

```
1 index.html:
2 <!DOCTYPE html>
3 <html lang="ru">
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Редактор логических схем</title>
8     <link rel="stylesheet" href="css/styles.css">
9 </head>
10 <body>
11     <div id="app">
12         <div id="menu">
13             <button class="menu-btn" id="btn-new"> Новый</button>
14             <button class="menu-btn" id="btn-save"> Сохранить</button>
15             <button class="menu-btn" id="btn-load"> Загрузить</button>
16             <button class="menu-btn" id="btn-generate-code"> Код</button>
17             <button class="menu-btn" id="btn-project-settings"> Свойства проекта</
button>
18         <div class="menu-separator"></div>
19         <div class="zoom-controls">
20             <button class="menu-btn zoom-btn" id="btn-zoom-out">-</button>
21             <span id="zoom-level">100%</span>
22             <button class="menu-btn zoom-btn" id="btn-zoom-in">+</button>
23             <button class="menu-btn" id="btn-zoom-fit"> Вписать</button>
24             <button class="menu-btn" id="btn-zoom-reset">1:1</button>
25         </div>
26         <input type="file" id="file-input" accept=".json">
27     </div>
28
29     <div id="main">
30         <div id="palette">
31             <h3> Элементы</h3>
32
33             <div class="palette-section">
34                 <div class="palette-section-title">ВХОДЫ</div>
35
36                 <div class="palette-item" data-type="input-signal">
37                     <svg viewBox="0 0 60 40">
38                         <polygon points="0,5 40,5 55,20 40,35 0,35" fill="#0f3460" stroke="#4a90d9" stroke-width="2"/>
39                         <text x="12" y="24" fill="#eee" font-size="10">IN</text>
40                     </svg>
41                     <div class="palette-item-name">Входной сигнал</div>
42                 </div>
43             </div>
44             <div class="palette-section">
45                 <div class="palette-section-title">ВЫХОДЫ</div>
46
47                 <div class="palette-item" data-type="output">
48                     <svg viewBox="0 0 60 40">
49                         <rect x="5" y="5" width="50" height="30" rx="6" fill="none" stroke="#10b981" stroke-width="2" stroke-dasharray="4,2"/>
50                         <text x="12" y="24" fill="#10b981" font-size="9">Выход</text>
51                     </svg>
52                     <div class="palette-item-name">Выход</div>
53                 </div>
54             </div>
55
56             <div class="palette-section">
57                 <div class="palette-section-title">ЛОГИЧЕСКИЕ</div>
58
59                 <div class="palette-item" data-type="and">
60                     <svg viewBox="0 0 60 40">
61                         <rect x="5" y="5" width="50" height="30" rx="5"
```

```
fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
62           <text x="22" y="25" fill="#eee" font-size="12" font-
weight="bold">И</text>
63           </svg>
64           <div class="palette-item-name">И (AND)</div>
65       </div>
66
67           <div class="palette-item" data-type="or">
68               <svg viewBox="0 0 60 40">
69                   <rect x="5" y="5" width="50" height="30" rx="5"
fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
70                   <text x="12" y="25" fill="#eee" font-size="11" font-
weight="bold">ИЛИ</text>
71               </svg>
72               <div class="palette-item-name">ИЛИ (OR)</div>
73           </div>
74
75           <div class="palette-item" data-type="not">
76               <svg viewBox="0 0 60 40">
77                   <rect x="5" y="5" width="50" height="30" rx="5"
fill="#0f3460" stroke="#a855f7" stroke-width="2"/>
78                   <text x="12" y="25" fill="#eee" font-size="11" font-
weight="bold">НЕТ</text>
79               </svg>
80               <div class="palette-item-name">НЕТ (NOT)</div>
81           </div>
82       </div>
83
84           <div class="palette-section">
85               <div class="palette-section-title">СРАВНЕНИЕ</div>
86
87               <div class="palette-item" data-type="if">
88                   <svg viewBox="0 0 60 40">
89                       <polygon points="30,3 57,20 30,37 3,20" fill="#0f3460"
stroke="#e94560" stroke-width="2"/>
90                       <text x="14" y="24" fill="#eee" font-size="9" font-
weight="bold">ЕСЛИ</text>
91                   </svg>
92                   <div class="palette-item-name">ЕСЛИ (IF)</div>
93               </div>
94           </div>
95
96           <div class="palette-section">
97               <div class="palette-section-title">РАЗВЕТВЛЕНИЕ</div>
98
99               <div class="palette-item" data-type="separator">
100                  <svg viewBox="0 0 60 40">
101                      <rect x="5" y="8" width="50" height="24" rx="3"
fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
102                      <text x="8" y="25" fill="#f59e0b" font-size="10" font-
weight="bold">/>/</text>
103                  </svg>
104                  <div class="palette-item-name">Сепаратор</div>
105              </div>
106          </div>
107
108          <div class="palette-section">
109              <div class="palette-section-title">ЗНАЧЕНИЯ</div>
110
111              <div class="palette-item" data-type="const">
112                  <svg viewBox="0 0 60 40">
113                      <rect x="10" y="8" width="40" height="24" rx="3"
fill="#0f3460" stroke="#3b82f6" stroke-width="2"/>
114                      <text x="24" y="25" fill="#3b82f6" font-size="14" font-
weight="bold">С</text>
```

```
115                     </svg>
116                     <div class="palette-item-name">Константа</div>
117                 </div>
118
119                     <div class="palette-item" data-type="formula">
120                         <svg viewBox="0 0 60 40">
121                             <rect x="5" y="5" width="50" height="30" rx="5"
122                             fill="#0f3460" stroke="#f59e0b" stroke-width="2"/>
123                             <text x="12" y="25" fill="#f59e0b" font-size="11" font-
124                             weight="bold">f(x)</text>
125                         </svg>
126                         <div class="palette-item-name">Формула</div>
127                     </div>
128
129                     <div class="type-legend">
130                         <div class="type-legend-item">
131                             <div class="type-legend-dot logic"></div>
132                             <span>Логический</span>
133                         </div>
134                         <div class="type-legend-item">
135                             <div class="type-legend-dot number"></div>
136                             <span>Числовой</span>
137                         </div>
138                     </div>
139
140                     <div id="workspace-container">
141                         <svg id="connections-svg"></svg>
142                         <div id="workspace"></div>
143                         <!-- Прямоугольник для выделения элементов -->
144                         <div id="selection-rect"></div>
145
146                         <!-- Мини-карта -->
147                         <div id="minimap">
148                             <div id="minimap-viewport"></div>
149                             <canvas id="minimap-canvas"></canvas>
150                         </div>
151
152                         <!-- Координаты и информация -->
153                         <div id="viewport-info">
154                             <span id="cursor-pos">X: 0, Y: 0</span>
155                             <span id="selection-info"></span>
156                         </div>
157                     </div>
158                 </div>
159             </div>
160
161             <!-- Модальные окна -->
162             <div id="modal-overlay">
163                 <div id="modal">
164                     <h3 id="modal-title">Свойства элемента</h3>
165                     <div id="modal-content"></div>
