Dissecando e aplicando SHA-1 em Hardware Reconfigurável

Eduardo Barros Santos
@edusantos33

Agenda

- Motivação
- SHA-1
- FPGA
- Implementação
- Resultados

About me

- Analista de TI - Suporte e Redes
- Tecnologia em Redes de Computadores
- Especialização em Segurança de Redes
- Mestrando e Pesquisador do Grupo de Pesquisa em Sistemas Embarcados e Hardware Reconfigurável (GPSEHR) - DCA - UFRN
  - Grupo vem trabalhando a mais de cinco anos em soluções associada a aceleração de algoritmos em várias áreas através do desenvolvimento de hardwares dedicados em FPGA.
- Técnico em Eletrônica
- ex LFCS e LFCE

MOTIVAÇÃO

MOTIVAÇÃO

AUG 31, 2016 @ 08:21 AM 16,248

The Little Black Book of Billionaire Secrets

Why You Shouldn't Panic About Dropbox Leaking 68 Million Passwords

Thomas Fox-Brewster, FORBES STAFF

I cover crime, privacy and security in digital and physical forms. FULL BIO

Fonte: https://www.forbes.com/sites/thomasbrewster/2016/08/31/dropbox-hacked-but-its-not-that-bad/#2308b4495576

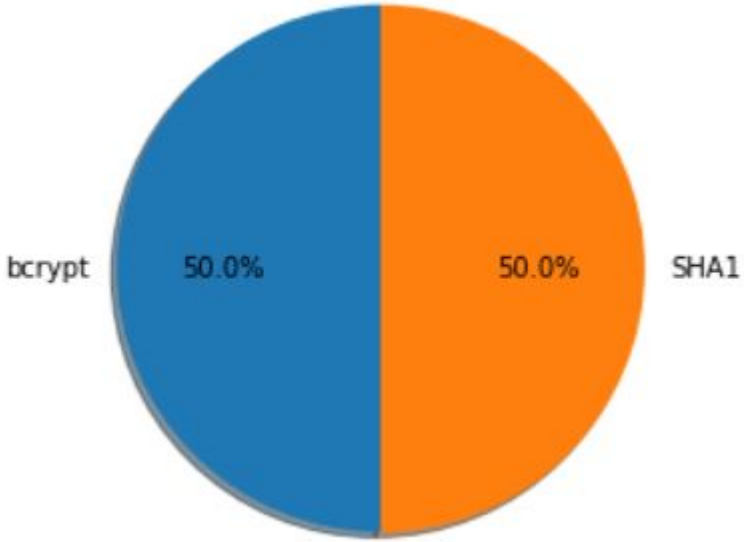
MOTIVAÇÃO

AUG 31, 2016 @ 08:21 AM 16,248

The Little Black Book of Billionaire Secrets

Why You Shouldn't Panic About Dropbox Leaking 68 Million Passwords

Thomas Fox-Brewster, FORBES STAFF
I cover crime, privacy and security in digital and physical forms. FULL BIO



Fonte: https://www.forbes.com/sites/thomasbrewster/2016/08/31/dropbox-hacked-but-its-not-that-bad/#2308b4495576

MOTIVAÇÃO

The Register®
Biting the hand that feeds IT

DATA CENTRE SOFTWARE **SECURITY** TRANSFORMATION DEVOPS BUSINESS PERSONAL TECH

Security

LinkedIn mass hack reveals ... yup, you're all still crap at passwords

'Linkedin'? 'P4ssw0rd'? '123456'? Come on, people

By John Leyden 24 May 2016 at 08:26 82 SHARE ▼

Analysis of passwords from the LinkedIn leak has revealed, should there be any doubt, that users remain terrible at choosing secure login credentials.

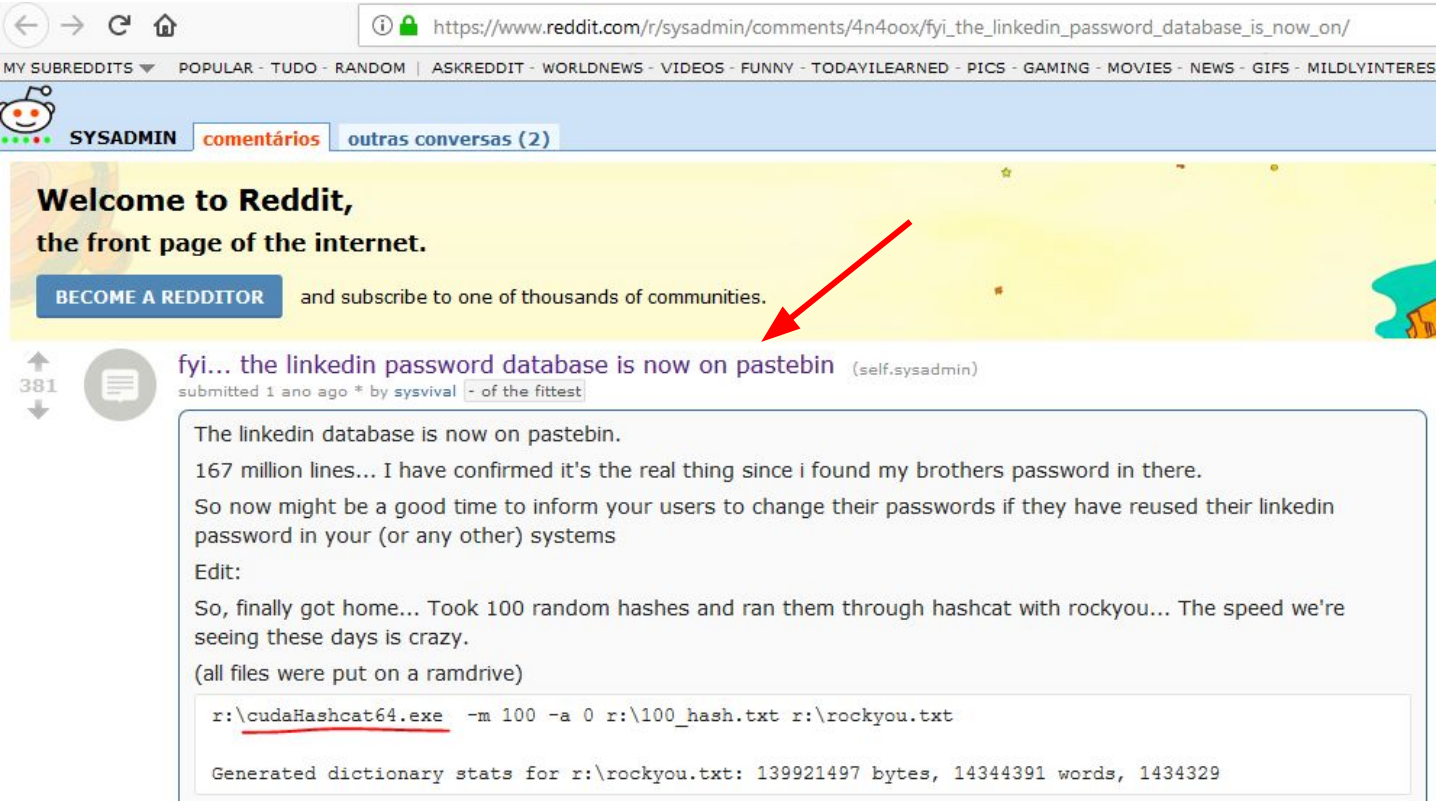
Last week a black hat hacker using the nickname Peace was revealed as attempting to sell 117 million LinkedIn users' emails and passwords on the dark web.

Fonte: https://www.theregister.co.uk/2016/05/24/linkedin_password_leak_hack_crack/

MOTIVAÇÃO

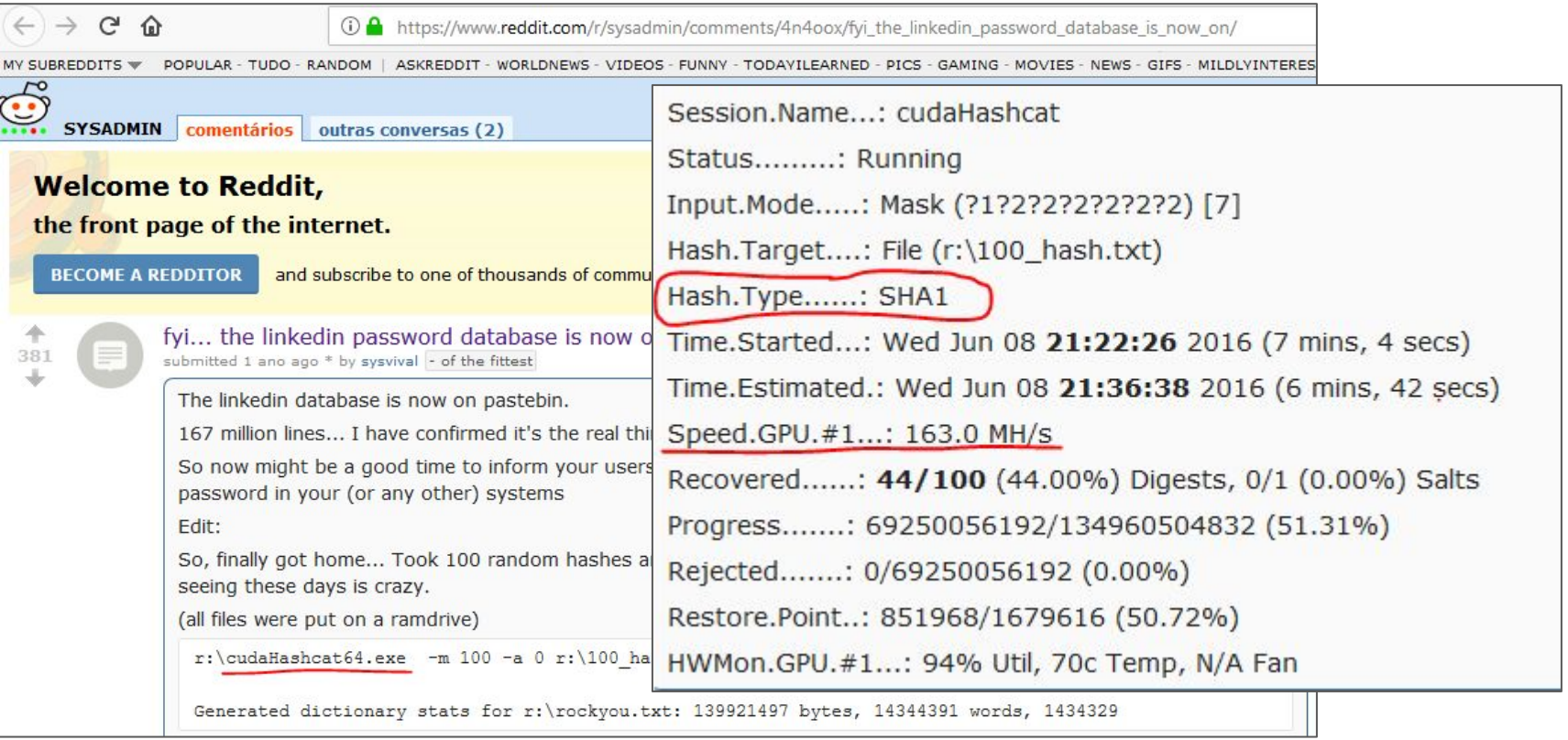
Fonte: https://www.theregister.co.uk/2016/05/24/linkedin_password_leak_hack_crack/

