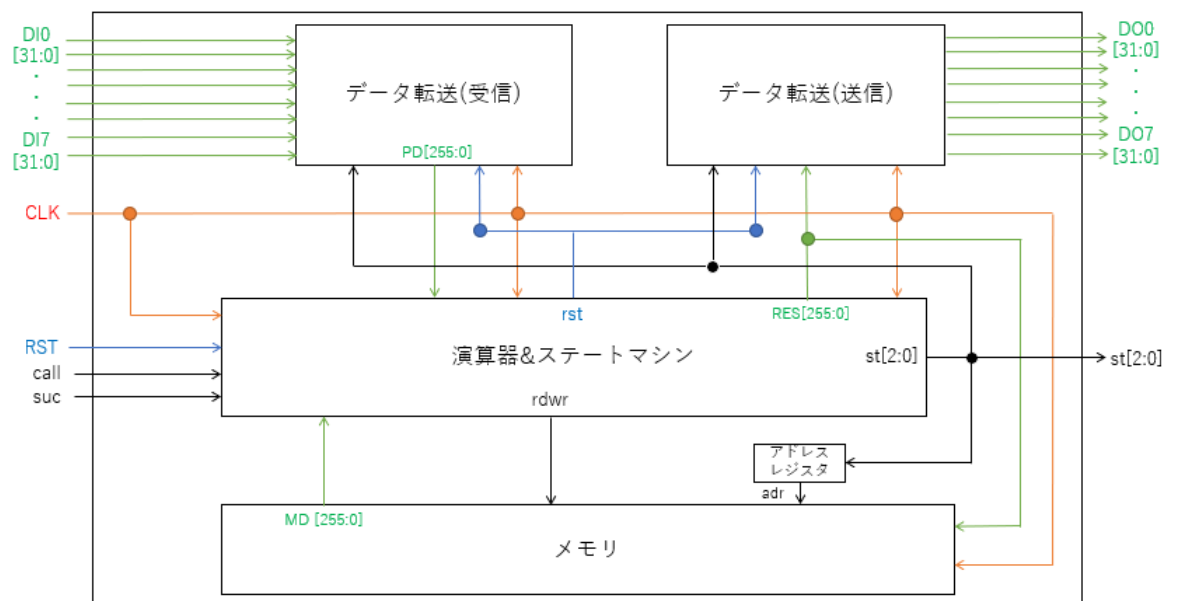


図 6.8 SAS 認証回路のモジュール図

第7章 実装

本章では本研究で実装した対象と実装後の結果を示す。

7.1 検証方法・実装対象

今回設計した SAS 認証回路が正しく動作するかを検証するにあたって FPGA への実装を行う前に、コンピュータ上でのシミュレーションを実施し、モジュールごとに正しく動作しているかの検証と、回路全体で正しく動作しているかを検証する。

表 7.1 シミュレーション環境

使用 OS	シミュレーション環境
Windows 10	Modelsim

そして、シミュレーションで SAS 認証回路の動作に問題がないことを確認した後に、設計した SAS 認証回路を FPGA の MAX10 に実装し、この回路を動作させるために簡易的なサーバの処理を今回使用する FPGA である MAX10 で利用可能なエンデベッド・プロセッサである NIOS II に実装する。