166                     <div class="modal-buttons">
167                         <button class="modal-btn cancel" id="modal-cancel">Отмена</button>
168                         <button class="modal-btn save" id="modal-save">Сохранить</button>
169                     </div>
170                 </div>
171             </div>
172
173             <!-- Модальное окно свойств проекта -->
174             <div id="project-modal-overlay" class="modal-overlay-class">
175                 <div id="project-modal" class="modal-class">
176                     <h3>Свойства проекта</h3>
177                     <div id="project-modal-content"></div>
```

```
178         <div class="modal-buttons">
179             <button class="modal-btn cancel" id="project-modal-cancel">Отмена</
180         button>
181             <button class="modal-btn save" id="project-modal-save">Сохранить</
182         button>
183     </div>
184 
185     <div id="code-modal-overlay" class="modal-overlay-class">
186         <div id="code-modal" class="modal-class">
187             <h3>Сгенерированный код</h3>
188             <textarea id="code-output" style="width:100%; height:300px;"></textarea>
189             <div class="modal-buttons">
190                 <button class="modal-btn cancel" id="code-modal-close">Закрыть</
191             button>
192                 </div>
193             </div>
194 
195     <div id="context-menu">
196         <div class="context-item" id="ctx-properties"> Свойства</div>
197         <div class="context-item" id="ctx-delete"> Удалить</div>
198     </div>
199 
200     <!-- Модули JavaScript -->
201     <!-- Модули JavaScript -->
202     <script src="js/config.js"></script>
203     <script src="js/state.js"></script>
204     <script src="js/utils.js"></script>
205     <script src="js/viewport.js"></script>
206     <script src="js/elements.js"></script>
207     <script src="js/connections.js"></script>
208     <script src="js/outputs.js"></script> <!-- ← Этот файл опционален теперь -->
209     <script src="js/modal.js"></script>
210     <script src="js/project.js"></script>
211     <script src="js/codegen_graph.js"></script>
212     <script src="js/codegen_optimizer.js"></script>
213     <script src="js/codegen.js"></script>
214     <script src="js/settings.js"></script>
215 
216     <script src="js/app.js"></script>
217 
218     <div id="modal-project-list" class="modal hidden">
219         <div class="modal__content modal__content--wide">
220             <h2 class="modal__title">Выбор проекта</h2>
221 
222             <div class="project-list__toolbar">
223                 <input id="project-search" type="text" placeholder="Фильтр по имени или
описанию..." />
224                 <button id="project-refresh" class="btn btn-secondary">Обновить</button>
225             </div>
226 
227             <div class="project-list__table-container">
228                 <table class="project-list__table">
229                     <thead>
230                         <tr>
231                             <th>Файл</th>
232                             <th>Tagname</th>
233                             <th>Description</th>
234                             <th>Тип</th>
235                         </tr>
236                     </thead>
237                     <tbody id="project-list-body">
238                         <tr><td colspan="4" class="project-list__empty">Загрузка...</td></tr>
```

```
239             </tbody>
240         </table>
241     </div>
242
243     <div class="modal__actions">
244         <button id="project-cancel" class="btn btn-secondary">Отмена</button>
245         <button id="project-load" class="btn btn-primary" disabled>Загрузить</
246         button>
247             </div>
248     </div>
249
250 </body>
251 </html>
252
253 styles.css:
254 *
255     margin: 0;
256     padding: 0;
257     box-sizing: border-box;
258 }
259
260 body {
261     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
262     background: #lala2e;
263     color: #eee;
264     overflow: hidden;
265 }
266
267 #app {
268     display: flex;
269     flex-direction: column;
270     height: 100vh;
271 }
272
273 /* ===== МЕНЮ ===== */
274 #menu {
275     background: #16213e;
276     padding: 10px 20px;
277     display: flex;
278     gap: 10px;
279     align-items: center;
280     border-bottom: 2px solid #0f3460;
281     z-index: 100;
282     flex-wrap: wrap;
283 }
284
285 .menu-btn {
286     background: #0f3460;
287     color: #eee;
288     border: none;
289     padding: 8px 16px;
290     border-radius: 5px;
291     cursor: pointer;
292     transition: background 0.3s;
293     font-size: 13px;
294 }
295
296 .menu-btn:hover {
297     background: #e94560;
298 }
299
300 .menu-separator {
301     width: 1px;
302     height: 30px;
```

```
303     background: #0f3460;
304     margin: 0 10px;
305 }
306
307 .zoom-controls {
308     display: flex;
309     align-items: center;
310     gap: 8px;
311     background: #0a0a1a;
312     padding: 5px 10px;
313     border-radius: 5px;
314 }
315
316 .zoom-btn {
317     width: 30px;
318     height: 30px;
319     padding: 0;
320     font-size: 18px;
321     font-weight: bold;
322 }
323
324 #zoom-level {
325     min-width: 50px;
326     text-align: center;
327     font-size: 12px;
328     color: #aaa;
329 }
330
331 /* ===== ОСНОВНАЯ ОБЛАСТЬ ===== */
332 #main {
333     display: flex;
334     flex: 1;
335     overflow: hidden;
336 }
337
338 /* ===== ПАЛИТРА ===== */
339 #palette {
340     width: 200px;
341     background: #16213e;
342     padding: 15px;
343     border-right: 2px solid #0f3460;
344     overflow-y: auto;
345     z-index: 10;
346     flex-shrink: 0;
347 }
348
349 #palette h3 {
350     margin-bottom: 15px;
351     color: #e94560;
352     text-align: center;
353     font-size: 14px;
354 }
355
356 .palette-section {
357     margin-bottom: 15px;
358 }
359
360 .palette-section-title {
361     font-size: 11px;
362     color: #888;
363     margin-bottom: 8px;
364     padding-bottom: 3px;
365     border-bottom: 1px solid #333;
366 }
367
```

```
368 .palette-item {  
369     background: #0f3460;  
370     padding: 8px;  
371     margin-bottom: 6px;  
372     border-radius: 8px;  
373     cursor: grab;  
374     text-align: center;  
375     transition: all 0.3s;  
376     border: 2px solid transparent;  
377     user-select: none;  
378 }  
379  
380 .palette-item:hover {  
381     border-color: #e94560;  
382     transform: scale(1.02);  
383 }  
384  
385 .palette-item:active {  
386     cursor: grabbing;  
387 }  
388  
389 .palette-item svg {  
390     width: 50px;  
391     height: 32px;  
392     margin-bottom: 2px;  
393     pointer-events: none;  
394 }  
395  
396 .palette-item-name {  
397     font-size: 10px;  
398     color: #aaa;  
399     pointer-events: none;  
400 }  
401  
402 .type-legend {  
403     margin-top: 15px;  
404     padding-top: 10px;  
405     border-top: 1px solid #333;  
406     font-size: 10px;  
407 }  
408  
409 .type-legend-item {  
410     display: flex;  
411     align-items: center;  
412     gap: 8px;  
413     margin-bottom: 5px;  
414 }  
415  
416 .type-legend-dot {  
417     width: 12px;  
418     height: 12px;  
419     border-radius: 50%;  
420     border: 2px solid #fff;  
421 }  
422 .type-legend-dot.logic { background: #a855f7; }  
423 .type-legend-dot.number { background: #3b82f6; }  
424  
425 /* ===== РАБОЧАЯ ОБЛАСТЬ ===== */  
426 #workspace-container {  
427     flex: 1;  
428     position: relative;  
429     overflow: hidden;  
430     background-color: #0a0a1a;  
431     background-image:  
432         linear-gradient(rgba(255,255,255,0.04) 1px, transparent 1px),
```

```
433         linear-gradient(90deg, rgba(255,255,255,0.04) 1px, transparent 1px);  
434     background-size: 25px 25px;  
435 }  
436  
437 #workspace {  
438     position: absolute;  
439     transform-origin: 0 0;  
440     width: 5000px;  
441     height: 5000px;  
442 }  
443  
444 #connections-svg {  
445     position: absolute;  
