Contents

Constraint Profiles

Constraint Type	Rows per constraint	Arithmetic Degree	Description
Poseidon	12	8*N	Poseidon Permutation Rounds: 53, width: 3, sbox alpha: 7
EC Addition	1	4*N	Addition of (non-special constrained) EC points
EC Doubling	1	8*N	Doubling of (non-special constrained) EC points
Scalar Multiplication, With Packing	103	8*N	Scalar multiplication of EC point by 256-bit integer
Endo-Scalar Multiplication, With Packing	64	8*N	Endo-scalar multiplication of EC point by 256-bit integer

Permutation Constraints

Wire permutation argument is executed/checked only on 7 (out of total 15) left gate wires of the circuit designated by green background color in the tables below. The other 8 advice (local memory) right-most gate wires do not participate in the permutation argument and designated by red background color in the tables below.

EC Operations

Variable-base Scalar Multiplication

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Type
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	x_T	y_T	x_S	y_S	x_P	y_P	n=0	x_r	y_r	s1	s2	b1	s3	s4	b2	**VBSM**
i+1	s5	b3	x_S	y_S	x_P	y_P	n	x_r	y_r	x_v	y_v	s1	b1	s3	b2	**ZERO**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i+100	x_T	y_T	x_S	y_S	x_P	y_P	n	x_r	y_r	s1	s2	b1	s3	s4	b2	**VBSM**
i+101	s5	b3	x_S	y_S	x_P	y_P	n	x_r	y_r	x_v	y_v	s1	b1	s3	b2	**ZERO**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

VBSM gate constraints for THIS witness row

- b1*(b1-1) = 0
- b2*(b2-1) = 0
- (xp - xt) * s1 = yp – (2b1-1)*yt
- s1^2 - s2^2 = xt - xr
- (2*xp + xt – s1^2) * (s1 + s2) = 2*yp
- (xp – xr) * s2 = yr + yp
- (xr - xt) * s3 = yr – (2b2-1)*yt
- S3^2 – s4^2 = xt - xs
- (2*xr + xt – s3^2) * (s3 + s4) = 2*yr
- (xr – xs) * s4 = ys + yr
- n = 32*n_next + 16*b1 + 8*b2 + 4*b1_next + 2*b2_next + b3_next

The constraints above are derived from the following EC Affine arithmetic equations:

```
(xq1 - xp) * s1 = yq1 - yp
s1^2 - s2^2 = xq1 - xr
(2*xp + xq1 – s1^2) * (s1 + s2) = 2*yp
(xp – xr) * s2 = yr + yp

(xq2 - xr) * s3 = yq2 - yr
s3^2 – s4^2 = xq2 - xs
(2*xr + xq2 – s3^2) * (s3 + s4) = 2*yr
(xr – xs) * s4 = ys + yr
```

VBSM gate constraints for NEXT witness row

- b1*(b1-1) = 0
- b2*(b2-1) = 0
- b3*(b3-1) = 0
- (**xt** - xp) * s1 = **(2b1-1)*yt** - yp
- (2*xp – s1^2 + **xt**) * ((xp – xr) * s1 + yr + yp) = (xp – xr) * 2*yp
- (yr + yp)^2 = (xp – xr)^2 * (s1^2 – **xt** + xr)
- (**xt** - xr) * s3 = **(2b2-1)*yt** - yr
- (2*xr – s3^2 + **xt**) * ((xr – xv) * s3 + yv + yr) = (xr – xv) * 2*yr
- (yv + yr)^2 = (xr – xv)^2 * (s3^2 – **xt** + xv)
- (**xt** - xv) * s5 = **(2b3-1)*yt** - yv
- (2*xv – s5^2 + **xt**) * ((xv – xs) * s5 + ys + yv) = (xv – xs) * 2*yv
- (ys + yv)^2 = (xv – xs)^2 * (s5^2 – **xt** + xs)

