Верхнеуровневые модули проекта asic_core.v:

- K_generator.v
- Logic_module.v
- Memory_switch.v

Пояснения:

Все переменные беззнаковые, имеют размер 32 бита и при вычислениях суммируются по модулю 2^{32}

message – исходное двоичное сообщение

m – преобразованное сообщение

Инициализация переменных

(первые 32 бита *дробных частей* квадратных корней первых восьми простых чисел [от 2 до 19]):

```
h0 := 0x6A09E667

h1 := 0xBB67AE85

h2 := 0x3C6EF372

h3 := 0xA54FF53A

h4 := 0x510E527F

h5 := 0x9B05688C

h6 := 0x1F83D9AB

h7 := 0x5BE0CD19
```

Таблица констант(первые 32 бита *дробных частей* кубических корней первых 64 простых чисел [от 2 до 311]):

```
k[0..63] :=

   0x428A2F98, 0x71374491, 0xB5C0FBCF, 0xE9B5DBA5, 0x3956C25B, 0x59F111F1, 0x923F82A4, 0xAB1C5ED5,

   0xD807AA98, 0x12835B01, 0x243185BE, 0x550C7DC3, 0x72BE5D74, 0x80DEB1FE, 0x9BDC06A7, 0xC19BF174,

   0xE49B69C1, 0xEFBE4786, 0x0FC19DC6, 0x240CA1CC, 0x2DE92C6F, 0x4A7484AA, 0x5CB0A9DC, 0x76F988DA,

   0x983E5152, 0xA831C66D, 0xB00327C8, 0xBF597FC7, 0xC6E00BF3, 0xD5A79147, 0x06CA6351, 0x14292967,
```

0x27B70A85, 0x2E1B2138, 0x4D2C6DFC, 0x53380D13, 0x650A7354, 0x766A0ABB, 0x81C2C92E, 0x92722C85,

0xA2BFE8A1, 0xA81A664B, 0xC24B8B70, 0xC76C51A3, 0xD192E819, 0xD6990624, 0xF40E3585, 0x106AA070,

0x19A4C116, 0x1E376C08, 0x2748774C, 0x34B0BCB5, 0x391C0CB3, 0x4ED8AA4A, 0x5B9CCA4F, 0x682E6FF3,

0x748F82EE, 0x78A5636F, 0x84C87814, 0x8CC70208, 0x90BEFFFA, 0xA4506CEB, 0xBEF9A3F7, 0xC67178F2

Предварительная обработка:

m := message ‖ [*единичный бит*]

m := m ‖ [k *нулевых бит*], где **k** — наименьшее неотрицательное число, такое что

(L + 1 + K) mod 512 = 448, где L — число бит в сообщении (сравнима по модулю 512 с 448)

m := m ‖ *Длина*(message) — длина исходного сообщения в битах в виде 64-битного числа

с порядком байтов от старшего к младшему*Далее сообщение обрабатывается последовательными порциями по 512 бит:*

разбить сообщение на куски по 512 бит

для каждого куска

разбить кусок на 16 слов длиной 32 бита (с порядком байтов от старшего к младшему внутри слова): w[0..15]

Сгенерировать дополнительные 48 слов:

для i **от** 16 **до** 63

s0 := (w[i-15] **rotr** 7) **xor** (w[i-15] **rotr** 18) **xor** (w[i-15] **shr** 3)

s1 := (w[i-2] **rotr** 17) **xor** (w[i-2] **rotr** 19) **xor** (w[i-2] **shr** 10)

w[i] := w[i-16] **+** s0 **+** w[i-7] **+** s1

Инициализация вспомогательных переменных:

a := h0

b := h1

c := h2

```
    d := h3
    e := h4
    f := h5
    g := h6
    h := h7

    Основной цикл:
    для i от 0 до 63
        Σ0 := (a rotr 2) xor (a rotr 13) xor (a rotr 22)
        Ma := (a and b) xor (a and c) xor (b and c)
        t2 := Σ0 + Ma
        Σ1 := (e rotr 6) xor (e rotr 11) xor (e rotr 25)
        Ch := (e and f) xor ((not e) and g)
        t1 := h + Σ1 + Ch + k[i] + w[i]

        h := g
        g := f
        f := e
        e := d + t1
        d := c
        c := b
        b := a
        a := t1 + t2

    Добавить полученные значения к ранее вычисленному результату:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
    h4 := h4 + e
```

```
    h5 := h5 + f
    h6 := h6 + g
    h7 := h7 + h


Получить итоговое значение хеша:
```

`digest = hash = h0 ‖ h1 ‖ h2 ‖ h3 ‖ h4 ‖ h5 ‖ h6 ‖ h7`