446     transform-origin: 0 0;  
447     pointer-events: none;  
448     z-index: 5;  
449     width: 5000px;  
450     height: 5000px;  
451 }  
452  
453 #connections-svg path {  
454     pointer-events: stroke;  
455 }  
456  
457 /* ===== ЭЛЕМЕНТЫ ===== */  
458 .element {  
459     position: absolute;  
460     background: #0f3460;  
461     border: 2px solid #4a90d9;  
462     border-radius: 8px;  
463     cursor: move;  
464     user-select: none;  
465     z-index: 10;  
466     display: flex;  
467     flex-direction: column;  
468 }  
469  
470 .element.selected {  
471     border-color: #e94560;  
472     box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);  
473 }  
474  
475 .element-header {  
476     background: #4a90d9;  
477     padding: 5px 10px;  
478     border-radius: 5px 5px 0 0;  
479     font-size: 11px;  
480     font-weight: bold;  
481     text-align: center;  
482     white-space: nowrap;  
483     overflow: hidden;  
484     text-overflow: ellipsis;  
485 }  
486  
487 .element-body {  
488     padding: 10px;  
489     display: flex;  
490     justify-content: space-between;  
491     align-items: center;  
492     flex: 1;  
493     gap: 8px;  
494 }  
495  
496 .element-symbol {  
497     font-size: 16px;
```

```
498     font-weight: bold;
499     flex: 1;
500     text-align: center;
501     padding: 0 5px;
502     word-break: break-all;
503     color: #eee;
504 }
505
506 /* ===== ПОРТЫ ===== */
507 .ports-left, .ports-right {
508     display: flex;
509     flex-direction: column;
510     justify-content: space-around;
511     gap: 10px;
512     height: 100%;
513 }
514
515 .port {
516     width: 14px;
517     height: 14px;
518     border-radius: 50%;
519     border: 2px solid #fff;
520     cursor: crosshair;
521     transition: all 0.2s;
522     position: relative;
523     flex-shrink: 0;
524 }
525
526 .port:hover { transform: scale(1.3); }
527 .port.input { margin-left: -8px; }
528 .port.output { margin-right: -8px; }
529 .port.connected { background: #4ade80; }
530
531 /* Типы портов */
532 .port.logic-port { background: #a855f7; border-color: #e9d5ff; }
533 .port.logic-port:hover { background: #c084fc; }
534 .port.logic-port.connected { background: #7c3aed; }
535
536 .port.number-port { background: #3b82f6; border-color: #bfdbfe; }
537 .port.number-port:hover { background: #60a5fa; }
538 .port.number-port.connected { background: #2563eb; }
539
540 .port.any-port { background: #6b7280; border-color: #d1d5db; }
541 .port.any-port:hover { background: #9ca3af; }
542 .port.any-port.connected { background: #4b5563; }
543
544 .port.output.yes-port { background: #4ade80 !important; border-color: #bbf7d0 !important; }
545 .port.output.no-port { background: #f87171 !important; border-color: #fecaca !important; }
546
547 .port.incompatible { opacity: 0.3; cursor: not-allowed; }
548 .port.compatible-highlight { box-shadow: 0 0 10px 3px #4ade80; }
549
550 /* ===== RESIZE HANDLES ===== */
551 .resize-handle {
552     position: absolute;
553     width: 12px;
554     height: 12px;
555     background: #e94560;
556     border: 1px solid #fff;
557     border-radius: 3px;
558     z-index: 20;
559     opacity: 0;
560     transition: opacity 0.2s;
```

```
561 }
562 .element.selected .resize-handle { opacity: 0.8; }
563 .resize-handle:hover { opacity: 1; }
564 .resize-handle.handle-se { bottom: -6px; right: -6px; cursor: se-resize; }
565 .resize-handle.handle-e { top: 50%; right: -6px; transform: translateY(-50%); cursor: ew-resize; }
566 .resize-handle.handle-s { bottom: -6px; left: 50%; transform: translateX(-50%); cursor: ns-resize; }
567
568
569 /* ===== ВХОДНОЙ СИГНАЛ (ТРАПЕЦИЯ) ===== */
570 .element.input-signal {
571     background: transparent;
572     border: none;
573 }
574
575 .element.input-signal .element-header {
576     display: none; /* У трапеции нет заголовка */
577 }
578
579 .element.input-signal .element-body {
580     padding: 0;
581     background: #0f3460;
582     border: 2px solid #4a90d9;
583     clip-path: polygon(0 0, 80% 0, 100% 50%, 80% 100%, 0 100%);
584     display: flex;
585     justify-content: space-between;
586     align-items: center;
587     padding-left: 15px;
588     padding-right: 25px;
589 }
590
591 .element.input-signal .element-symbol {
592     text-align: left;
593     color: #eee;
594 }
595
596 .element.input-signal.selected .element-body {
597     border-color: #e94560;
598 }
599
600 /* ===== ЭЛЕМЕНТ ВЫХОДА (ПУНКТИР) ===== */
601 .element.output {
602     background: rgba(16, 185, 129, 0.1);
603     border: 2px dashed #10b981;
604 }
605
606 .element.output .element-header {
607     display: none; /* У выхода нет заголовка */
608 }
609
610 .element.output .element-body {
611     padding-left: 20px;
612 }
613
614 .element.output .element-symbol {
615     color: #10b981;
616     font-size: 14px;
617 }
618
619 .element.output.selected {
620     border-color: #e94560;
621     border-style: dashed;
622 }
623
```

```
624
625 /* Formula condition port */
626 /* Универсальный стиль для технического порта (сверху) */
627 .element.has-condition-port {
628     margin-top: 30px; /* Даем место порту над элементом */
629 }
630
631 .condition-port-wrapper {
632     position: absolute;
633     top: -28px;
634     left: 50%;
635     transform: translateX(-50%);
636     display: flex;
637     flex-direction: column;
638     align-items: center;
639     gap: 4px;
640     pointer-events: none;
641     z-index: 21;
642 }
643
644 .condition-port-label {
645     font-size: 10px;
646     color: #f59e0b;
647     font-weight: 600;
648     white-space: nowrap;
649 }
650
651 .port.condition-port {
652     pointer-events: auto;
653     width: 16px;
654     height: 16px;
655     border-radius: 50%;
656     border: 2px solid #f59e0b;
657     background: #fff7ed;
658     margin: 0; /* Сбрасываем лишние отступы */
659 }
660 .element.formula .condition-port:hover { background: #fde68a; }
661
662
663 /* ===== СОЕДИНЕНИЯ ===== */
664 .connection {
665     fill: none !important; /* ← добавляем !important */
666     stroke: #4a90d9;
667     stroke-width: 2.5;
668 }
669 .connection:hover {
670     stroke: #e94560;
671     stroke-width: 4;
672 }
673
674 .connection.logic-conn { stroke: #a855f7; }
675 .connection.numeric-conn { stroke: #3b82f6; }
676 .connection.any-conn { stroke: #6b7280; }
677 .connection.true-conn { stroke: #4ade80; }
678 .connection.false-conn { stroke: #f87171; }
679
680 .connection.yes-conn { stroke: #4ade80; }
681 .connection.no-conn { stroke: #f87171; }
682
683 .temp-connection {
684     fill: none !important; /* ← добавляем !important */
685     stroke: #e94560;
686     stroke-width: 2;
687     stroke-dasharray: 5, 5;
688 }
```

```
689 .temp-connection.invalid { stroke: #ef4444; }
690
691 /* ===== ПРОЧЕЕ ===== */
692 .drag-preview {
693     position: fixed;
694     pointer-events: none;
695     opacity: 0.8;
696     z-index: 1000;
697     background: #0f3460;
698     border: 2px solid #e94560;
699     border-radius: 8px;
700     padding: 10px 15px;
701     color: #fff;
702     font-size: 12px;
703 }
704
705 #minimap {
706     position: absolute;
707     bottom: 20px;
708     right: 20px;
709     width: 200px;
710     height: 150px;
711     background: #16213e;
712     border: 2px solid #0f3460;
713     border-radius: 8px;
714     overflow: hidden;
715     z-index: 50;
716 }
717
718 #minimap-canvas { width: 100%; height: 100%; }
719 #minimap-viewport {
720     position: absolute;
721     border: 2px solid #e94560;
722     background: rgba(233, 69, 96, 0.2);
723     pointer-events: none;
724 }
725
726 #viewport-info {
727     position: absolute;
728     bottom: 20px;
729     left: 20px;
730     background: rgba(22, 33, 62, 0.9);
731     padding: 8px 12px;
732     border-radius: 5px;
733     font-size: 11px;
734     color: #888;
735     z-index: 50;
736     display: flex;
737     gap: 15px;
738 }
739 #selection-info { color: #e94560; }
740
741 #modal-overlay, .modal-overlay-class {