表 7.2 実装環境

実装内容	実装環境	開発言語
クライアント側の処理	MAX10	verilog HDL
サーバ側の処理	MAX10 (NIOS II)	C 言語

7.2 シミュレーション結果

シミュレーションを実行する際に、モジュールごとにシミュレーションを行った後に回路全体のシミュレーションを行った。

はじめに、データ受信モジュールについてシミュレーションを行った結果を以下の図 7.1 に示す。

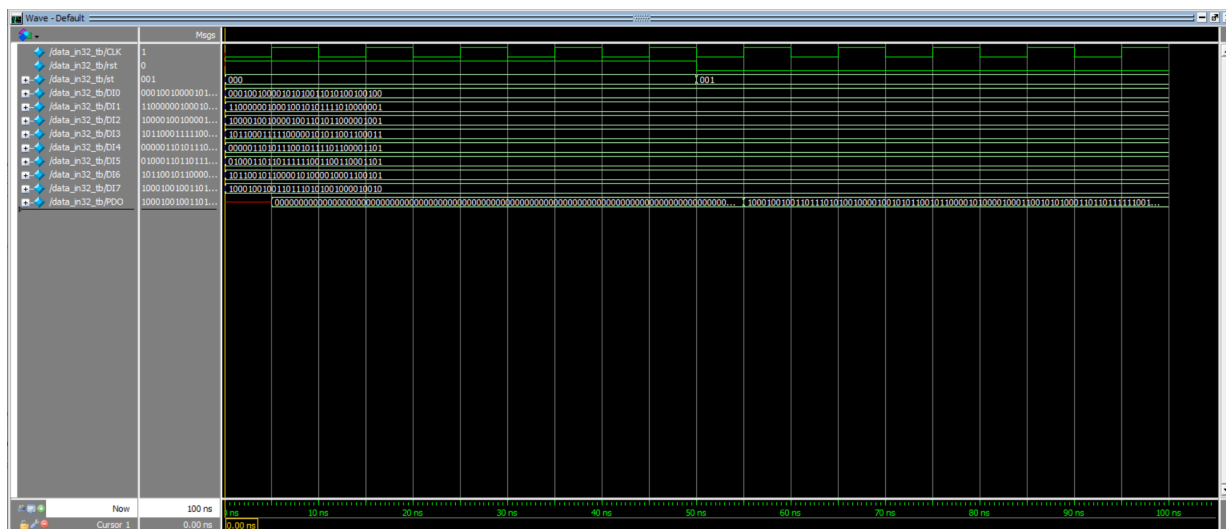


図 7.1 データ受信モジュール波形図

この図 7.1 のシミュレーション結果から各 DI から 32bit の入力を受け取り、状態が S1 に遷移すると 256bit で受け取った入力を出力していることが確認できたので、データ受信モジュールが正しく動作していると言える。

次に、データ送信モジュールについてシミュレーションを行った結果を以下の図 7.2 に示す。

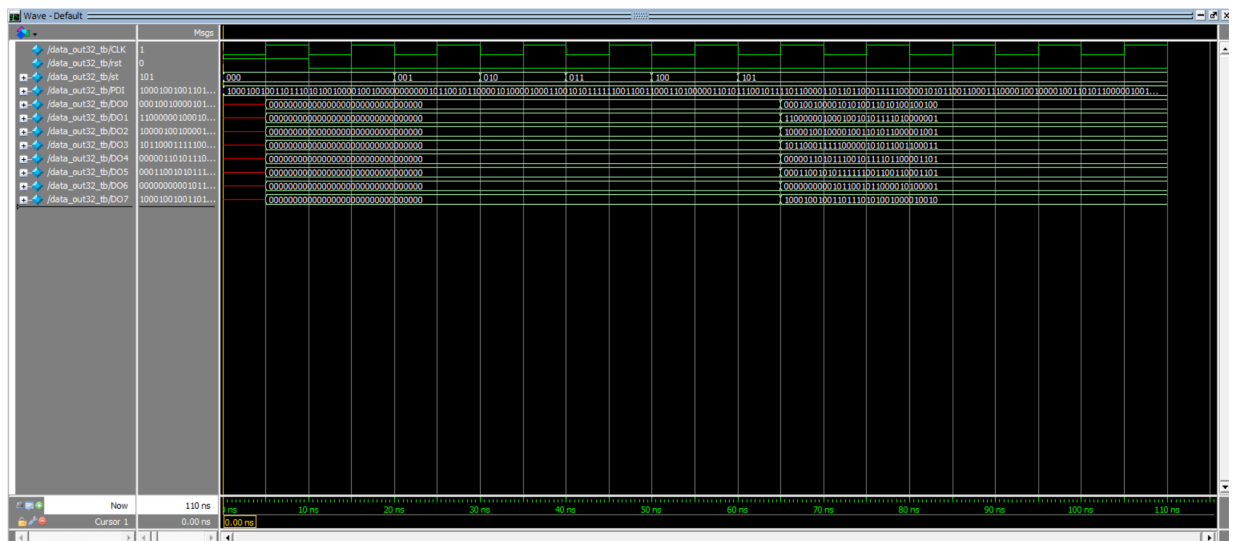


図 7.2 データ送信モジュール波形図

図 7.2 のシミュレーション結果から 256bit の入力を受け取り、状態が S5 に遷移すると各 DO から 32bit の出力がされていることが確認できたので、データ送信モジュールが正しく動作していると言える。