MOTIVAÇÃO



Fonte: https://www.reddit.com/r/sysadmin/comments/4n4oox/fyi_the_linkedin_password_database_is_now_on/

MOTIVAÇÃO



Fonte: https://www.reddit.com/r/sysadmin/comments/4n4oox/fyi_the_linkedin_password_database_is_now_on/

MOTIVAÇÃO

Dating the ginormous MySpace breach

01 JUNE 2016

It's been a crazy time for data breaches and as I wrote yesterday, we've seen a very distinct pattern of historical mega breaches lately. Fling in 2011, LinkedIn in 2012, tumblr in 2013 and the mother of them all, MySpace in, well, we don't quite know. There's been no information forthcoming from anyone about when this breach actually occurred and there's no explicit indicators in the data dump either (sometimes there are timestamps on account creation or website activity). So when did it actually happen? Let's work out.

Firstly, the only data in the breach is an incrementing ID (possibly an internal MySpace identifier which would enable *them* to date it), an email address, username and one or two passwords. The passwords are stored as SHA1 hashes of the first 10 characters of the password converted to lowercase. That's right, truncated and case insensitive passwords stored without a salt. There are likely some interesting insights to take away from the passwords alone, but it's the email addresses that can help us actually date the thing.

Fonte: https://www.troyhunt.com/dating-the-ginormous-myspace-breach/

MOTIVAÇÃO

Fonte: https://motherboard.vice.com/en_us/article/qkj3n7/myspace-data-breach-427-million-passwords-available-online

MOTIVAÇÃO

The Register®
Biting the hand that feeds IT

CENTRE SOFTWARE SECURITY TRANSFORMATION DEVOPS **BUSINESS** PERSONAL TECH

Business

MySpace 'passwords dump'

By Shaun Nichols in San Francisco 27 May 2016 at 23:55 8 SHARE ▼

Zillions of usernames and SHA-1 hashed passwords for accounts on social networking relic MySpace have been swiped from the site's databases and leaked online, apparently.

Lorenzo Franceschi-Bicchierai
Jun 29 2016, 12:35pm

Fonte: https://www.theregister.co.uk/2016/05/27/myspace_hacked_passwords_pulled/

MOTIVAÇÃO

ANDY GREENBERG SECURITY 09.22.16 12:15 PM

HACK BRIEF: YAHOO BREACH HITS HALF A *BILLION* USERS

Fonte: https://www.wired.com/2016/09/hack-brief-yahoo-looks-set-confirm-big-old-data-breach/

MOTIVAÇÃO

DEC 14, 2016 @ 0

Billionaire Secrets

Yaho

UPD

Yahoo just admitted to another astonishingly big breach, surpassing its previous ignominious record where data on 500 million accounts was stolen, revealing a whopping 1 billion hit in 2013.

The two breaches were entirely separate, the bigger of the two in August 2013, the other a year later, Yahoo confirmed in a Tumblr post today. The data "may have included names, email addresses, telephone numbers, dates of birth, hashed passwords (using MD5) and, in some cases, encrypted or unencrypted security questions and answers," explained chief information security officer Bob Lord. No plaintext passwords were leaked, nor were payment card data or bank account information.

Fonte: https://www.forbes.com/sites/thomasbrewster/2016/12/14/yahoo-admits-another-billion-user-accounts-were-leaked-in-2013/#6abf0dded5c2

MOTIVAÇÃO

Fonte: https://nakedsecurity.sophos.com/2016/12/15/yahoo-breach-ive-closed-my-account-because-it-uses-md5-to-hash-my-password/

HASH?!

HASH?!

- Algoritmo utilizado para verificar a integridade de sequências de dados
- Uma função hash possui como saída um código de comprimento fixo C.

HASH?!

- A saída da função hash, chamada de código hash, é uma assinatura da mensagem de entrada (fingerprint).

HASH?!

- A saída da função hash, chamada de código hash, é uma assinatura da mensagem de entrada (fingerprint).

=

```
79054025
255fb1a2
6e4bc422
aef54eb4
```

Fonte da figura: https://blog.codinghorror.com/content/images/uploads/2012/04/6a0120a85dcdae970b016303bd99fb970d-800wi.png

SHA-1

- Versão revisada do SHA-0
- Substituto do algoritmo MD5 em 1995 pelo NIST
- Publicado como um padrão de processamento de informações federais (FIPS) de número 180-1 e RFC 3174

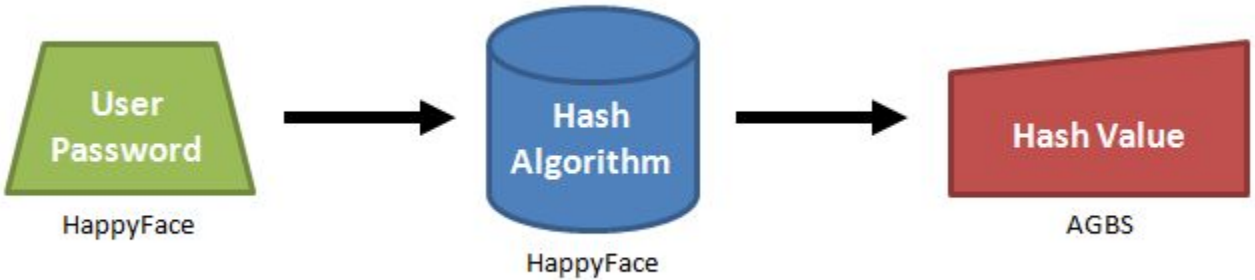
SHA-1

- A função hash SHA-1 foi selecionada para o Algoritmo de Assinatura Digital (DSA) - padronizada pela FIPS 186-4
- Verifica sequência de dados associados a mensagens
    - protocolos de comunicação
    - arquivos
    - armazenamento de senhas
- Usada em certificados digitais antes de ser substituída pelo SHA-2.

SHA-1

- O SHA-1 gera códigos hash de C=160 bits para qualquer tamanho de K
- Big Data
- Internet das coisas (Internet of Things - IoT)
- Velocidade
- Consumo de energia

Hash overview



Fonte:
https://static1.squarespace.com/static/52b5f43ee4b02301e647b446/t/53f3917ae4b04507993d9f00/1408471419525/How+Microsoft+Excel+2010+%26+2007+Password+Protection+Algorithm+Works

O Algoritmo SHA-1

- Para cada i-ésima mensagem de entrada, **m**i, (de comprimento **K**i bits), expressa como

$$\mathbf{m}_i = \left[\begin{array}{cccc} m_0 & m_1 & \dots & m_{K_i-1} \end{array} \right] \text{ onde } m_k \in \{0,1\} \forall\, k,$$

O Algoritmo SHA-1

- O algoritmo SHA-1 gera uma mensagem de saída, hi, chamada de código hash, de tamanho fixo C = 160 bits, caracterizado como

$$\mathrm{h}_i = \left[\begin{array}{cccc} h_0 & h_1 & \ldots & h_{C-1} \end{array} \right] \text{ onde } h_k \in \{0,1\} \, \forall \, k.$$

Pseudo-código

Algoritmo 1 SHA-1 para cada i-ésima mensagem $\mathbf{W}_i$

1: $\mathbf{z}_i \leftarrow [\mathbf{m}_i]$
2: $\mathbf{p}_i \leftarrow \mathit{GeraçãoPreenchimento}(K_i)$
3: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i]$
4: $\mathbf{v}_i \leftarrow \mathit{GeraçãoComprimento}(K_i)$
5: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i\, \mathbf{v}_i]$
6: $\mathbf{h_i} \leftarrow \mathit{InicializaçãoHash}(\,)$
7: **para** $j \leftarrow 0$ **até** $L_i - 1$ **faça**
8: $\quad \mathbf{b}_j \leftarrow \mathit{DivisãoMensagem}(\mathbf{z}_i)$
9: $\quad n \leftarrow -1$
10: $\quad \mathbf{H}(n) \leftarrow \mathit{InicializaVariáveisHash}(\,)$
11: $\quad$ **para** $n \leftarrow 0$ **até** 79 **faça**
12: $\quad\quad \mathbf{w}(n) \leftarrow \mathit{CálculoFunçãoW}(n, \mathbf{b}_j)$
13: $\quad\quad \mathbf{f}(n) \leftarrow \mathit{CálculoFunçãoF}(n, \mathbf{B}(n), \mathbf{C}(n), \mathbf{D}(n))$
14: $\quad\quad \mathbf{H}(n) \leftarrow \mathit{AtualizaVariáveisHash}(\mathbf{H}(n))$
15: $\quad$ **fim para**
16: $\quad \mathbf{h}_i \leftarrow \mathit{AtualizaHash}(\mathbf{H}(n))$
17: **fim para**

Inserção do Preenchimento

- tem como função deixar o comprimento da i-ésima mensagem, mi, divisível por M = 512
- Padding - formada por uma palavra binária de Pi bits no qual o bit mais significativo e 1 e o resto dos bits são 0.

$$\mathbf{p}_i = \left[\begin{array}{cccc} p_0 & p_1 & \cdots & p_{P_i-1} \end{array} \right],$$

onde, $p_0 = 1$ e $p_i = 0$ para $i = 1 \ldots P_i - 1$.