742     display: none;
743     position: fixed;
744     top: 0; left: 0;
745     width: 100%; height: 100%;
746     background: rgba(0, 0, 0, 0.7);
747     z-index: 1000;
748     justify-content: center;
749     align-items: center;
750 }
751
752 #modal, .modal-class {
753     background: #16213e;
```

```
754     border-radius: 10px;
755     padding: 20px;
756     min-width: 400px;
757     max-width: 600px;
758     max-height: 80vh;
759     overflow-y: auto;
760     border: 2px solid #0f3460;
761 }
762
763 #modal h3, .modal-class h3 { margin-bottom: 15px; color: #e94560; }
764 .modal-row { margin-bottom: 15px; }
765 .modal-row label { display: block; margin-bottom: 5px; color: #aaa; font-size: 13px; }
766 .modal-row input, .modal-row select, .modal-row textarea {
767     width: 100%;
768     padding: 10px;
769     background: #0f3460;
770     border: 1px solid #4a90d9;
771     border-radius: 5px;
772     color: #eee;
773     font-size: 14px;
774 }
775 .modal-row input:focus, .modal-row select:focus, .modal-row textarea:focus { outline: none; border-color: #e94560; }
776 .modal-row textarea { min-height: 80px; font-family: inherit; resize: vertical; }
777 .signal-list { max-height: 100px; overflow-y: auto; background: #0f3460; border-radius: 5px; padding: 5px; margin-top: 5px; }
778 .signal-item { padding: 5px 10px; cursor: pointer; border-radius: 3px; font-size: 12px; }
779 .signal-item:hover { background: #4a90d9; }
780 .modal-buttons { display: flex; gap: 10px; justify-content: flex-end; margin-top: 20px; }
781 .modal-btn { padding: 10px 25px; border: none; border-radius: 5px; cursor: pointer; font-size: 14px; transition: background 0.3s; }
782 .modal-btn.save { background: #4ade80; color: #000; }
783 .modal-btn.save:hover { background: #22c55e; }
784 .modal-btn.cancel { background: #6b7280; color: #fff; }
785 .modal-btn.cancel:hover { background: #4b5563; }
786
787 #context-menu {
788     display: none;
789     position: fixed;
790     background: #16213e;
791     border: 1px solid #0f3460;
792     border-radius: 5px;
793     padding: 5px 0;
794     z-index: 1001;
795     min-width: 150px;
796     box-shadow: 0 5px 20px rgba(0,0,0,0.3);
797 }
798 .context-item { padding: 10px 15px; cursor: pointer; font-size: 13px; transition: background 0.2s; }
799 .context-item:hover { background: #0f3460; }
800
801 #file-input { display: none; }
802
803 .project-type-selector { display: flex; gap: 10px; margin-bottom: 15px; }
804 .project-type-btn { flex: 1; padding: 15px; background: #0f3460; border: 2px solid #4a90d9; border-radius: 8px; color: #eee; cursor: pointer; text-align: center; transition: all 0.3s; }
805 .project-type-btn:hover { border-color: #e94560; }
806 .project-type-btn.active { background: #4a90d9; border-color: #4a90d9; }
807 .project-type-btn .type-icon { font-size: 24px; margin-bottom: 5px; }
808 .project-type-btn .type-name { font-weight: bold; }
809 .project-type-btn .type-desc { font-size: 11px; color: #aaa; margin-top: 3px; }
810
```

```
811 .conditional-fields { display: none; padding: 15px; background: #0a0a1a; border-radius: 8px; margin-top: 10px; }
812 .conditional-fields.visible { display: block; }
813
814 ::-webkit-scrollbar { width: 8px; height: 8px; }
815 ::-webkit-scrollbar-track { background: #0a0a1a; }
816 ::-webkit-scrollbar-thumb { background: #4a90d9; border-radius: 4px; }
817 ::-webkit-scrollbar-thumb:hover { background: #e94560; }
818
819 /* Стили для выходов */
820 .output-btn { position: relative; }
821 .output-counter { display: inline-block; background: #e94560; color: white; font-size: 11px; font-weight: bold; padding: 2px 6px; border-radius: 10px; margin-left: 5px; min-width: 18px; text-align: center; }
822 .output-counter:empty, .output-counter[style*="display: none"] { display: none; }
823 .element.has-output { box-shadow: 0 0 10px rgba(16, 185, 129, 0.3); }
824 .element.output-highlighted { box-shadow: 0 0 20px rgba(251, 191, 36, 0.6) !important; border-color: #fbbf24 !important; }
825 .port.output-active { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); animation: pulse-output 1.5s infinite; }
826 @keyframes pulse-output {
827     0%, 100% { box-shadow: 0 0 8px 2px rgba(16, 185, 129, 0.8); }
828     50% { box-shadow: 0 0 12px 4px rgba(16, 185, 129, 1); }
829 }
830
831 .outputs-container { background: #0a0a1a; border-radius: 8px; padding: 15px; max-height: 250px; overflow-y: auto; }
832 .outputs-section { margin-bottom: 15px; }
833 .outputs-section:last-child { margin-bottom: 0; }
834 .outputs-section-title { color: #10b981; font-weight: bold; font-size: 13px; margin-bottom: 10px; padding-bottom: 5px; border-bottom: 1px solid #333; display: flex; align-items: center; gap: 8px; }
835 .outputs-section-title .section-icon { font-size: 16px; }
836 .outputs-list { display: flex; flex-direction: column; gap: 5px; }
837 .output-item { display: flex; align-items: center; gap: 10px; padding: 8px 12px; background: rgba(16, 185, 129, 0.1); border: 1px solid rgba(16, 185, 129, 0.3); border-radius: 5px; cursor: pointer; transition: all 0.2s; }
838 .output-item:hover { background: rgba(16, 185, 129, 0.2); border-color: #10b981; transform: translateX(5px); }
839 .output-item.numeric { background: rgba(59, 130, 246, 0.1); border-color: rgba(59, 130, 246, 0.3); }
840 .output-item.numeric:hover { background: rgba(59, 130, 246, 0.2); border-color: #3b82f6; }
841 .output-icon { font-size: 14px; }
842 .output-name { font-weight: bold; color: #eee; }
843 .output-port { color: #888; font-size: 12px; margin-left: auto; }
844 .no-outputs { color: #666; font-style: italic; padding: 10px; text-align: center; }
845 .outputs-hint { margin-top: 10px; padding: 10px; background: rgba(59, 130, 246, 0.1); border-radius: 5px; font-size: 12px; color: #888; line-height: 1.4; }
846 .element.output-ambiguous { box-shadow: 0 0 18px 4px rgba(240, 80, 80, 0.55); border-color: rgba(240, 80, 80, 0.8) !important; }
847 .element.output-missing { box-shadow: 0 0 14px 3px rgba(250, 200, 30, 0.5); border-color: rgba(250, 200, 30, 0.8) !important; }
848 /* TRUE/FALSE порты (для сепаратора) */
849 .port.true-port {
850     background: #4ade80 !important;
851     border-color: #bbf7d0 !important;
852 }
853 .port.true-port:hover {
854     background: #22c55e !important;
855 }
856 .port.true-port.connected {
857     background: #16a34a !important;
858 }
859 }
```

```
860 .port.false-port {  
861     background: #f87171 !important;  
862     border-color: #fecaca !important;  
863 }  
864 .port.false-port:hover {  
865     background: #ef4444 !important;  
866 }  
867 .port.false-port.connected {  
868     background: #dc2626 !important;  
869 }  
870  
871 /* Сепаратор стиль */  
872 .element.separator {  
873     background: #0f3460;  
874     border: 2px solid #f59e0b;  
875 }  
876  
877 .element.separator.selected {  
878     border-color: #e94560;  
879     box-shadow: 0 0 15px rgba(233, 69, 96, 0.5);  
880 }  
881  
882 /* === Выделение рамкой === */  
883 #selection-rect {  
884     position: absolute;  
885     border: 1px dashed #e94560;  
886     background: rgba(233, 69, 96, 0.1);  
887     pointer-events: none;  
888     display: none;  
889     z-index: 200;  
890 }  
891  
892 /* === Кастомный элемент “Группа” === */  
893 .element.group {  
894     background: rgba(107, 114, 128, 0.15);  
895     border: 2px dashed #6b7280;  
896     border-radius: 8px;  
897     pointer-events: none; /* не мешает клику по внутренним элементам */  
898     position: absolute;  
899 }  
900  
901 .group-title {  
902     position: absolute;  
903     top: -20px;  
904     left: 5px;  
905     font-size: 11px;  
906     color: #ccc;  
907     background: #16213e;  
908     padding: 2px 6px;  
909     border-radius: 4px;  
910     pointer-events: auto; /* можно кликнуть для выбора */  
911 }  
912  
913 .modal.hidden { display: none; }  
914 .modal { position: fixed; inset: 0; display: flex; align-items: center; justify-content: center; background: rgba(0,0,0,0.4); z-index: 1000; }  