次に、ステートマシンについてシミュレーションを行った結果を以下の図 7.3、図 7.4 に示す。

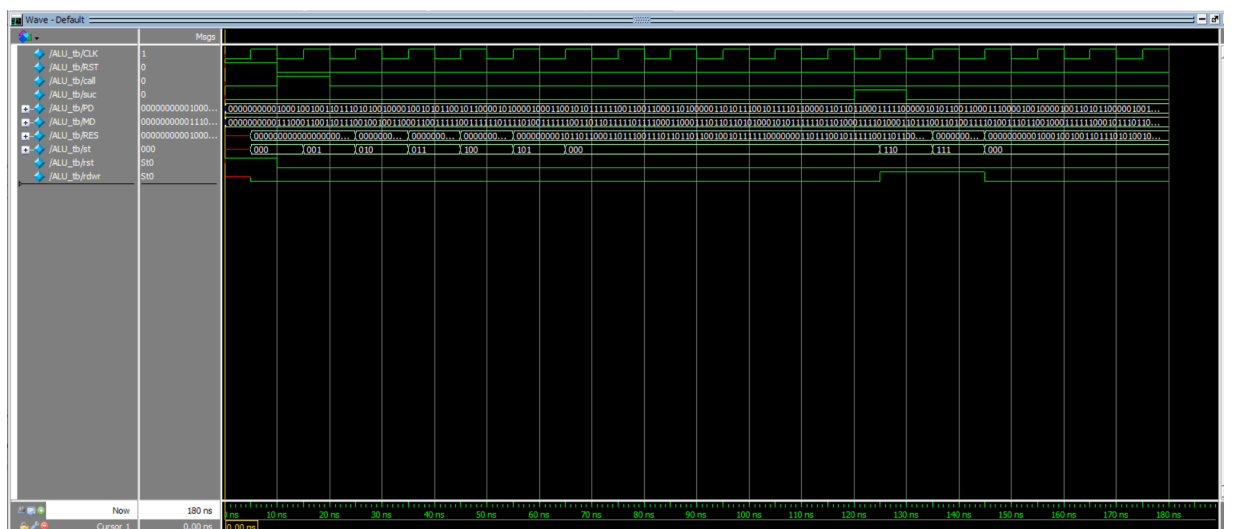


図 7.3 ステートマシン波形図 1

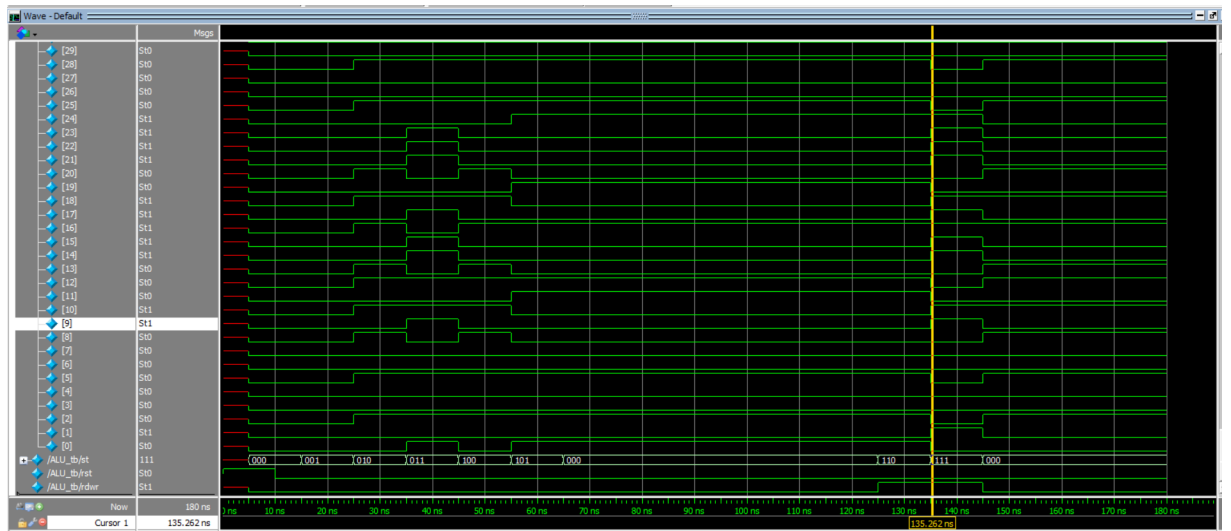


図 7.4 ステートマシン波形図 2

図 7.3 から状態遷移が設計した通り、 $call = 1$ のタイミングで $S0 \rightarrow S1$ と遷移しており、その後も $S1 \rightarrow S2 \rightarrow S3 \rightarrow S4 \rightarrow S5$ と遷移していることが確認できる。そして、 $suc = 1$ のタイミングで $S0 \rightarrow S6$ 遷移し、その後 $S6 \rightarrow S7 \rightarrow S0$ と遷移していることが確認できる。

図 7.4 のシミュレーション結果から正しい演算が行われていることが確認できる。以上 2 つのシミュレーション結果からステートマシンが正しく動作していると言える。

次に、メモリについてシミュレーションを行った結果を以下の図 7.5 に示す。



図 7.5 メモリ波形図

図 7.5 のシミュレーション結果からアドレスによって読み出す値が変化していることと、

書き込みが行われていることが確認できたので、メモリが正しく動作していると言える。

次に、アドレスレジスタについてシミュレーションを行った結果を以下の図 7.6 に示す。



図 7.6 アドレスレジスタ波形図全体

図 7.6 のシミュレーション結果から状態遷移信号が入力されるとそれに対応したアドレスを出力していることが確認できるので、アドレスレジスタが正しく動作していると言える。