Inserção do Comprimento

- é adicionada a mensagem **v**i caracterizada por uma palavra binária de **T = 64 bits** e expressa como

$$\mathbf{v}_i = \left[\begin{array}{cccc} v_0 & v_1 & \ldots & v_{T-1} \end{array} \right] \text{ onde } v_k \in \{0,1\} \, \forall \, k.$$

Inserção do Comprimento

A mensagem vi armazena o valor do comprimento da i-esima mensagem de entrada mi, ou seja,

$$\mathbf{v}_i = Binary(K, T)$$

- onde Binary(a; b) e uma função que retorna um vetor de tamanho b com a representação binária de um número decimal qualquer a em b bits no padrão big-endian.

Exemplo

M1 = abcde

M1 (base16) = 61 62 63 64 65

M1 (base2) = 01100001 01100010 01100011 01100100 01100101

Padding => 01100001 01100010 01100011 01100100 01100101 **1**

M1 + Pi =

61626364 65**8**00000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000

Exemplo

len(M1) = 40 bits

Dec2Hex(40) = 28

M1+Pi+vi =

616263646580000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000
000000000000000000000000000028

Algoritmo 1 SHA-1 para cada i-ésima mensagem $\mathbf{W}_i$

1: $\mathbf{z}_i \leftarrow [\mathbf{m}_i]$
2: $\mathbf{p}_i \leftarrow \mathit{GeraçãoPreenchimento}(K_i)$
3: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i]$
4: $\mathbf{v}_i \leftarrow \mathit{GeraçãoComprimento}(K_i)$
5: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i\, \mathbf{v}_i]$
6: $\mathbf{h_i} \leftarrow \mathit{InicializaçãoHash}(\,)$
7: **para** $j \leftarrow 0$ **até** $L_i - 1$ **faça**
8: $\quad \mathbf{b}_j \leftarrow \mathit{DivisãoMensagem}(\mathbf{z}_i)$
9: $\quad n \leftarrow -1$
10: $\quad \mathbf{H}(n) \leftarrow \mathit{InicializaVariáveisHash}(\,)$
11: $\quad$ **para** $n \leftarrow 0$ **até** 79 **faça**
12: $\quad\quad \mathbf{w}(n) \leftarrow \mathit{CálculoFunçãoW}(n, \mathbf{b}_j)$
13: $\quad\quad \mathbf{f}(n) \leftarrow \mathit{CálculoFunçãoF}(n, \mathbf{B}(n), \mathbf{C}(n), \mathbf{D}(n))$
14: $\quad\quad \mathbf{H}(n) \leftarrow \mathit{AtualizaVariáveisHash}(\mathbf{H}(n))$
15: $\quad$ **fim para**
16: $\quad \mathbf{h}_i \leftarrow \mathit{AtualizaHash}(\mathbf{H}(n))$
17: **fim para**

Inicialização do Código Hash

Padronizada pela FIPS 180-4 [NIST 2015]

$$\mathbf{ha} = \left[\ h_0 \quad \ldots \quad h_{31}\ \right] = Binary(1732584193, 32),$$
$$\mathbf{hb} = \left[\ h_{32} \quad \ldots \quad h_{63}\ \right] = Binary(4023233417, 32),$$
$$\mathbf{hc} = \left[\ h_{64} \quad \ldots \quad h_{95}\ \right] = Binary(2562383102, 32),$$
$$\mathbf{hd} = \left[\ h_{96} \quad \ldots \quad h_{127}\ \right] = Binary(0271733878, 32),$$

e

$$\mathbf{he} = \left[\ h_{128} \quad \ldots \quad h_{159}\ \right] = Binary(3285377520, 32),$$

onde

$$\mathbf{h}_i = \left[\ \mathbf{ha} \quad \mathbf{hb} \quad \mathbf{hc} \quad \mathbf{hd} \quad \mathbf{he}\ \right].$$

Algoritmo 1 SHA-1 para cada i-ésima mensagem $\mathbf{W}_i$

1: $\mathbf{z}_i \leftarrow [\mathbf{m}_i]$
2: $\mathbf{p}_i \leftarrow \mathit{GeraçãoPreenchimento}(K_i)$
3: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i]$
4: $\mathbf{v}_i \leftarrow \mathit{GeraçãoComprimento}(K_i)$
5: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i\, \mathbf{v}_i]$
6: $\mathbf{h_i} \leftarrow \mathit{InicializaçãoHash}(\,)$
7: **para** $j \leftarrow 0$ **até** $L_i - 1$ **faça**
8: $\quad \mathbf{b}_j \leftarrow \mathit{DivisãoMensagem}(\mathbf{z}_i)$
9: $\quad n \leftarrow -1$
10: $\quad \mathbf{H}(n) \leftarrow \mathit{InicializaVariáveisHash}(\,)$
11: $\quad$ **para** $n \leftarrow 0$ **até** 79 **faça**
12: $\quad\quad \mathbf{w}(n) \leftarrow \mathit{CálculoFunçãoW}(n, \mathbf{b}_j)$
13: $\quad\quad \mathbf{f}(n) \leftarrow \mathit{CálculoFunçãoF}(n, \mathbf{B}(n), \mathbf{C}(n), \mathbf{D}(n))$
14: $\quad\quad \mathbf{H}(n) \leftarrow \mathit{AtualizaVariáveisHash}(\mathbf{H}(n))$
15: $\quad$ **fim para**
16: $\quad \mathbf{h}_i \leftarrow \mathit{AtualizaHash}(\mathbf{H}(n))$
17: **fim para**

Divisão da Mensagem

bj = M1+Pi+vi

$$\mathbf{b}_j = \left[\begin{array}{cccc} \mathbf{u}_j[0] & \mathbf{u}_j[1] & \dots & \mathbf{u}_j[15] \end{array} \right],$$

onde ui [k] é uma mensagem de 32 bits, ou seja

$$\mathbf{u}_j[k] = \left[\begin{array}{cccc} u_j[k,0] & u_j[k,1] & \dots & b_j[k,31] \end{array} \right]$$

onde $u_j[k,l] \in \{0,1\} \, \forall \, l.$

Divisão da Mensagem

- 6162636465**8**00000000000000000000000000000000000
0000000000000000000000000000000000000000000000
0000000000000000000000000000000000**28**

Uj[0] = 61626364

Uj[1] = 65**8****00000**

...

Uj[15] = 00000028

Algoritmo 1 SHA-1 para cada i-ésima mensagem $\mathbf{W}_i$

1: $\mathbf{z}_i \leftarrow [\mathbf{m}_i]$
2: $\mathbf{p}_i \leftarrow \mathit{GeraçãoPreenchimento}(K_i)$
3: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i]$
4: $\mathbf{v}_i \leftarrow \mathit{GeraçãoComprimento}(K_i)$
5: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i\, \mathbf{v}_i]$
6: $\mathbf{h_i} \leftarrow \mathit{InicializaçãoHash}(\,)$
7: **para** $j \leftarrow 0$ **até** $L_i - 1$ **faça**
8: $\quad \mathbf{b}_j \leftarrow \mathit{DivisãoMensagem}(\mathbf{z}_i)$
9: $\quad n \leftarrow -1$
10: $\quad \mathbf{H}(n) \leftarrow \mathit{InicializaVariáveisHash}(\,)$
11: $\quad$ **para** $n \leftarrow 0$ **até** 79 **faça**
12: $\quad\quad \mathbf{w}(n) \leftarrow \mathit{CálculoFunçãoW}(n, \mathbf{b}_j)$
13: $\quad\quad \mathbf{f}(n) \leftarrow \mathit{CálculoFunçãoF}(n, \mathbf{B}(n), \mathbf{C}(n), \mathbf{D}(n))$
14: $\quad\quad \mathbf{H}(n) \leftarrow \mathit{AtualizaVariáveisHash}(\mathbf{H}(n))$
15: $\quad$ **fim para**
16: $\quad \mathbf{h}_i \leftarrow \mathit{AtualizaHash}(\mathbf{H}(n))$
17: **fim para**

Inicialização das Variáveis Hash H(n)

- O algoritmo SHA-1 possui cinco variáveis de 32 bits, chamadas de A(n), B(n), C(n), D(n) e E(n) que se atualizam durante as iterações do algoritmo.

- A junção destas cinco variáveis compõem um vetor de 160 posições caracterizado como

$$\mathrm{H}(n) = \left[\begin{array}{ccccc} \mathrm{A}(n) & \mathrm{B}(n) & \mathrm{C}(n) & \mathrm{D}(n) & \mathrm{E}(n) \end{array} \right].$$

Inicialização das Variáveis Hash H(n)

- A inicialização destas variáveis, no instante n = 1, de acordo com a FIPS 180-4 [NIST 2015] ocorre com o recebimento dos mesmos valores que iniciam o hash hi, logo

A(-1) = ha, B(-1) = hb,

C(-1) = hc, D(-1) = hd

E(-1) = he.