915 .modal__content { background: #fff; padding: 24px; border-radius: 8px; width: 640px; max-height: 80vh; display: flex; flex-direction: column; gap: 16px; overflow: hidden; }  
916 .modal__content--wide { width: 800px; }  
917 .modal__title { margin: 0; }  
918  
919 .project-list__toolbar { display: flex; gap: 12px; }  
920 .project-list__toolbar input { flex: 1; padding: 6px 10px; }  
921 .project-list__table-container { flex: 1; overflow: auto; border: 1px solid #ddd;
```

```
border-radius: 6px; }
922 .project-list__table { width: 100%; border-collapse: collapse; }
923 .project-list__table th, .project-list__table td { padding: 8px 12px; border-bottom: 1px solid #eee; }
924 .project-list__table tbody tr { cursor: pointer; transition: background 0.15s ease; }
925 .project-list__table tbody tr:hover { background: #f0f6ff; }
926 .project-list__empty { text-align: center; color: #888; padding: 16px; }
927 .modal__actions { display: flex; justify-content: flex-end; gap: 12px; }
928 .project-list__table th,
929 .project-list__table td {
930   color: #111; /* насыщенный чёрный текст */
931   padding: 8px 12px;
932   border-bottom: 1px solid #eee;
933 }
934 .modal__content--wide {
935   width: 860px;
936   max-height: 90vh; /* занимает 90% экрана */
937 }
938
939 .project-list__table-container {
940   flex: 1;
941   overflow: auto;
942   border: 1px solid #ddd;
943   border-radius: 6px;
944   max-height: 60vh; /* много строк */
945 }
946 app.js:
947 /**
948 * Главный модуль приложения
949 */
950
951 const App = {
952   /**
953    * Инициализация приложения
954    */
955   init() {
956     //Settings.init().then(() => {
957     //  // если хочешь – можно обновить UI (например, статус “Сигналы
загружены”)
958     //  console.log('Settings loaded, signals:', Settings.signals.length);
959     //  }).catch(err => console.error(err));
960     //console.log('signals loaded:', Settings.signals.slice(0, 5));
961     this.setupPaletteDragDrop();
962     this.setupGlobalMouseHandlers();
963     this.setupContextMenu();
964     this.setupWorkspaceClick();
965     this.setupOutputCounter();
966     this.setupMultiSelection();
967
968     // Инициализация модулей
969     Viewport.init();
970     Modal.init();
971     Project.init();
972
973     // Первоначальное определение выходов (только если модуль загружен)
974     if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
975       Outputs.updateOutputStatus();
976     }
977
978     console.log('Logic Scheme Editor initialized');
979     document.getElementById('btn-generate-code').addEventListener('click', () => {
980       const code = CodeGen.generate();
981       document.getElementById('code-output').value = code;
982       document.getElementById('code-modal-overlay').style.display = 'flex';
983     });
984 }
```

```
984     document.getElementById('code-modal-close').addEventListener('click', () => {
985         document.getElementById('code-modal-overlay').style.display = 'none';
986     });
987 },
988 /**
989 * Отмена состояния drag из палитры (helper)
990 */
991 cancelPaletteDrag() {
992     if (AppState.dragPreview) {
993         try { AppState.dragPreview.remove(); } catch (e) { /* ignore */ }
994         AppState.dragPreview = null;
995     }
996     AppState.isDraggingFromPalette = false;
997     AppState.dragType = null;
998 },
999
1000 /**
1001 * Настройка счётчика выходов в меню
1002 */
1003 setupOutputCounter() {
1004     // Не создавать повторно, если уже есть
1005     if (document.getElementById('btn-outputs')) return;
1006
1007     const menu = document.getElementById('menu');
1008
1009     // Создаём кнопку с счётчиком выходов
1010     const outputBtn = document.createElement('button');
1011     outputBtn.className = 'menu-btn output-btn';
1012     outputBtn.id = 'btn-outputs';
1013     outputBtn.innerHTML =
1014         `  Выходы
1015         <span id="output-counter" class="output-counter">0</span>
1016     `;
1017
1018     // Вставляем после кнопки свойств проекта
1019     const projectBtn = document.getElementById('btn-project-settings');
1020     if (projectBtn) {
1021         projectBtn.after(outputBtn);
1022     } else {
1023         menu.appendChild(outputBtn);
1024     }
1025
1026     outputBtn.addEventListener('click', () => {
1027         Modal.showProjectPropertiesModal();
1028     });
1029 },
1030
1031 /**
1032 * Настройка drag & drop из палитры
1033 */
1034 setupPaletteDragDrop() {
1035     document.querySelectorAll('.palette-item').forEach(item => {
1036         item.addEventListener('mousedown', (e) => {
1037             // Только левая кнопка мыши должна запускать drag из палитры
1038             if (e.button !== 0) return;
1039             e.preventDefault();
1040
1041             AppState.isDraggingFromPalette = true;
1042             AppState.dragType = item.dataset.type;
1043
1044             AppState.dragPreview = document.createElement('div');
1045             AppState.dragPreview.className = 'drag-preview';
1046             AppState.dragPreview.textContent =
1047
1048 }
```

```
ELEMENT_TYPES[AppState.dragType]?.name || 'Элемент';
1049                     AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1050                     AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1051                     document.body.appendChild(AppState.dragPreview);
1052                 });
1053             });
1054         },
1055     /**
1056      * Глобальные обработчики мыши
1057      */
1058 /**
1059  * Глобальные обработчики мыши
1060 */
1061
1062 setupGlobalMouseHandlers() {
1063     document.addEventListener('mousemove', (e) => {
1064         if (AppState.isDraggingFromPalette && AppState.dragPreview) {
1065             AppState.dragPreview.style.left = `${e.clientX - 40}px`;
1066             AppState.dragPreview.style.top = `${e.clientY - 20}px`;
1067         }
1068         if (AppState.resizing) {
1069             Elements.handleResize(e);
1070             return;
1071         }
1072         if (AppState.draggingElement) {
1073             Elements.handleDrag(e);
1074         }
1075         if (AppState.tempLine && AppState.connectingFrom) {
1076             Connections.drawTempConnection(e);
1077         }
1078     });
1079
1080     document.addEventListener('mouseup', (e) => {
1081         if (AppState.resizing) {
1082             AppState.resizing = null;
1083             if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1084         }
1085
1086         if (AppState.isDraggingFromPalette) {
1087             try {
1088                 if (AppState.dragPreview) {
1089                     AppState.dragPreview.remove();
1090                     AppState.dragPreview = null;
1091                 }
1092
1093                 const container = document.getElementById('workspace-container');
1094                 const rect = container.getBoundingClientRect();
1095
1096                 if (e.clientX >= rect.left && e.clientX <= rect.right &&
1097                     e.clientY >= rect.top && e.clientY <= rect.bottom) {
1098
1099                     const canvasPos = screenToCanvas(e.clientX, e.clientY);
1100                     const config = ELEMENT_TYPES[AppState.dragType];
1101                     if (config) {
1102                         const defaultWidth = config.minLength || 120;
1103                         const defaultHeight = config.minLength || 60;
1104
1105                         // ИСПРАВЛЕНО: addElement возвращает DOM-элемент, его надо
1106                         // обработать
1107                         const newElement = Elements.addElement(
1108                             AppState.dragType,
1109                             canvasPos.x - defaultWidth / 2,
1110                             canvasPos.y - defaultHeight / 2
1111                         );
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
```

```
1112             if (newElement && typeof Outputs !== 'undefined') {