最後に SAS 認証回路全体についてシミュレーションを行った結果を以下の図 7.7 に示す。

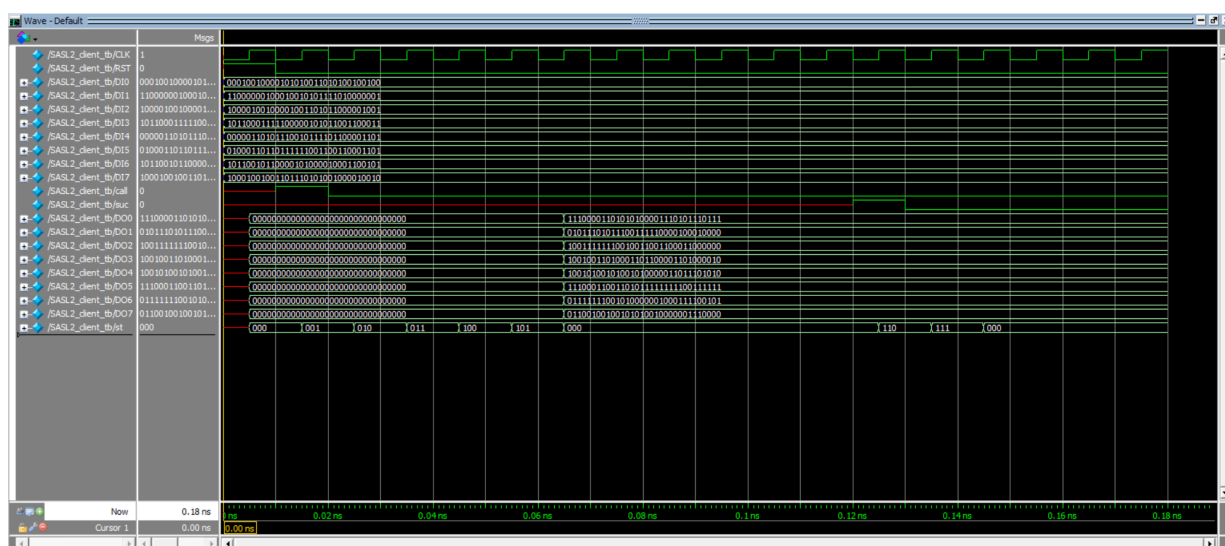


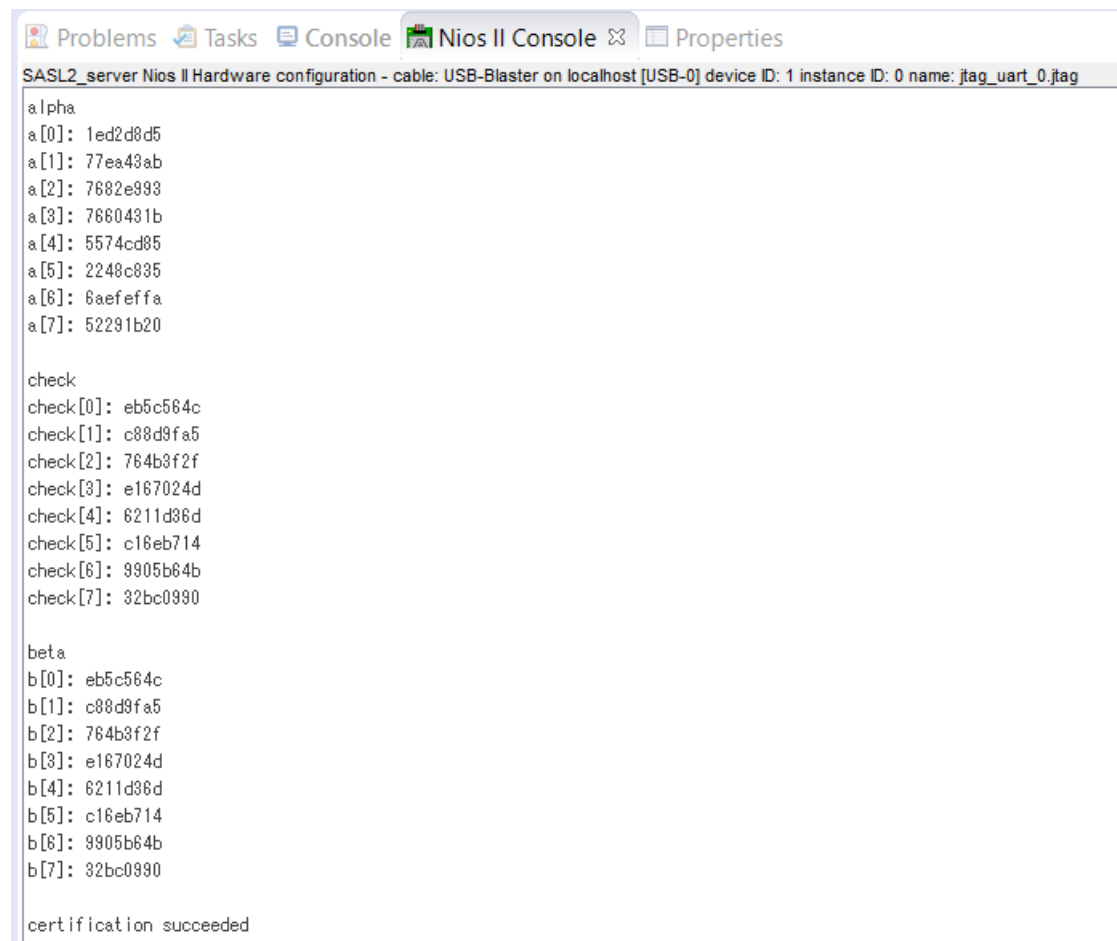
図 7.7 SAS 認証回路の波形図

図 7.7 のシミュレーション結果から各モジュールが正しく接続されており、動作していることが確認できた。

7.3 FPGA における実装結果

シミュレーションの結果からコンピュータ上では設計したハードウェアが正しく動作していることが確認できたので、次は FPGA 上で正しく動作しているかどうかを確認していく。