Algoritmo 1 SHA-1 para cada i-ésima mensagem $\mathbf{W}_i$

1: $\mathbf{z}_i \leftarrow [\mathbf{m}_i]$
2: $\mathbf{p}_i \leftarrow \mathit{GeraçãoPreenchimento}(K_i)$
3: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i]$
4: $\mathbf{v}_i \leftarrow \mathit{GeraçãoComprimento}(K_i)$
5: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i\, \mathbf{v}_i]$
6: $\mathbf{h_i} \leftarrow \mathit{InicializaçãoHash}(\,)$
7: **para** $j \leftarrow 0$ **até** $L_i - 1$ **faça**
8: $\quad \mathbf{b}_j \leftarrow \mathit{DivisãoMensagem}(\mathbf{z}_i)$
9: $\quad n \leftarrow -1$
10: $\quad \mathbf{H}(n) \leftarrow \mathit{InicializaVariáveisHash}(\,)$
11: $\quad$ **para** $n \leftarrow 0$ **até** 79 **faça**
12: $\quad\quad \mathbf{w}(n) \leftarrow \mathit{CálculoFunçãoW}(n, \mathbf{b}_j)$
13: $\quad\quad \mathbf{f}(n) \leftarrow \mathit{CálculoFunçãoF}(n, \mathbf{B}(n), \mathbf{C}(n), \mathbf{D}(n))$
14: $\quad\quad \mathbf{H}(n) \leftarrow \mathit{AtualizaVariáveisHash}(\mathbf{H}(n))$
15: $\quad$ **fim para**
16: $\quad \mathbf{h}_i \leftarrow \mathit{AtualizaHash}(\mathbf{H}(n))$
17: **fim para**

Cálculo da variável w(n)

- No SHA-1 são necessárias 80 iterações para uma saída válida.
  - Para cada iteração calculada uma variável, w(n):

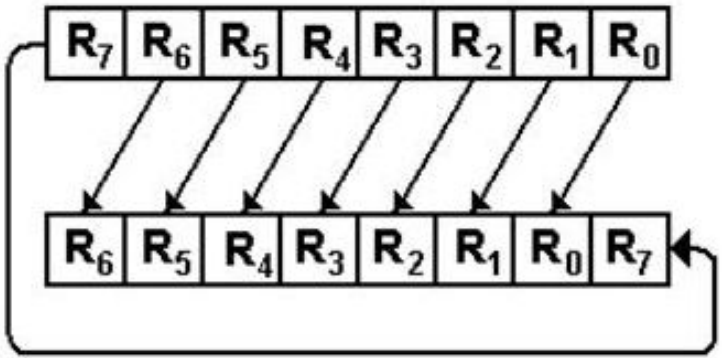
$$\mathbf{w}(n) = \begin{cases} \mathbf{u}_j[n] & \text{para } 0 \leq n \leq 15 \\ \mathbf{sw}[n] & \text{para } 16 \leq n \leq 79 \end{cases},$$

onde

$$\mathbf{sw}[n] = \mathrm{lr}\left(\mathbf{u}_j[n-3] \oplus \mathbf{u}_j[n-8] \oplus \mathbf{u}_j[n-14] \oplus \mathbf{u}_j[n-16], 1\right)$$

$$\mathrm{lr}(\mathbf{r}, \mathbf{s}) = (\mathbf{r} \ll \mathbf{s}) \vee (\mathbf{r} \gg (32 - \mathbf{s})),$$

Leftrotate



Cálculo da Função f(.)

- Em cada n-ésima iteração de cada j-ésimo bloco, bj(n) e calculada uma função não linear, f(.), a partir das informações das variáveis hash B(n), C(n) e D(n). A saída da função, f(.) e armazenada no vetor f (n)

$$\mathrm{f}(n) = f(n, \mathrm{B}, \mathrm{C}, \mathrm{D}) = \begin{cases} \alpha(n) \text{ para } n = 0 \ldots 19 \\ \beta(n) \text{ para } n = 20 \ldots 39 \\ \gamma(n) \text{ para } n = 40 \ldots 59 \\ \delta(n) \text{ para } n = 60 \ldots 79 \end{cases},$$

Cálculo da Função f(.)

onde

$$\alpha(n) = (\mathrm{B}(n-1) \wedge \mathrm{C}(n-1)) \vee (\neg \mathrm{B}(n-1) \wedge \mathrm{D}(n-1)),$$

$$\beta(n) = \mathrm{B}(n-1) \oplus \mathrm{C}(n-1) \oplus \mathrm{D}(n-1),$$

$$\gamma(n) = (\mathrm{B}(n-1) \wedge \mathrm{C}(n-1)) \vee (\mathrm{B}(n-1) \wedge \mathrm{D}(n-1)) \vee (C(n-1) \wedge D(n-1))$$

e

$$\delta(n) = \mathrm{B}(n-1) \oplus \mathrm{C}(n-1) \oplus \mathrm{D}(n-1),$$

onde $\neg$ e $\wedge$ são operações de negação e E bit a bit, respectivamente.

Atualização das Variáveis Hash

- O SHA-1 possui quatro constantes k(n) de 32 bits, as quais são usadas na n-ésima iteração de cada j-ésimo bloco bj(n), conforme especificado por

$$K(n) = \begin{cases} 1518500249 & \text{para } n = 0 \ldots 19 \\ 1859775393 & \text{para } n = 20 \ldots 39 \\ 2400959708 & \text{para } n = 40 \ldots 59 \\ 3395469782 & \text{para } n = 60 \ldots 79 \end{cases}.$$

Atualização das Variáveis Hash

$$\mathrm{E}(n) = \mathrm{D}(n-1),$$

$$\mathrm{D}(n) = \mathrm{C}(n-1),$$

$$\mathrm{C}(n) = \mathrm{lr}(\mathrm{B}(n-1), 30),$$

$$\mathrm{B}(n) = \mathrm{A}(n-1)$$

e

$$\mathrm{A}(n) = \mathrm{V}(n) + \mathrm{Z}(n) + \mathrm{lr}(A(n-1), 5),$$

no qual,

$$\mathrm{Z}(n) = \mathrm{W}(n) + \mathrm{E}(n-1)$$

e

$$\mathrm{V}(n) = \mathrm{f}(n) + \mathrm{k}(n).$$

Atualização do código hash

$$\mathbf{ha} = \mathbf{ha} + \mathrm{A}(79),$$

$$\mathbf{hb} = \mathbf{hb} + \mathrm{B}(79),$$

$$\mathbf{hc} = \mathbf{hc} + \mathrm{C}(79),$$

$$\mathbf{hd} = \mathbf{hd} + \mathrm{D}(79),$$

$$\mathbf{he} = \mathbf{he} + \mathrm{E}(79).$$

Atualização do código hash

03DE6C57

0BFE24BF

C328CCD7

CA46B76E

ADAF4334

$$\mathrm{ha} = \mathrm{ha} + \mathrm{A}(79),$$

$$\mathrm{hb} = \mathrm{hb} + \mathrm{B}(79),$$

$$\mathrm{hc} = \mathrm{hc} + \mathrm{B}(79),$$

$$\mathrm{hd} = \mathrm{hd} + \mathrm{D}(79),$$

$$\mathrm{he} = \mathrm{he} + \mathrm{E}(79).$$

$$\mathrm{h}_i = \left[\begin{array}{ccccc} \mathrm{ha} & \mathrm{hb} & \mathrm{hc} & \mathrm{hd} & \mathrm{he} \end{array} \right].$$

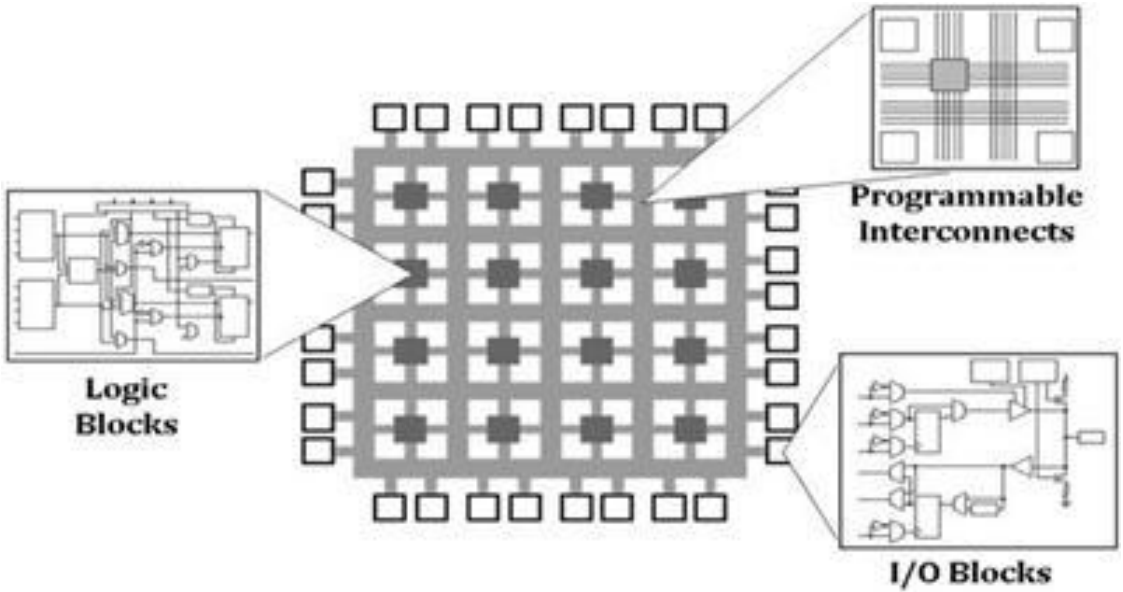
03DE6C570BFE24BFC328CCD7CA46B76EADAF4334

Algoritmo 1 SHA-1 para cada i-ésima mensagem $\mathbf{W}_i$

1: $\mathbf{z}_i \leftarrow [\mathbf{m}_i]$
2: $\mathbf{p}_i \leftarrow \mathit{GeraçãoPreenchimento}(K_i)$
3: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i]$
4: $\mathbf{v}_i \leftarrow \mathit{GeraçãoComprimento}(K_i)$
5: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i\, \mathbf{v}_i]$
6: $\mathbf{h_i} \leftarrow \mathit{InicializaçãoHash}(\,)$
7: **para** $j \leftarrow 0$ **até** $L_i - 1$ **faça**
8: $\quad \mathbf{b}_j \leftarrow \mathit{DivisãoMensagem}(\mathbf{z}_i)$
9: $\quad n \leftarrow -1$
10: $\quad \mathbf{H}(n) \leftarrow \mathit{InicializaVariáveisHash}(\,)$
11: $\quad$ **para** $n \leftarrow 0$ **até** 79 **faça**
12: $\quad\quad \mathbf{w}(n) \leftarrow \mathit{CálculoFunçãoW}(n, \mathbf{b}_j)$
13: $\quad\quad \mathbf{f}(n) \leftarrow \mathit{CálculoFunçãoF}(n, \mathbf{B}(n), \mathbf{C}(n), \mathbf{D}(n))$
14: $\quad\quad \mathbf{H}(n) \leftarrow \mathit{AtualizaVariáveisHash}(\mathbf{H}(n))$
15: $\quad$ **fim para**
16: $\quad \mathbf{h}_i \leftarrow \mathit{AtualizaHash}(\mathbf{H}(n))$
17: **fim para**