1113                 Outputs.updateOutputStatus();
1114             }
1115         } else {
1116             console.error('Неизвестный тип элемента при drop:',
1117             AppState.dragType);
1118         }
1119     } finally {
1120         App.cancelPaletteDrag();
1121     }
1122 }
1123
1124 if (AppState.draggingElement) {
1125     AppState.draggingElement = null;
1126 }
1127
1128 Connections.clearConnectionState();
1129 });
1130
1131 document.addEventListener('keydown', (e) => {
1132     if (e.key === 'Delete' && AppState.selectedElement) {
1133         Elements.deleteElement(AppState.selectedElement);
1134         if (typeof Outputs !== 'undefined') Outputs.updateOutputStatus();
1135     }
1136     if (e.key === 'Escape') {
1137         Elements.deselectAll();
1138         Connections.clearConnectionState();
1139         if (AppState.isDraggingFromPalette) App.cancelPaletteDrag();
1140     }
1141 });
1142 },
1143
1144 /**
1145 * Настройка контекстного меню
1146 */
1147 setupContextMenu() {
1148     document.addEventListener('click', (e) => {
1149         const menu = document.getElementById('context-menu');
1150         if (!menu.contains(e.target)) {
1151             menu.style.display = 'none';
1152         }
1153     });
1154
1155     document.getElementById('ctx-properties').addEventListener('click', () => {
1156         const elemId = document.getElementById('context-menu').dataset.elementId;
1157         document.getElementById('context-menu').style.display = 'none';
1158         const config = ELEMENT_TYPES[AppState.elements[elemId]?.type];
1159         if (config?.hasProperties) {
1160             Modal.showPropertiesModal(elemId);
1161         }
1162     });
1163
1164     document.getElementById('ctx-delete').addEventListener('click', () => {
1165         const elemId = document.getElementById('context-menu').dataset.elementId;
1166         document.getElementById('context-menu').style.display = 'none';
1167         Elements.deleteElement(elemId);
1168         // Обновляем выходы только если модуль загружен
1169         if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
1170             Outputs.updateOutputStatus();
1171         }
1172     });
1173 },
1174
1175 /**
```

```
1176     * Клик по рабочей области
1177     */
1178     setupWorkspaceClick() {
1179         const workspace = document.getElementById('workspace');
1180
1181         workspace.addEventListener('click', (e) => {
1182             if (e.target === workspace) {
1183                 Elements.deselectAll();
1184             }
1185         });
1186     },
1187     /**
1188     * --- Выделение рамкой и множественное перемещение ---
1189     */
1190     setupMultiSelection() {
1191         const container = document.getElementById('workspace-container');
1192         const rectEl = document.getElementById('selection-rect');
1193
1194         container.addEventListener('mousedown', (e) => {
1195             if (e.button !== 0) return;
1196             if (e.target !== document.getElementById('workspace')) return;
1197
1198             const pos = screenToCanvas(e.clientX, e.clientY);
1199             AppState.multiSelecting = true;
1200             AppState.selectionRect = { startX: pos.x, startY: pos.y, x: pos.x, y:
pos.y, w: 0, h: 0 };
1201
1202             rectEl.style.left = e.clientX + 'px';
1203             rectEl.style.top = e.clientY + 'px';
1204             rectEl.style.width = '0px';
1205             rectEl.style.height = '0px';
1206             rectEl.style.display = 'block';
1207         });
1208
1209         document.addEventListener('mousemove', (e) => {
1210             if (!AppState.multiSelecting) return;
1211
1212             const pos = screenToCanvas(e.clientX, e.clientY);
1213             const sx = AppState.selectionRect.startX;
1214             const sy = AppState.selectionRect.startY;
1215             const x = Math.min(sx, pos.x);
1216             const y = Math.min(sy, pos.y);
1217             const w = Math.abs(pos.x - sx);
1218             const h = Math.abs(pos.y - sy);
1219
1220             rectEl.style.left = x * AppState.viewport.zoom + AppState.viewport.panX +
'px';
1221             rectEl.style.top = y * AppState.viewport.zoom + AppState.viewport.panY +
'px';
1222             rectEl.style.width = w * AppState.viewport.zoom + 'px';
1223             rectEl.style.height = h * AppState.viewport.zoom + 'px';
1224
1225             const selected = [];
1226             for (const [id, elData] of Object.entries(AppState.elements)) {
1227                 if (!elData || elData.type === 'output-frame') continue;
1228                 if (
1229                     elData.x >= x && elData.x + elData.width <= x + w &&
1230                     elData.y >= y && elData.y + elData.height <= y + h
1231                 ) selected.push(id);
1232             }
1233
1234             AppState.selectedElements = selected;
1235             document.querySelectorAll('.element').forEach(el =>
1236                 el.classList.toggle('selected', selected.includes(el.id))
1237             );
}
```

```
1238     });
1239
1240     document.addEventListener('mouseup', () => {
1241         if (AppState.multiSelecting) {
1242             AppState.multiSelecting = false;
1243             rectEl.style.display = 'none';
1244         }
1245     });
1246 };
1247 };
1248
1249 // Запуск приложения при загрузке страницы
1250 document.addEventListener('DOMContentLoaded', () => {
1251     App.init();
1252 });
1253
1254 codegen_graph.js:
1255 // js/codegen_graph.js
1256
1257 const CodeGenGraph = {
1258     /**
1259      * Собрать все условия вверх по цепочке cond-портов (до корня).
1260      * Возвращает null или объединённое через AND условие.
1261      */
1262     /**
1263      * Собрать ВСЕ условия: и через cond-порты, и через контекст обычных входов
1264      */
1265     collectAllCond(graph) {
1266         if (!graph) return null;
1267
1268         let c = null;
1269         const elem = graph.elem;
1270
1271         // 1. Собираем условия через cond-порт (как было)
1272         if (graph.condInput) {
1273             const condConn = graph.condInput.conn;
1274             const fromGraph = graph.condInput.fromGraph;
1275             const oneCond = this.evalConditionFromPort(fromGraph, condConn.fromPort);
1276             c = oneCond;
1277
1278             // Рекурсивно идём вверх по cond-цепочке
1279             const upCond = this.collectAllCond(fromGraph);
1280             if (upCond) {
1281                 c = c ? Optimizer.And(c, upCond) : upCond;
1282             }
1283         }
1284
1285         // 2. НОВОЕ: если это separator – учитываем контекст его входа
1286         if (elem.type === 'separator' && graph.inputs.length > 0) {
1287             const inputGraph = graph.inputs[0].fromGraph;
1288             const inputContext = this.collectAllCond(inputGraph);
1289             if (inputContext) {
1290                 c = c ? Optimizer.And(c, inputContext) : inputContext;
1291             }
1292         }
1293
1294         return c;
1295     },
1296     buildDependencyGraph(elementId) {
1297         const graph = {
1298             nodeId: elementId,
1299             elem: AppState.elements[elementId],
1300             inputs: [],
1301             condInput: null,
1302         };
1303     }
1304 }
```

```
1303
1304     if (!graph.elem) return null;
1305
1306     const inConns = AppState.connections.filter(c =>
1307         c.toElement === elementId && c.toPort.startsWith('in-')
1308     );
1309     inConns.forEach(conn => {
1310         graph.inputs.push({
1311             conn,
1312             fromGraph: this.buildDependencyGraph(conn.fromElement)
1313         });
1314     });
1315
1316     const condConn = AppState.connections.find(c =>
1317         c.toElement === elementId && c.toPort === 'cond-0'
1318     );
1319     if (condConn) {
1320         graph.condInput = {
1321             conn: condConn,
1322             fromGraph: this.buildDependencyGraph(condConn.fromElement)