The constraints above are derived from the following EC Affine arithmetic equations:

```
    (xq1 - xp) * s1 = yq1 - yp
    s1^2 - s2^2 = xq1 - xr
    (2*xp + xq1 – s1^2) * (s1 + s2) = 2*yp
    (xp – xr) * s2 = yr + yp

    (xq2 - xr) * s3 = yq2 - yr
    s3^2 – s4^2 = xq2 - xv
    (2*xr + xq2 – s3^2) * (s3 + s4) = 2*yr
    (xr – xv) * s4 = yv + yr

    (xq3 - xv) * s5 = yq3 - yv
    s5^2 – s6^2 = xq3 - xs
    (2*xv + xq3 – s5^2) * (s5 + s6) = 2*yv
    (xv – xs) * s6 = ys + yv
```

=>

```
    (xq1 - xp) * s1 = yq1 - yp
    (2*xp – s1^2 + xq1) * ((xp – xr) * s1 + yr + yp) = (xp – xr) * 2*yp
    (yr + yp)^2 = (xp – xr)^2 * (s1^2 – xq1 + xr)

    (xq2 - xr) * s3 = yq2 - yr
    (2*xr – s3^2 + xq2) * ((xr – xv) * s3 + yv + yr) = (xr – xv) * 2*yr
    (yv + yr)^2 = (xr – xv)^2 * (s3^2 – xq2 + xv)

    (xq3 - xv) * s5 = yq3 - yv
    (2*xv – s5^2 + xq3) * ((xv – xs) * s5 + ys + yv) = (xv – xs) * 2*yv
    (ys + yv)^2 = (xv – xs)^2 * (s5^2 – xq3 + xs)
```

Variable-base Endo-scalar Multiplication

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Type
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	x_T	y_T	x_S	y_S	x_P	y_P	n	x_r	y_r	s1	s3	b1	b2	b3	b4	EVBSM
i+1	x_T	y_T	x_S	y_S	x_P	y_P	n	x_r	y_r	s1	s3	b1	b2	b3	b4	EVBSM
⋮	x_T	y_T	x_S	y_S	x_P	y_P	n	x_r	y_r	s1	s3	b1	b2	b3	b4	⋮
i+62	x_T	y_T	x_S	y_S	x_P	y_P	n	x_r	y_r	s1	s3	b1	b2	b3	b4	EVBSM
i+63	x_T	y_T	x_S	y_S	x_P	y_P	n	x_r	y_r	s1	s3	b1	b2	b3	b4	EVBSM
⋮	Z	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

EVBSM gate constraints

- b1*(b1-1) = 0
- b2*(b2-1) = 0
- b3*(b3-1) = 0
- b4*(b4-1) = 0
- ((1 + (endo - 1) * b2) * xt - xp) * s1 = (2*b1-1)*yt - yp
- (2*xp – s1^2 + (1 + (endo - 1) * b2) * xt) * ((xp – xr) * s1 + yr + yp) = (xp – xr) * 2*yp
- (yr + yp)^2 = (xp – xr)^2 * (s1^2 – (1 + (endo - 1) * b2) * xt + xr)
- ((1 + (endo - 1) * b2) * xt - xr) * s3 = (2*b3-1)*yt - yr
- (2*xr – s3^2 + (1 + (endo - 1) * b4) * xt) * ((xr – xs) * s3 + ys + yr) = (xr – xs) * 2*yr
- (ys + yr)^2 = (xr – xs)^2 * (s3^2 – (1 + (endo - 1) * b4) * xt + xs)
- n = 16*n_next + 8*b1 + 4*b2 + 2*b3 + b4