FPGA - *Field Programmable Gate Array*

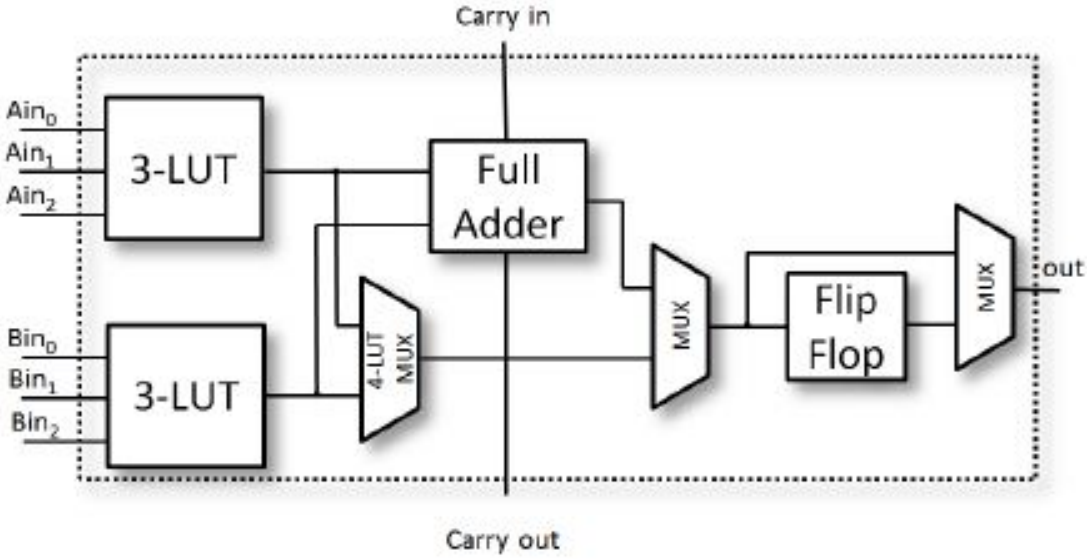
- Hardware reconfigurável composto basicamente por blocos de configuração lógica, blocos de entrada e saídas e chaves de interconexão.



Fonte da Figura: http://www.ni.com/cms/images/devzone/tut/figure3-%20fpga_400x212.jpg

FPGA - *Field Programmable Gate Array*

- Dentro de uma célula lógica, que por alguns fabricantes é chamado do slice



https://www.researchgate.net/profile/Ken_Hawick/publication/267404991/figure/fig2/AS:295589133733895@1447485349726/Figure-2-Simplified-example-of-an-FPGA-logic-slice.png

Xilinx Virtex-6 FPGA
Logic Block Slice Description

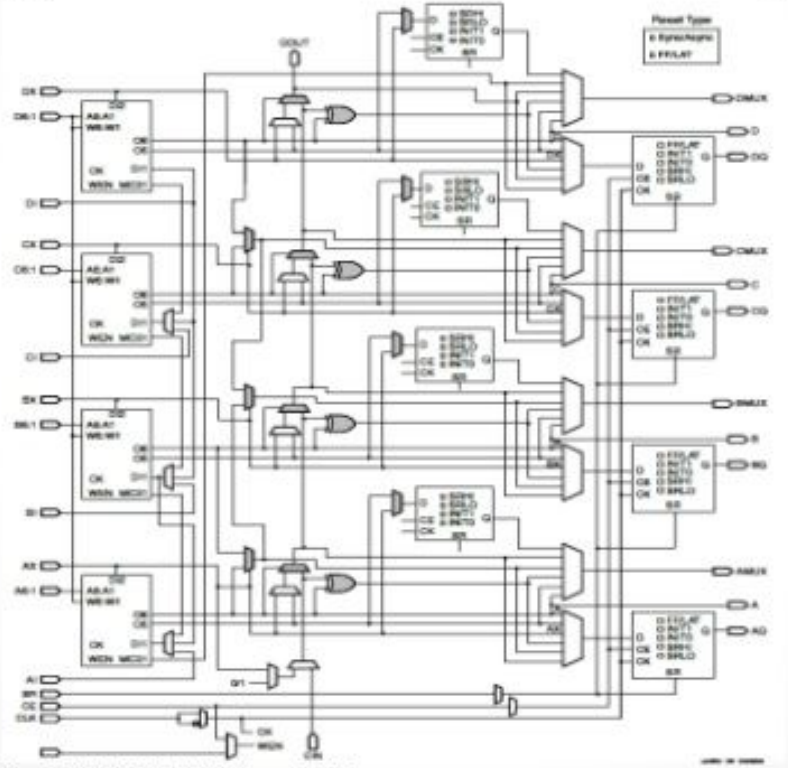
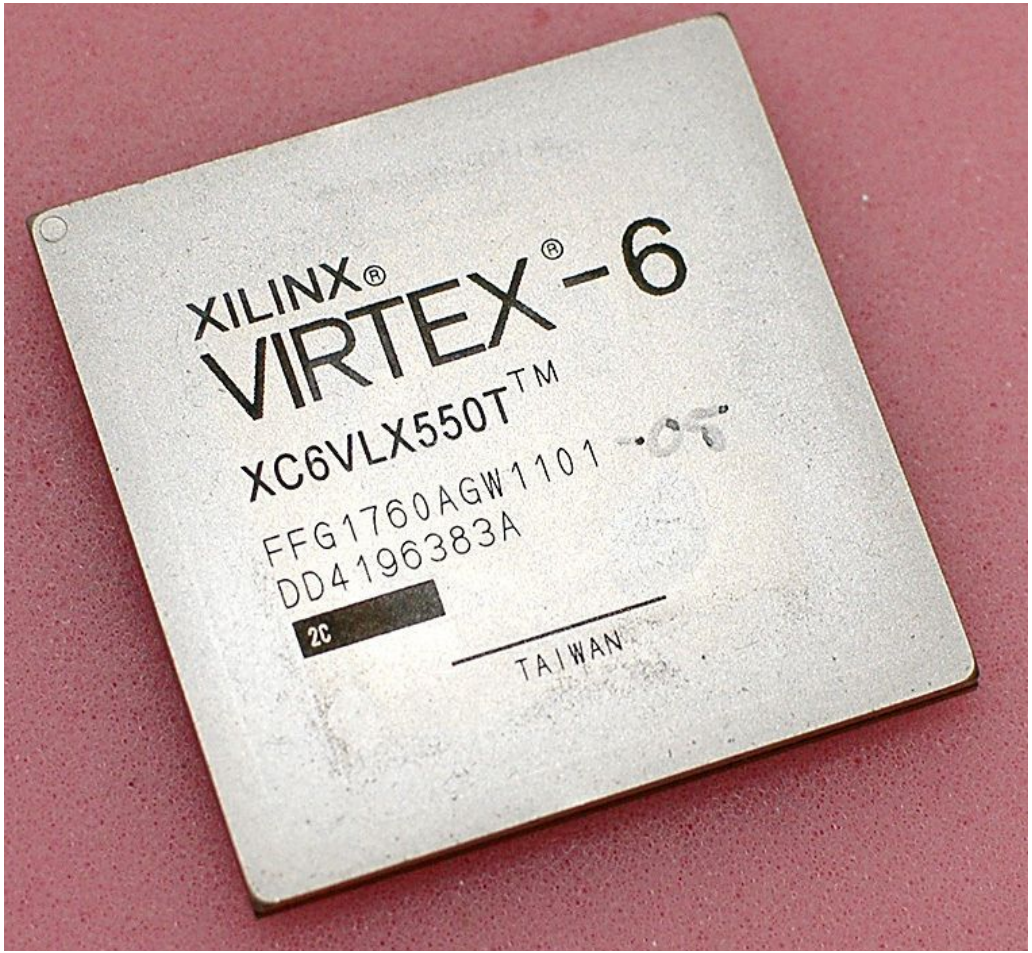
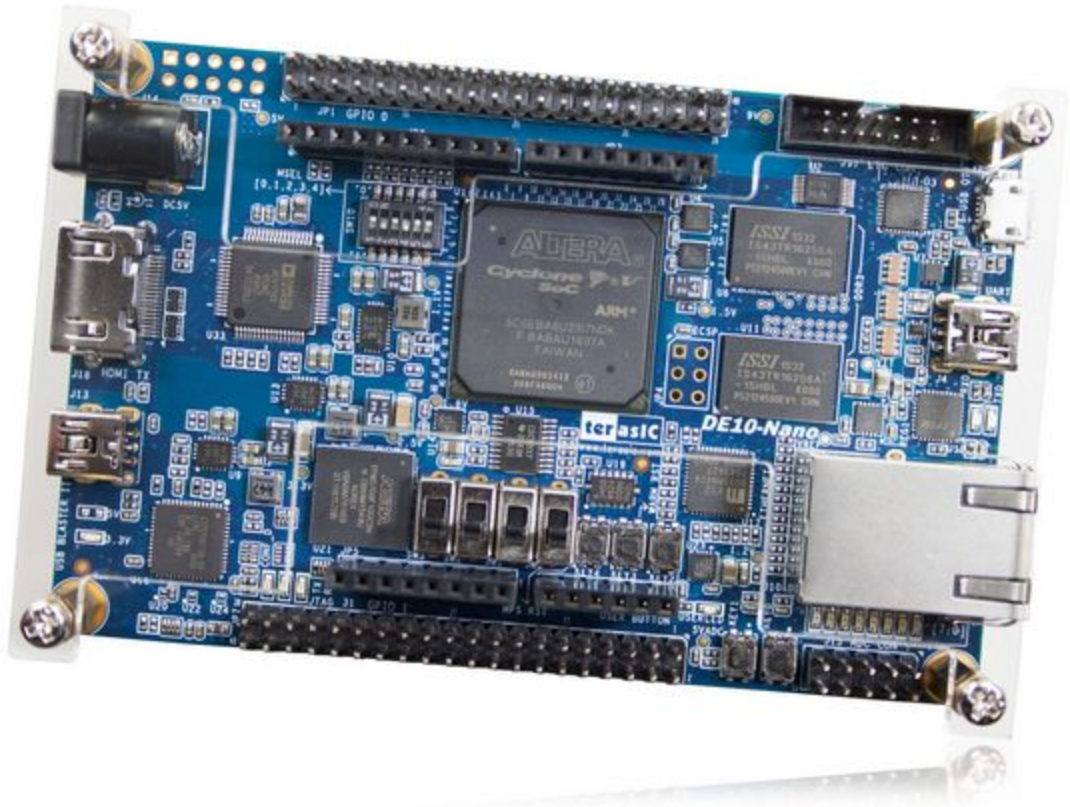