1323         };
1324     }
1325
1326     return graph;
1327 },
1328 /**
1329 * Получить ЛОГИКУ из графа (для IF/AND/OR/NOT/SEPARATOR)
1330 */
1331 evalLogic(graph) {
1332     if (!graph) return Optimizer.TrueCond;
1333     const elem = graph.elem;
1334
1335     switch (elem.type) {
1336         case 'if':
1337             const left = graph.inputs[0]?.fromGraph;
1338             const right = graph.inputs[1]?.fromGraph;
1339
1340             const leftVal = left ? this.evalValue(left) : Optimizer.Const(0);
1341             const rightVal = right ? this.evalValue(right) : Optimizer.Const(0);
1342
1343             const op = elem.props.operator || '=';
1344             return this.buildIfLogic(leftVal, op, rightVal);
1345         }
1346
1347         case 'and':
1348             let result = null;
1349             for (const inp of graph.inputs) {
1350                 const inLogic = this.evalLogic(inp.fromGraph);
1351                 result = result ? Optimizer.And(result, inLogic) : inLogic;
1352             }
1353             return result || Optimizer.TrueCond;
1354         }
1355
1356         case 'or':
1357             let result = null;
1358             for (const inp of graph.inputs) {
1359                 const inLogic = this.evalLogic(inp.fromGraph);
1360                 result = result ? Optimizer.Or(result, inLogic) : inLogic;
1361             }
1362             return result || Optimizer.FalseCond;
1363         }
1364
1365         case 'not':
1366             const inLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
```

```
1368         return Optimizer.Not(inLogic);
1369     }
1370
1371     case 'separator': {
1372         return this.evalLogic(graph.inputs[0]?.fromGraph);
1373     }
1374
1375     default:
1376         return Optimizer.TrueCond;
1377     }
1378 },
1379
1380 /**
1381 * Получить ЗНАЧЕНИЕ из графа (для INPUT/CONST/FORMULA)
1382 */
1383 evalValue(graph) {
1384     if (!graph) return Optimizer.Const(0);
1385     const elem = graph.elem;
1386
1387     switch (elem.type) {
1388         case 'input-signal':
1389             return Optimizer.Var(elem.props.name || graph.nodeId);
1390
1391         case 'const':
1392             return Optimizer.Const(Number(elem.props.value) || 0);
1393
1394         case 'formula': {
1395             const expr = this.buildFormulaExpr(elem);
1396             return Optimizer.Var(expr);
1397         }
1398
1399         case 'separator':
1400             return this.evalValue(graph.inputs[0]?.fromGraph);
1401
1402         default:
1403             return Optimizer.Const(0);
1404     }
1405 },
1406
1407 // js/codegen_graph.js
1408
1409 /**
1410 * Рекурсивно собрать полный контекст условий для элемента
1411 * через всю цепочку cond-портов вверх
1412 */
1413 // В codegen_graph.js, в evalFullContext добавь:
1414
1415 evalFullContext(graph) {
1416     if (!graph) return null;
1417
1418     let context = null;
1419     const elem = graph.elem;
1420
1421     console.log(`evalFullContext для ${elem.id} (${elem.type})`);
1422
1423     // 1. Если сам элемент имеет cond-порт – собираем его условие
1424     if (graph.condInput) {
1425         const condConn = graph.condInput.conn;
1426         console.log(` → имеет cond-0 от ${graph.condInput.fromGraph.elem.id}.${condConn.fromPort}`);
1427
1428         const condLogic = this.evalConditionFromPort(
1429             graph.condInput.fromGraph,
1430             condConn.fromPort
1431         );
1432     }
1433 }
```

```
1432         console.log(` → условие от cond-0: ${Optimizer.printCond(condLogic)} `);
1433         context = condLogic;
1434 
1435         // 2. Рекурсивно собираем контекст элемента, на который указывает cond-
1436         // порт
1437         const upstreamContext = this.evalFullContext(graph.condInput.fromGraph);
1438         if (upstreamContext) {
1439             console.log(` → upstreamContext: ${Optimizer.printCond(upstreamContext)} `);
1440             context = context ? Optimizer.And(context, upstreamContext) :
1441             upstreamContext;
1442         }
1443         } else {
1444             console.log(` → нет cond-0 `);
1445         }
1446 
1447         console.log(` → итоговый контекст: ${Optimizer.printCond(context)} `);
1448         return context;
1449     },
1450 
1451     /**
1452      * Получить УСЛОВИЕ для cond-порта элемента
1453      * Учитывает цепочку сепараторов с TRUE/FALSE ветвлением
1454      */
1455     evalConditionFromPort(graph, fromPort) {
1456         if (!graph) return null;
1457         const elem = graph.elem;
1458 
1459         // Если это сепаратор – вычисляем его выход и применяем ветвление
1460         if (elem.type === 'separator') {
1461             const inputLogic = this.evalLogic(graph.inputs[0]?.fromGraph);
1462 
1463             if (fromPort === 'out-0') {
1464                 return inputLogic;
1465             } else if (fromPort === 'out-1') {
1466                 return Optimizer.Not(inputLogic);
1467             }
1468         }
1469 
1470         // Если это логический элемент (AND/OR/NOT/IF) – просто вычисляем логику
1471         if (elem.type === 'and' || elem.type === 'or' || elem.type === 'not' ||
1472             elem.type === 'if') {
1473             return this.evalLogic(graph);
1474         }
1475 
1476         return null;
1477     },
1478 
1479     /**
1480      * Главная функция: получить {cond, expr} для элемента
1481      */
1482     evalGraphValue(graph) {
1483 
1484         if (!graph) return { cond: null, expr: Optimizer.Const(0) };
1485 
1486         const elem = graph.elem;
1487         //let cond = null;
1488 
1489         // ← НОБОЕ: собираем полный контекст через цепочку cond-портов
1490         let cond = this.collectAllCond(graph);
1491 
1492         let expr = null;
1493 
1494         switch (elem.type) {
1495             case 'input-signal':
```

```
1493         expr = Optimizer.Var(elem.props.name || graph.nodeId);
1494         break;
1495
1496     case 'const':
1497         expr = Optimizer.Const(Number(elem.props.value) || 0);
1498         break;
1499
1500     case 'formula': {
1501         // Для формулы также собираем условия от всех входных элементов
1502         const inputConds = graph.inputs.map(inp => {
1503             const inResult = this.evalGraphValue(inp.fromGraph);
1504             return inResult.cond;
1505         }).filter(c => c);
1506
1507         // Объединяем cond-порт с условиями от входов
1508         for (const inCond of inputConds) {
1509             cond = cond ? Optimizer.And(cond, inCond) : inCond;
1510         }
1511
1512         expr = Optimizer.Var(this.buildFormulaExpr(elem));
1513         break;
1514     }
1515
1516     case 'separator':
1517         // Сепаратор – просто пробрасываем значение дальше
1518         return this.evalGraphValue(graph.inputs[0]?.fromGraph);
1519
1520         // Логические элементы не должны здесь быть
1521         case 'and':
1522         case 'or':
1523         case 'not':
1524         case 'if':
1525         default:
1526             expr = Optimizer.Const(0);
1527         }
1528
1529         return { cond, expr };
1530     },
1531
1532     buildIfLogic(leftVal, op, rightVal) {
1533         const leftName = leftVal.type === 'var' ? leftVal.name : String(leftVal.n);
1534         const rightName = rightVal.type === 'var' ? rightVal.name :
1535 String(rightVal.n);
1536
1537         const leftZero = leftVal.type === 'const' && leftVal.n === 0;
1538         const rightZero = rightVal.type === 'const' && rightVal.n === 0;
1539
1540         switch (op) {
1541             case '=':
1542                 if (rightZero) return Optimizer.Eq0(leftName);
1543                 if (leftZero) return Optimizer.Eq0(rightName);
1544                 return Optimizer.Cmp(leftName, '=', rightName);
1545             case '!=':
1546                 if (rightZero) return Optimizer.Ne0(leftName);
1547                 if (leftZero) return Optimizer.Ne0(rightName);
1548                 return Optimizer.Cmp(leftName, '!=', rightName);