The constraints above are derived from the following EC Affine arithmetic equations:

```
(xq1 - xp) * s1 = yq1 - yp
(2*xp – s1^2 + xq1) * ((xp – xr) * s1 + yr + yp) = (xp – xr) * 2*yp
(yr + yp)^2 = (xp – xr)^2 * (s1^2 – xq1 + xr)

(xq2 - xr) * s3 = yq2 - yr
(2*xr – s3^2 + xq2) * ((xr – xs) * s3 + ys + yr) = (xr – xs) * 2*yr
(ys + yr)^2 = (xr – xs)^2 * (s3^2 – xq2 + xs)
```

EC Point Addition

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
⋮	x_1	y_1	x_2	y_2	x_3	y_3	r	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

ADD gate constraints

- (x2 - x1) * (y3 + y1) - (y1 - y2) * (x1 - x3)
- (x1 + x2 + x3) * (x1 - x3) * (x1 - x3) - (y3 + y1) * (y3 + y1)
- (x2 - x1) * r = 1

The constraints above are derived from the following EC Affine arithmetic equations:

```
(x2 - x1) * s = y2 - y1
s * s = x1 + x2 + x3
(x1 - x3) * s = y3 + y1

=>

(x2 - x1) * (y3 + y1) - (y1 - y2) * (x1 - x3)
(x1 + x2 + x3) * (x1 - x3) * (x1 - x3) - (y3 + y1) * (y3 + y1)
```

-

EC Point Doubling-Tripling

This constrains the computation of the following multiples of EC point: [2]P, [3]P. This, in particular, can be used to efficiently augment (with only one of these constraints) the scalar multiplication computation where double and triple operations are needed.

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
⋮	x_1	y_1	x_2	y_2	x_3	y_3	r_1	r_2	⋮	⋮	⋮	⋮	⋮	⋮	⋮

DOUBLE gate constraints

- 4 * y1^2 * (x2 + 2*x1) = 9 * x1^4
- 2 * y1 * (y2 + y1) = (3 * x1^2) * (x1 – x2)
- y1 * r1 = 1
-
- (x2 - x1) * (y3 + y1) - (y1 - y2) * (x1 - x3)
- (x1 + x2 + x3) * (x1 - x3) * (x1 - x3) - (y3 + y1) * (y3 + y1)
- (x2 - x1) * r2 = 1

The constraints above are derived from the following EC Affine arithmetic equations:

Doubling

```
    2 * s * y1 = 3 * x1^2
    x2 = s^2 – 2*x1
    y2 = y1 + s * (x2 – x1)

    =>

    2 * s * y1 = 3 * x1^2
    x2 = s^2 – 2*x1
    2 * y1 * (y2 - y1) = 3 * x1^2 * (x2 – x1)

    =>

    4 * y1^2 * (x2 + 2*x1) = 9 * x1^4
    2 * y1 * (y2 + y1) = 3 * x1^2 * (x1 – x2)
```

Addition

```
    (x2 - x1) * s = y2 - y1
    s * s = x1 + x2 + x3
    (x1 - x3) * s = y3 + y1

    =>

    (x2 - x1) * (y3 + y1) - (y1 - y2) * (x1 - x3)
    (x1 + x2 + x3) * (x1 - x3) * (x1 - x3) - (y3 + y1) * (y3 + y1)
```

Poseidon Hash

Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Type
i+0	**T0_0**	**T0_1**	**T0_2**	T1_0	T1_1	T1_2	T2_0	T2_1	T2_2	T3_0	T3_1	T3_2	T4_0	T4_1	T4_2	PSDN
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	PSDN
i+10	**T50_0**	**T50_1**	**T50_2**	T51_0	T51_1	T51_2	T52_0	T52_1	T52_2	T53_0	T53_1	T53_2	T54_0	T54_1	T54_2	PSDN
i+11	**T55_0**	**T55_1**	**T55_2**	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	ZERO

53-round Poseidon permutation state starts with T0_0 T0_1 T0_2 and ends up with T55_0 55_1 T55_2. Notice that the last row, being the zero-constraint, intentionally does not constraint its row.

POSEIDON gate constraints

- STATE(i+1) = STATE(i)^alpha * MDS + RC