Image Source: "Virtex-6 FPGA CLB User Guide" UG364 (v1.2) February 3, 2012

ML605 REV D
HW-V6-ML605
S/N 0950 – 670
whizzsystems.com
PCI-E-FINGER 8X
DDR3 SODIMM
ENET-MODE SEL
SACE MODE
COMPACT FLASH
XILINX
VIRTEX
00:0A:35:01:F3:0B


XILINX®
VIRTEX®-6
XC6VLX550T™
FFG1760AGW1101
DD4196383A
2C
TAIWAN

DE10-Nano Kit



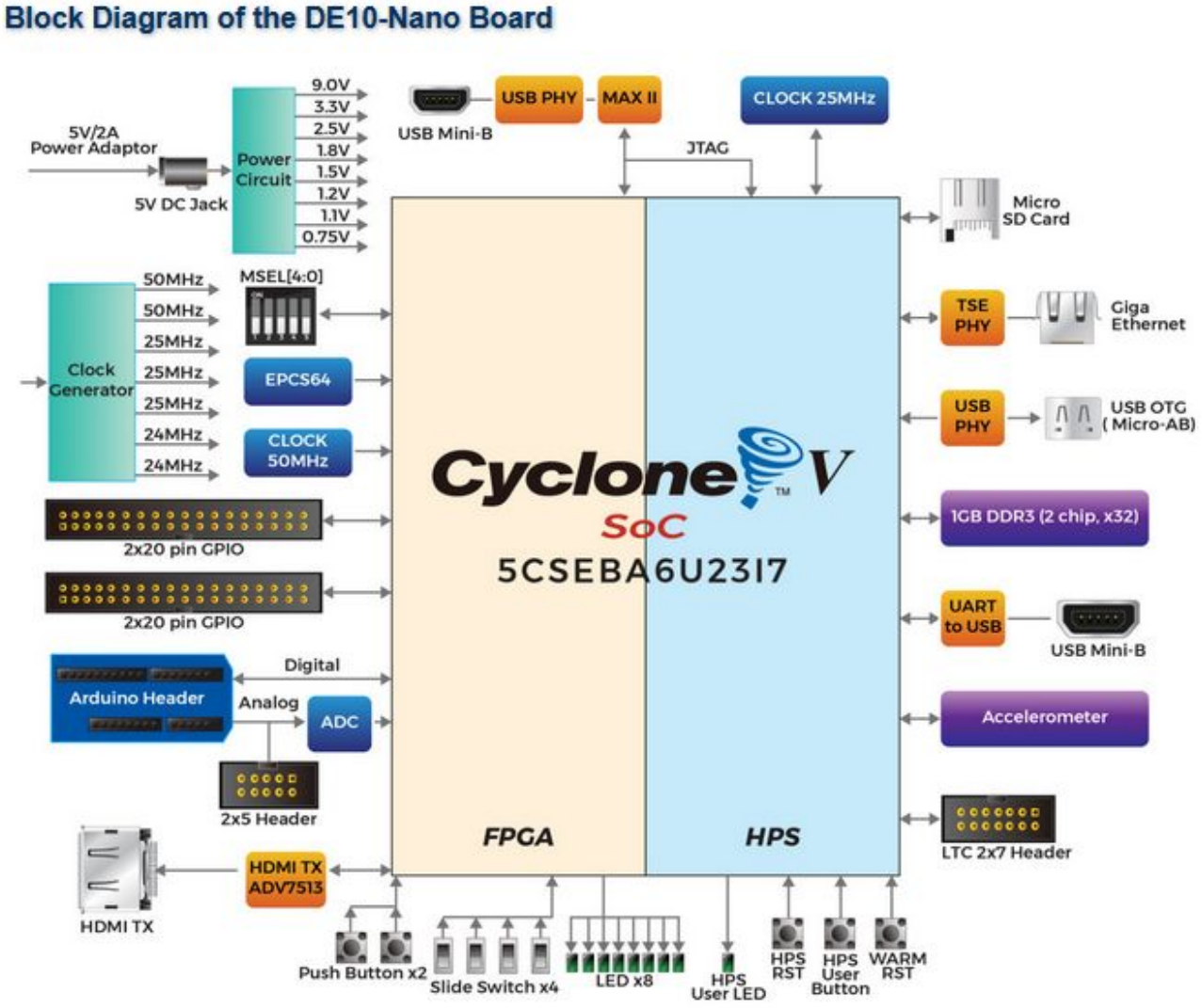
(Currency: USD)

Price: $130

Academic: $99

Buy it now

FPGA


Block Diagram of the DE10-Nano Board
5V/2A Power Adaptor
5V DC Jack
Power Circuit
9.0V
3.3V
2.5V
1.8V
1.5V
1.2V
1.1V
0.75V
USB Mini-B
USB PHY
MAX II
CLOCK 25MHz
JTAG
Micro SD Card
Clock Generator
50MHz
50MHz
25MHz
25MHz
25MHz
24MHz
24MHz
MSEL[4:0]
EPCS64
CLOCK 50MHz
TSE PHY
Giga Ethernet
USB PHY
USB OTG (Micro-AB)
Cyclone V
SoC
5CSEBA6U23I7
1GB DDR3 (2 chip, x32)
2x20 pin GPIO
2x20 pin GPIO
UART to USB
USB Mini-B
Arduino Header
Digital
Analog
ADC
Accelerometer
2x5 Header
FPGA
HPS
LTC 2x7 Header
HDMI TX
HDMI TX ADV7513
Push Button x2
Slide Switch x4
LED x8
HPS User LED
HPS RST
HPS User Button
WARM RST

FPGA Intel-Terasic DE10-Nano

FPGA Device

- Intel Cyclone® V SE 5CSEBA6U23I7 device (110K LEs)
- Serial configuration device – EPCS64 (revision B2 or later)
- USB-Blaster II onboard for programming; JTAG Mode
- HDMI TX, compatible with DVI 1.0 and HDCP v1.4
- 2 push-buttons
- 4 slide switches
- 8 green user LEDs
- Three 50MHz clock sources from the clock generator
- Two 40-pin expansion headers
- One Arduino expansion header (Uno R3 compatibility), can be connected with Ard
- One 10-pin Analog input expansion header (shared with Arduino Analog input)
- A/D converter, 4-pin SPI interface with FPGA

HPS (Hard Processor System)

- 800MHz Dual-core ARM Cortex-A9 processor
- 1GB DDR3 SDRAM (32-bit data bus)
- 1 Gigabit Ethernet PHY with RJ45 connector
- USB OTG Port, USB Micro-AB connector
- Micro SD card socket
- Accelerometer (I2C interface + interrupt)
- UART to USB, USB Mini-B connector
- Warm reset button and cold reset button
- One user button and one user LED
- LTC 2x7 expansion header

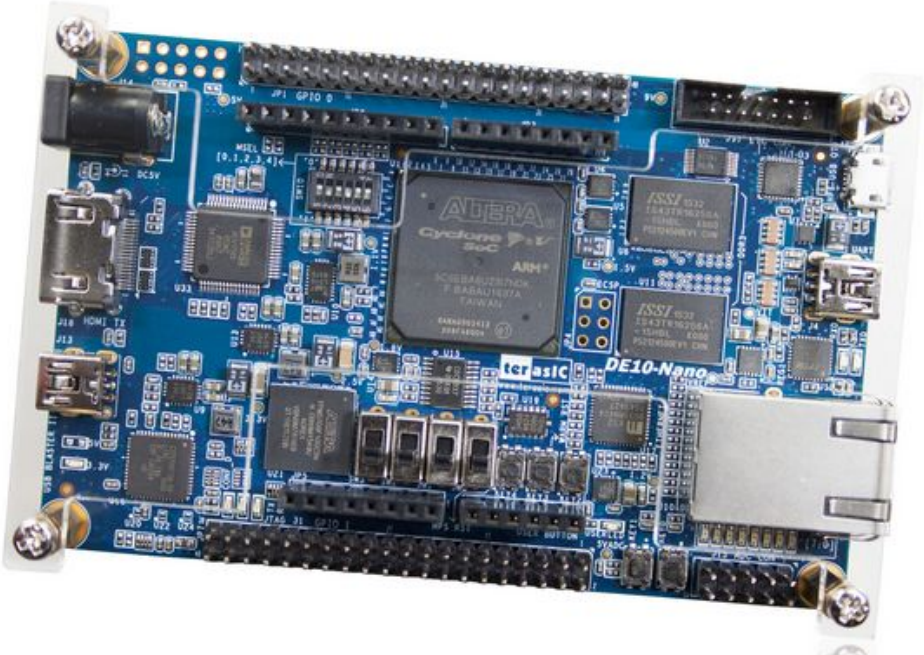
Raspberry-pi 3 DE10-Nano

The Raspberry Pi 3 is the third-generation Raspberry Pi. It replaced the Raspl

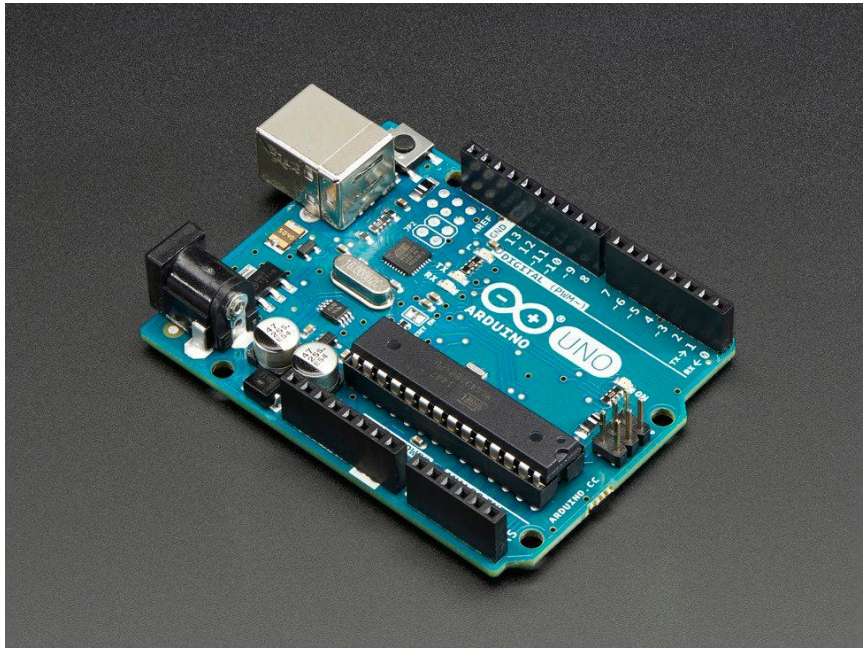
- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