サーバ側とクライアント側を接続して実行した結果を以下の図 7.8 に示す。



```
Problems Tasks Console Nios II Console Properties
SASL2_server Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtag_uart_0.jtag

alpha
a[0]: 1ed2d8d5
a[1]: 77ea43ab
a[2]: 7682e993
a[3]: 7660431b
a[4]: 5574cd85
a[5]: 2248c835
a[6]: 6aefffa
a[7]: 52291b20

check
check[0]: eb5c564c
check[1]: c88d9fa5
check[2]: 764b3f2f
check[3]: e167024d
check[4]: 6211d36d
check[5]: c16eb714
check[6]: 9905b64b
check[7]: 32bc0990

beta
b[0]: eb5c564c
b[1]: c88d9fa5
b[2]: 764b3f2f
b[3]: e167024d
b[4]: 6211d36d
b[5]: c16eb714
b[6]: 9905b64b
b[7]: 32bc0990

certification succeeded
```

図 7.8 認証成功

図 7.8 から今回設計した SAS 認証回路が FPGA 上でも正しく α を受信して β を生成し、サーバ側へ送信できていると言える。


```

SASL2_server Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtag_uart_0.jtag
a[0]: 1ed2d8d5
a[1]: 77ea43ab
a[2]: 7682e993
a[3]: 7680431b
a[4]: 5574cd85
a[5]: 2248c835
a[6]: 6aefffa
a[7]: 52291b20

check
check[0]: eb5c584c
check[1]: c88d9fa5
check[2]: 764b3f2f
check[3]: e167024d
check[4]: 8211d36d
check[5]: c16eb714
check[6]: 9905b64b
check[7]: 32bc0990

beta
b[0]: 2c9b89ba
b[1]: 37585f7a
b[2]: 71ad3ed0
b[3]: dda905b1
b[4]: e1dbcb6e
b[5]: c572b892
b[6]: a90549c3
b[7]: e8d3c871

certification failed

```

図 7.9 更新成功

SAS-L2 の認証では、毎回認証が終わるとクライアント側とサーバ側に格納されている認証情報を更新する同期処理が必要である。この同期処理によって、盗聴により認証情報が漏洩した場合でも次回認証では使用できないためリプレイアタックを防ぐことができる。

図 7.9 はサーバとクライアント間の同期処理が正しく実行できていない結果を示す。

図 7.8 で認証成功し、 α 、 A_n 、 M_n を用いて再度認証を行った結果であり、認証失敗となった。これは 1 度認証成功した段階でクライアント側で A_n と M_n は更新が行われているが、今回作成したサーバ側の処理では更新を行っていないため 2 回目の認証では認証失敗になっている。

以上の結果から今回設計した SAS 認証回路が FPGA 上でも正しく認証情報 A_n と秘匿情報 M_n の更新が実行できていると言える。

第8章 考察

本章では、本研究で設計した SAS 認証回路を FPGA に実装した結果についての考察を述べる。

本研究では、SAS-L2 のクライアント側の認証処理をハードウェア化して SAS 認証回路を設計、FPGA に実装した結果から一連の SAS-L2 のクライアント側の認証処理がハードウェアを用いて実現できることを確認することができた。

しかし、本研究の目標である JTAG の認証機構として実用化するにあたって現在の設計から改良すべき問題点が2つ挙げられる。

1 つ目は、データ転送である。本研究で設計した SAS 認証回路ではデータ転送を行う際に 256bit のデータをやり取りする際にデータの入出力どちらも 32bit の端子 8 本ずつ用いている。

しかし、実際の JTAG ではデータの入出力は TDI と TDO の 1 本ずつで行われているため、今回設計したデータ転送モジュールの入力と出力の端子を 1 本ずつに削減することが今後の課題となる。

2 つ目は、回路面積における問題である。FPGA では回路規模を表す単位としてロジックセル (LC) の数を利用する。そこで、本研究で設計した SAS 認証回路の規模についてを以下の表 8.1 に示す。