HPS (Hard Processor System)

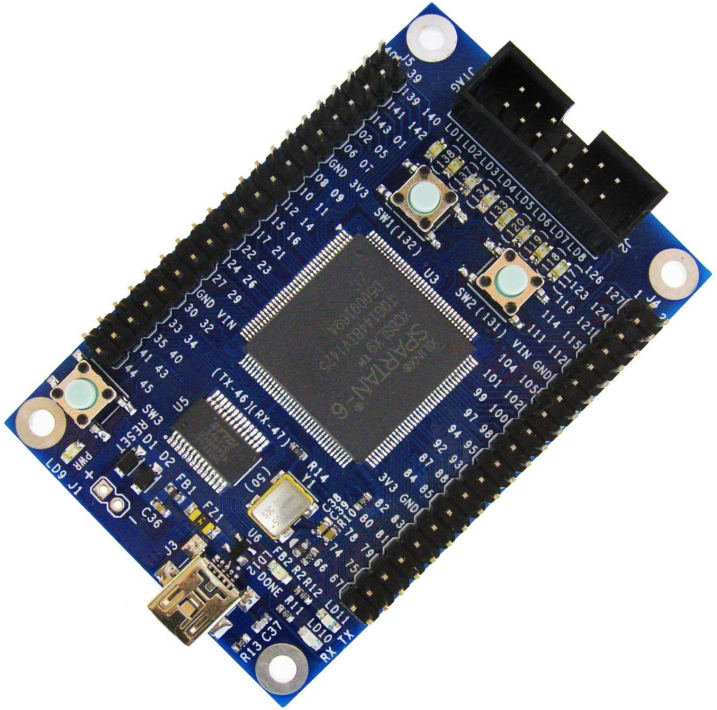
- 800MHz Dual-core ARM Cortex-A9 processor
- 1GB DDR3 SDRAM (32-bit data bus)
- 1 Gigabit Ethernet PHY with RJ45 connector
- USB OTG Port, USB Micro-AB connector
- Micro SD card socket
- Accelerometer (I2C interface + interrupt)
- UART to USB, USB Mini-B connector
- Warm reset button and cold reset button
- One user button and one user LED
- LTC 2x7 expansion header


ALTERA
Cyclone V
SoC
ARM
terasIC
DE10-Nano


Raspberry Pi 3 Model B V1.2
© Raspberry Pi 2015
BROADCOM
BCM2837RIFBG
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
HDMI
ETHERNET

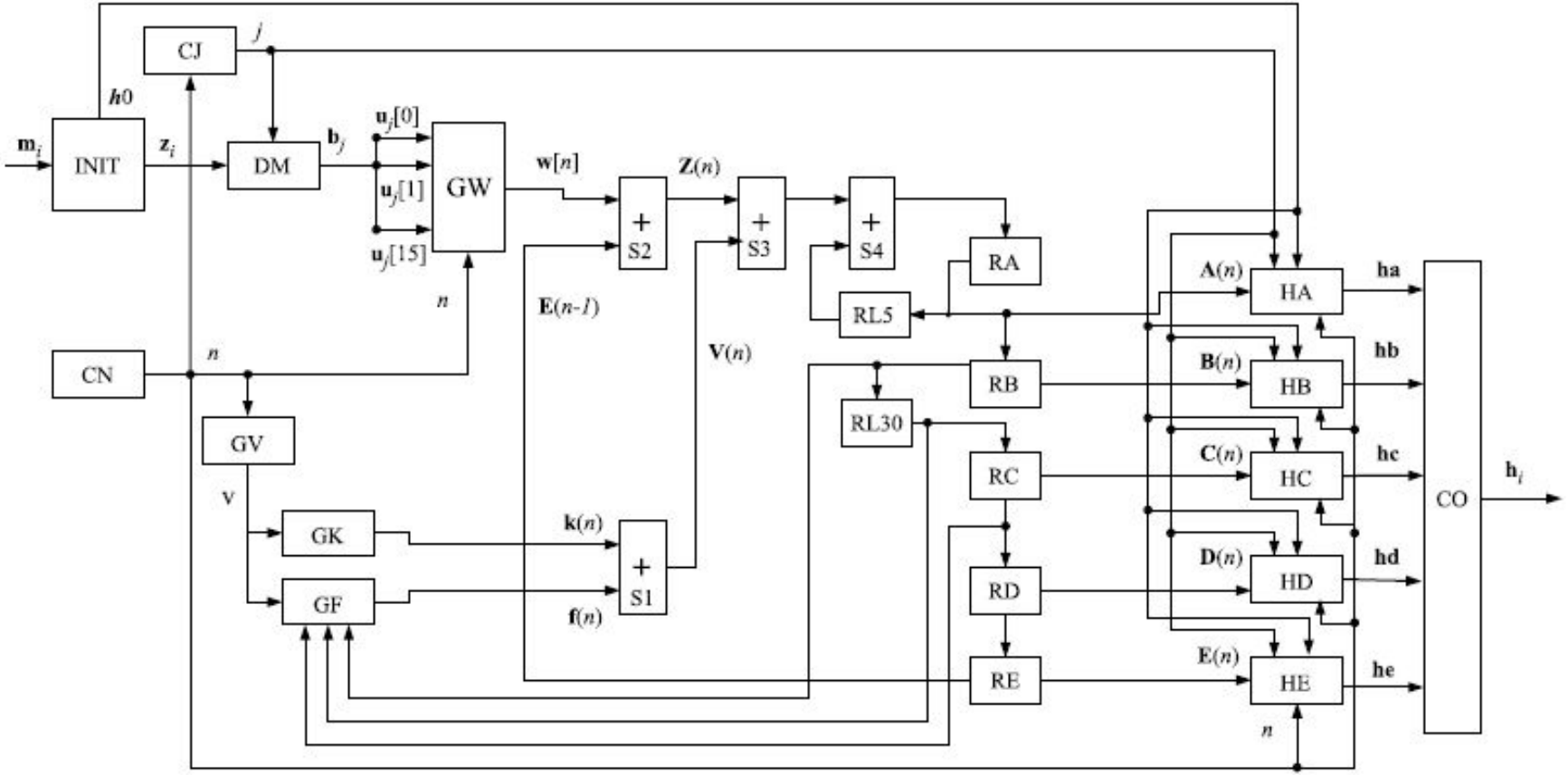


ARDUINO UNO R3



FPGA SPARTAN 6

Implementação Proposta



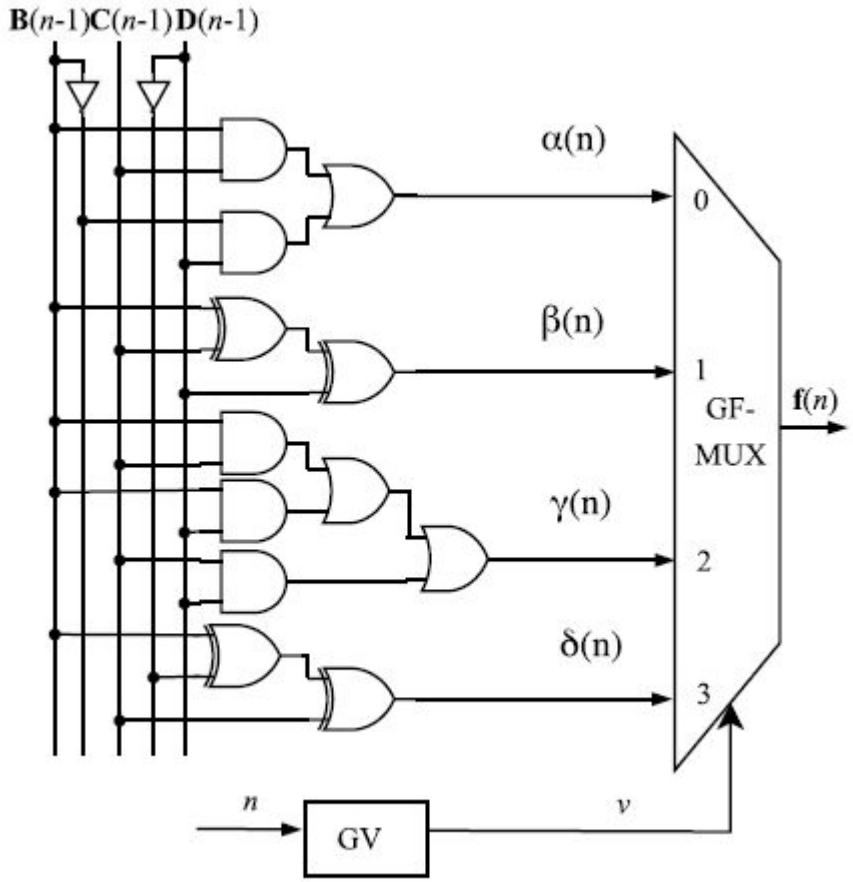
Algoritmo 1 SHA-1 para cada i-ésima mensagem $\mathbf{W}_i$