表 8.1 SAS 認証回路の規模

logic cells	LUT-Only LCs	Register-Only LCs	LUT/Register LCs
2570	1029	253	1288

表 8.1 の logic cells は回路で使用されているロジックセルの数を表しており、LUT-Only LCs は LUT(Look Up Table) のみを使用しているロジックセルの数を示している。そして、Register-Only LCs はレジスタのみを使用しているロジックセルの数を示し、LUT/Register

LCs は LUT とレジスタの両方を使用しているロジックセルの数を示している。

本研究で設計した SAS 認証回路では 256bit のデータを取り扱うために 256bit のレジスタを使用している。データ受信モジュールで 1 つ、データ送信モジュールで 1 つ、そしてステートマシンで演算を行うために 2 つ用意してそこにデータを格納し演算を行っている。また、認証情報の更新処理で使用する情報を保持しておくために、演算で使ったレジスタとは別に 256bit のレジスタをさらに 2 つ使用しており、合計で 6 つ使用している。そのため回路規模が増大していると考えられ、レジスタの使用量の削減が今後の課題となる。

これらの問題を解決する方法として 256bit の認証データを 32bit に分割し、8 回の認証処理を行うことで 1 回の認証とする方法が考えられる。

第9章 あとがき

本研究では SAS-L2 のクライアント側の認証処理をハードウェア化した SAS 認証回路を設計し、FPGA である MAX10 における実装を行った。また、サーバ側の処理を MAX10 で利用可能なエンデベッド・プロセッサである NIOS II 上で C 言語を用いて実装した。

その結果、SAS-L2 の認証処理がハードウェアで実現可能であることを確認することができた。

しかし、設計した SAS 認証回路を JTAG の認証機構として実用化するには認証データを JTAG で用いられる信号線で認証データの入出力を行う方法の検討と、SAS-L2 の実装のための回路面積の削減方法の検討が今後の課題として挙げられる。

また、従来法である共通鍵暗号方式を用いた認証方式やチャレンジレスポンス認証に基づいた認証方式との比較を行うことで SAS 認証回路の導入が JTAG の認証機構の軽量化に繋がることを検証する必要があることも今後の課題として挙げられる。

謝辞

本研究を進めるにあたり、懇篤な御指導、御鞭撻を賜りました本学高橋寛教授に深く御礼申し上げます。

本論文を作成するにあたり、詳細なるご検討、貴重な御教示を頂きました本学高橋寛教授ならびに甲斐博准教授、王森レイ講師に深く御礼申し上げます。

また、審査頂いた本学樋上喜信教授ならびに梶原智之助教に深く御礼申し上げます。

最後に、多大な御協力と貴重な御助言を頂いた計算機/ソフトウェアシステム研究室の諸氏に厚く御礼申し上げます。

参考文献

- [1] 王森レイ、亀山修一、高橋寛
“JTAG セキュリティ脅威-攻撃の現状とその対策-”
エレクトロニクス実装学会誌、2021 年
- [2] “JTAG 技術について”
<https://www.xjtag.com/ja/about-jtag/jtag-a-technical-overview/>、(参照 2022-02-03)
- [3] 馬 竣、岡本 悠、王 森レイ、甲斐 博、亀山 修一、高橋 寛、清水 明宏
“JTAG 認証機構の軽量化設計について”
実装学会春季講演大会論文、2022 年
- [4] “組み込みの FPGA とは?仕組み、意味、特徴をわかりやすく解説”
<https://www.kumikomi.jp/fpga/>、(参照 2022-02-07)
- [5] 芹井滋喜
“50K MAX10 搭載! FPGA スタートキット DE10-Lite 入門”
CQ 出版社、2020 年

卒論チェックシート

学籍番号 8535017K 氏名 岡本 悠

目的

卒論本文に関して、以下の項目 1)～5) に関する記述が必要です。5 項目についての記述も卒論評価の 1 部とします。この卒論チェックシートを完成させ、卒論提出前に記入漏れがないことを確認してください。なお、このシートは卒論審査資料の一つとなります。卒論と同様にしっかり完成させ、卒論と一緒に主査と副査へ提出してください。

提出方法

1. チェック項目について明確・簡潔に回答を記入する。また、対応記述を含む本文のページ番号を明記する（例：3 ページ, 3,5,7 ページ, 3-10 ページなど）。全ての項目について回答し、卒論チェックシートを完成させる。
2. 完成した卒論チェックシートを、卒論を収めたファイルの最後尾に綴じる。
3. 主査（1 名）と副査（2 名）に卒論と卒論チェックシートを綴じたファイルを提出する（従って、卒論とともに卒論チェックシートも 3 部用意する、卒論チェックシートの記述内容は 3 部とも同一で良い）。

1) 研究の目的・目標を明確に設定できる。（卒論評価項目 1）

【チェック項目】 研究目的・目標を説明してください。

近年の IC は複雑になっておりテストを容易に行うために JTAG が必要である。一方で、JTAG を起因とする脆弱性があることから、その対策として認証機構を導入する必要がある。しかしながら従来の認証機構はハードウェア的に、計算处理的に負荷が大きいため、低性能な機器への導入は困難である。そこで、認証機構ハードウェアおよび計算処理の軽量化を目指した SAS-L2 を FPGA 上にハードウェアとして実装する。

本文におけるページ番号： 1-2 ページ

2) 人類や社会に望まれ、貢献する研究目標を立てられる。（卒論評価項目 2）

【チェック項目】 論文に示された研究目標が、情報工学を応用し人類・社会に貢献するものであることを説明してください。（社会との関わりなど）

SAS-L2 を JTAG の認証機構として導入し、認証機構の軽量化を実現できれば、IoT デバイスのセキュリティ強化に繋がり、社会で普及している低性能な機器による IoT システムをより安全に使用できるようになる。

本文におけるページ番号： 1-2 ページ

（裏にもあります）

- 3) 研究の目的・目標を実現するための具体的研究方法を示し、実行できる。(卒論評価項目 3)

[チェック項目] 論文に示された研究方法の具体性や、研究目的・研究目標の達成を目指すためにどのような意味がありそのような研究方法を採用したのか説明してください。

本研究では、SAS-L2 を用いて JTAG の認証機構の軽量化方法を示すために、SAS-L2 をハードウェア化した SAS 認証回路を設計し、FPGA に実装することによって、SAS-L2 をハードウェアとして実現できるかどうかを検証した。

本文におけるページ番号： 13-30 ページ

- 4) 研究の内容が、情報工学技術の発展や応用に貢献するものである。(卒論評価項目 4)

[チェック項目] 論文で示された研究内容が、情報工学技術の発達や応用に貢献するものであることを説明してください。(研究内容の新規性など)

SAS-L2 として提案された処理能力の低い機器へセキュリティ機能を実現できる認証方式を JTAG の認証機構に適用し、それをハードウェアで実装できるため、低性能な機器による IoT デバイスのセキュリティの強化が可能となる。このことが IoT 分野の発展に貢献することができる。

本文におけるページ番号： 1-2 ページ

- 5) 卒業論文、卒業論文発表において、卒業研究の目的・目標、研究方法、研究成果が論理的に述べられる。(卒論評価項目 6)

[チェック項目] 論文で示された研究成果について説明してください。

本研究では、JTAG の認証機構の軽量化手法の提案のため SAS-L2 の認証処理をハードウェアで実現することを目的とし、SAS 認証回路を設計し FPGA に実装した。PC 上でのシミュレーションと FPGA 上で動作させることで認証処理が行われていることが確認できた。よって SAS-L2 がハードウェアで実装することができた。

本文におけるページ番号： 13-32 ページ

[チェック項目] 卒業研究の目的・目標、研究方法、研究成果がどのような章立てで述べられているか説明してください。

第 1 章では研究背景・目標・目的を述べる。第 2 章では JTAG についての説明を述べる。第 3 章では FPGA についての説明を述べる。第 4 章では SAS-L2 の説明を述べる。第 5 章では SAS-L2 を用いた JTAG プロトコルについて説明を述べる。第 6 章では SAS 認証回路の設計について示す。第 7 章ではシミュレーション結果と FPGA に実装した結果を示す。第 8 章では実装した SAS 認証回路についての考察を示す。第 9 章では本研究のまとめを行う。

以上