1: $\mathbf{z}_i \leftarrow [\mathbf{m}_i]$
2: $\mathbf{p}_i \leftarrow \mathit{GeraçãoPreenchimento}(K_i)$
3: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i]$
4: $\mathbf{v}_i \leftarrow \mathit{GeraçãoComprimento}(K_i)$
5: $\mathbf{z}_i \leftarrow [\mathbf{m}_i\, \mathbf{p}_i\, \mathbf{v}_i]$
6: $\mathbf{h_i} \leftarrow \mathit{InicializaçãoHash}(\,)$
7: **para** $j \leftarrow 0$ **até** $L_i - 1$ **faça**
8: $\quad \mathbf{b}_j \leftarrow \mathit{DivisãoMensagem}(\mathbf{z}_i)$
9: $\quad n \leftarrow -1$
10: $\quad \mathbf{H}(n) \leftarrow \mathit{InicializaVariáveisHash}(\,)$
11: $\quad$ **para** $n \leftarrow 0$ **até** 79 **faça**
12: $\quad\quad \mathbf{w}(n) \leftarrow \mathit{CálculoFunçãoW}(n, \mathbf{b}_j)$
13: $\quad\quad \mathbf{f}(n) \leftarrow \mathit{CálculoFunçãoF}(n, \mathbf{B}(n), \mathbf{C}(n), \mathbf{D}(n))$
14: $\quad\quad \mathbf{H}(n) \leftarrow \mathit{AtualizaVariáveisHash}(\mathbf{H}(n))$
15: $\quad$ **fim para**
16: $\quad \mathbf{h}_i \leftarrow \mathit{AtualizaHash}(\mathbf{H}(n))$
17: **fim para**

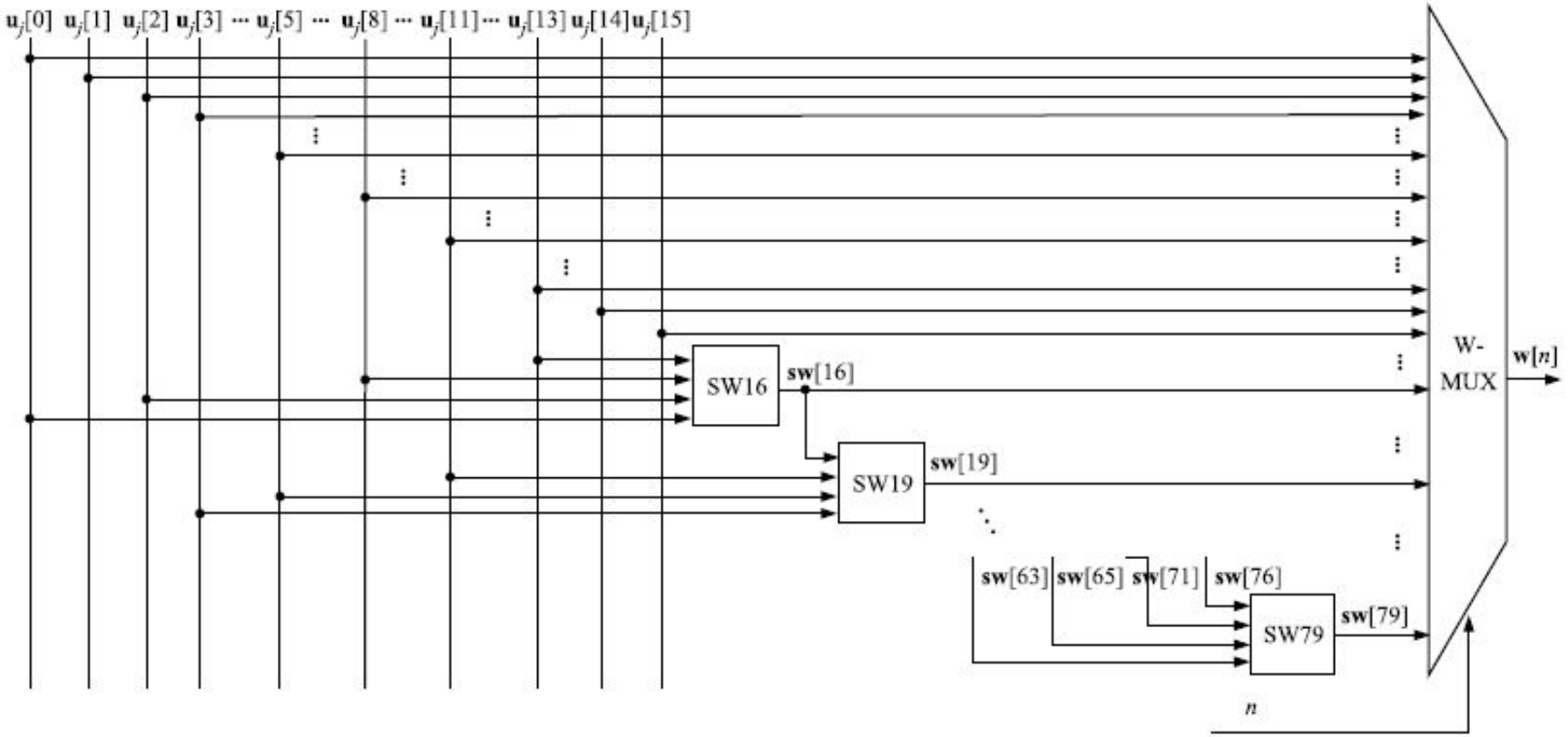
ulo GF

- Implementa a função f(n) apresentado na linha 13 do Algoritmo

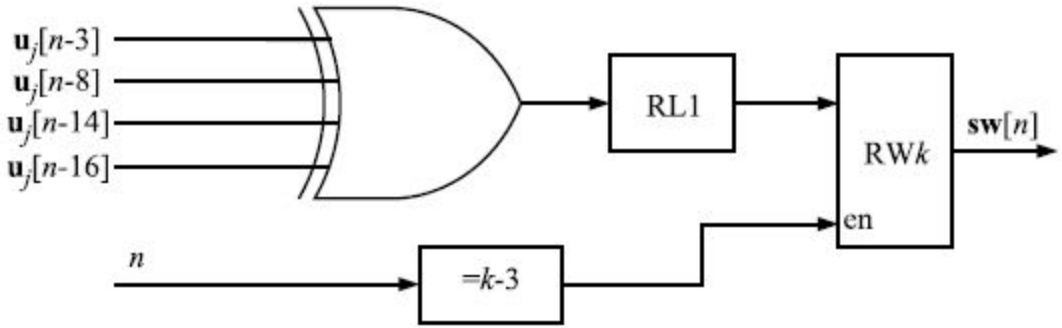
$$f(n, \mathrm{B}, \mathrm{C}, \mathrm{D}) = \begin{cases} \alpha(n) \text{ para } n = 0 \ldots 19 \\ \beta(n) \text{ para } n = 20 \ldots 39 \\ \gamma(n) \text{ para } n = 40 \ldots 59 \\ \delta(n) \text{ para } n = 60 \ldots 79 \end{cases}$$



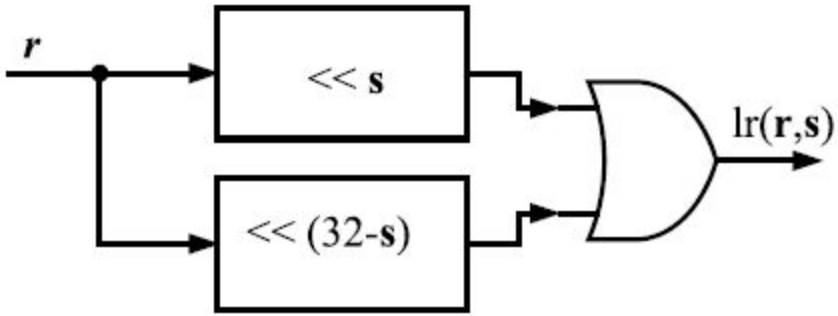
Módulo GW


$u_j[0]$ $u_j[1]$ $u_j[2]$ $u_j[3]$ ⋯ $u_j[5]$ ⋯ $u_j[8]$ ⋯ $u_j[11]$ ⋯ $u_j[13]$ $u_j[14]$ $u_j[15]$
SW16
sw[16]
SW19
sw[19]
sw[63] sw[65] sw[71] sw[76]
SW79
sw[79]
W-MUX
w[n]
n

Operações do modulo SWk



Arquitetura do módulo RL



$$\mathrm{lr}(\mathbf{r}, \mathbf{s}) = (\mathbf{r} \ll \mathbf{s}) \vee (\mathbf{r} \gg (32 - \mathbf{s})),$$

Resultados

Tabela 1. Resultados relativos a ocupação, taxa de amostragem e throughput para várias implementações paralelas do algoritmo SHA-1.

NI	NR	PR (%)	NLUT	PLUT (%)	T_s (ns)	R_s (Gbps)
1	2.154	0,71	2.605	1,72	9,932	0,644
4	8.575	2,84	10, 388	6,89	9,961	2,570
8	17.136	5,68	20.662	13,71	9,965	5,138
16	34.255	11,36	43.263	28,70	9,994	10,246
32	68.498	22,72	86.873	57,64	9,994	18,296
48	102.733	34,08	129.902	86,19	10,909	28,160

Resultados

$$\text{Hash/s} = \begin{cases} 1\ hash \text{ — } 9,932 \cdot 80 \cdot 10^{-9} \\ H\ hash \text{ — } 1s \end{cases}$$

$$\text{Hash/s} = 80,548\ \text{Mhash/s}$$

$$\text{Hash/s} = \begin{cases} 48\ hash \text{ — } 10,909 \cdot 80 \cdot 10^{-9} \\ H\ hash \text{ — } 1s \end{cases}$$

$$\text{Hash/s} = 3,52\ \text{Ghash/s}$$

Resultados

Tabela 3. Resultados relativos as potências dinâmicas dissipadas por cada plataforma de hardware reconfigurável, considerando o clock calculado para alcançar o tempo dos μP de 8 e 32-bits.

Plataforma	MD5	SHA-1
ATmega 2560	105mW	105mW
Intel Galileo Gen 2	550mW	550mW
Artix 7 - Intel Galileo Gen 2	<10 μW	<10 μW
Artix 7 - ATmega 2560	<10μW	<10μW
Spartan 6 - ATmega 2560	<10μW	1.0mW
Spartan 6 - Intel Galileo Gen 2	1.0mW	2.0mW

Obrigado!!!

Dúvidas?

@edusantos33

Se o sábio der ouvidos, aumentará seu conhecimento
Provérbios 1:5