1549             case '>':
1550             case '<':
1551             case '>=':
1552                 return Optimizer.Cmp(leftName, op, rightName);
1553             default:
1554                 return Optimizer.TrueCond;
1555         }
1556     },
1557 }
```

```
1557
1558     buildFormulaExpr(elem) {
1559         let result = elem.props.expression || '0';
1560         const formulaRefs = result.match(/formula-\d+/g) || [];
1561
1562         for (const ref of formulaRefs) {
1563             const refElem = AppState.elements[ref];
1564             if (refElem && refElem.type === 'formula') {
1565                 const refExpr = this.buildFormulaExpr(refElem);
1566                 result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
1567             }
1568         }
1569
1570         return result;
1571     }
1572 };
1573
1574 windowCodeGenGraph = CodeGenGraph;
1575
1576 codegen_optimizer.js:
1577
1578 // js/codegen_optimizer.js
1579
1580 let _depth = 0;
1581 const MAX_DEPTH = 200;
1582
1583 // === Конструкторы ===
1584 function Eq0(v) { return { kind: 'cond', type: 'eq0', v }; }
1585 function Ne0(v) { return { kind: 'cond', type: 'ne0', v }; }
1586 function Cmp(l, op, r) { return { kind: 'cond', type: 'cmp', l, op, r }; }
1587 function And(a, b) {
1588     if (!a) return b;
1589     if (!b) return a;
1590     return { kind: 'cond', type: 'and', a, b };
1591 }
1592 function Or(a, b) {
1593     if (!a) return b;
1594     if (!b) return a;
1595     return { kind: 'cond', type: 'or', a, b };
1596 }
1597 function Not(x) {
1598     if (!x) return null;
1599     return { kind: 'cond', type: 'not', x };
1600 }
1601 const TrueCond = { kind: 'cond', type: 'true' };
1602 const FalseCond = { kind: 'cond', type: 'false' };
1603
1604 function Const(n) { return { kind: 'expr', type: 'const', n }; }
1605 function Var(name) { return { kind: 'expr', type: 'var', name }; }
1606 function Op(op, l, r) { return { kind: 'expr', type: 'op', op, l, r }; }
1607 function When(c, t, e) { return { kind: 'expr', type: 'when', c, t, e }; }
1608
1609 // === Утилиты ===
1610 function atomKey(c) {
1611     if (!c) return null;
1612     switch (c.type) {
1613         case 'eq0': return `eq0:${c.v}`;
1614         case 'ne0': return `ne0:${c.v}`;
1615         case 'cmp': return `cmp:${c.l}: ${c.op}: ${c.r}`;
1616         case 'true': return 'true';
1617         case 'false': return 'false';
1618         default: return null;
1619     }
1620 }
1621
```

```
1622 function negateOp(op) {
1623     switch (op) {
1624         case '=': return '!=';
1625         case '!=': return '=';
1626         case '>': return '<';
1627         case '<': return '>';
1628         case '>=': return '<';
1629         case '<=': return '>';
1630         default: return null;
1631     }
1632 }
1633
1634 // Преобразует cmp-условие в интервал по одной переменной
1635 // Возвращает { varName, min, minInc, max, maxInc } или null
1636 function cmpToInterval(c) {
1637     if (!c || c.type !== 'cmp') return null;
1638
1639     const lNum = parseNumberLiteral(c.l);
1640     const rNum = parseNumberLiteral(c.r);
1641
1642     let varName, op, val;
1643
1644     if (lNum == null && rNum != null) {
1645         // var OP const
1646         varName = c.l;
1647         op = c.op;
1648         val = rNum;
1649     } else if (lNum != null && rNum == null) {
1650         // const OP var -> var (OP') const
1651         varName = c.r;
1652         op = reverseOp(c.op);
1653         if (!op) return null;
1654         val = lNum;
1655     } else {
1656         // Либо обе стороны числа, либо обе не числа – не трогаем
1657         return null;
1658     }
1659
1660     // Интересуют только упорядочивающие операторы
1661     switch (op) {
1662         case '<':
1663         case '<=':
1664         case '>':
1665         case '>=':
1666         case '=':
1667             break;
1668         default:
1669             return null;
1670     }
1671
1672     let min = Number.NEGATIVE_INFINITY;
1673     let max = Number.POSITIVE_INFINITY;
1674     let minInc = false;
1675     let maxInc = false;
1676
1677     switch (op) {
1678         case '<':
1679             max = val; maxInc = false; break;
1680         case '<=':
1681             max = val; maxInc = true; break;
1682         case '>':
1683             min = val; minInc = false; break;
1684         case '>=':
1685             min = val; minInc = true; break;
1686         case '=':
```

```
1687         min = val; minInc = true;
1688         max = val; maxInc = true;
1689         break;
1690     }
1691
1692     return { varName, min, minInc, max, maxInc };
1693 }
1694
1695 function intervalSubset(a, b) {
1696     if (!a || !b) return false;
1697
1698     // Нижняя граница: a.min >= b.min
1699     const amin = a.min, bmin = b.min;
1700     if (amin === Number.NEGATIVE_INFINITY) {
1701         if (bmin !== Number.NEGATIVE_INFINITY) return false;
1702         // оба -∞ – ок
1703     } else if (bmin === Number.NEGATIVE_INFINITY) {
1704         // b начинается “раньше” – ок
1705     } else if (amin > bmin) {
1706         // a стартует правее b – ок
1707     } else if (amin < bmin) {
1708         // a захватывает меньшее значение – не подмножество
1709         return false;
1710     } else {
1711         // amin === bmin
1712         if (a.minInc && !b.minInc) {
1713             // a включает границу, а b – нет → в a есть точка, не входящая в b
1714             return false;
1715         }
1716     }
1717
1718     // Верхняя граница: a.max <= b.max
1719     const amax = a.max, bmax = b.max;
1720     if (amax === Number.POSITIVE_INFINITY) {
1721         if (bmax !== Number.POSITIVE_INFINITY) return false;
1722     } else if (bmax === Number.POSITIVE_INFINITY) {
1723         // b идёт дальше – ок
1724     } else if (amax < bmax) {
1725         // a заканчивается раньше – ок
1726     } else if (amax > bmax) {
1727         return false;
1728     } else {
1729         // amax === bmax
1730         if (a.maxInc && !b.maxInc) {
1731             return false;
1732         }
1733     }
1734
1735     return true;
1736 }
1737
1738 // Удаляет избыточные cmp-условия в массиве атомов
1739 // mode: 'and' | 'or'
1740 function removeRedundantCmpAtoms(atoms, mode) {
1741     if (!atoms || atoms.length < 2) return atoms;
1742
1743     const keep = new Array(atoms.length).fill(true);
1744
1745     for (let i = 0; i < atoms.length; i++) {
1746         if (!keep[i]) continue;
1747         const a = atoms[i];
1748         if (!a || a.type !== 'cmp') continue;
1749
1750         for (let j = 0; j < atoms.length; j++) {
1751             if (i === j || !keep[j]) continue;
```

```
1752         const b = atoms[j];
1753         if (!b || b.type !== 'cmp') continue;
1754
1755         const rel = cmpImplicationRelation(a, b);
1756         if (!rel) continue;
1757
1758         if (rel === 'a_in_b') {
1759             if (mode === 'or') {
1760                 // A ⊆ B → A OR B = B → A лишнее
1761                 keep[i] = false;
1762                 break;
1763             } else if (mode === 'and') {
1764                 // A ⊆ B → A AND B = A → B лишнее
1765                 keep[j] = false;
1766             }
1767         } else if (rel === 'b_in_a') {
1768             if (mode === 'or') {
1769                 // B ⊆ A → A OR B = A → B лишнее
1770                 keep[j] = false;
1771             } else if (mode === 'and') {
1772                 // B ⊆ A → A AND B = B → A лишнее
1773                 keep[i] = false;
1774                 break;
1775             }
1776         }
1777     }
1778 }
1779
1780     return atoms.filter(_ , idx) => keep[idx]);
1781 }
1782
1783 // Отношение между двумя cmp-условиями через интервалы
1784 // 'a_in_b' – A ⊆ B
1785 // 'b_in_a' – B ⊆ A
1786 // 'equal' – одинаковые интервалы (редко используем)
1787 // null – не можем определить
1788 function cmpImplicationRelation(c1, c2) {
1789     const i1 = cmpToInterval(c1);
1790     const i2 = cmpToInterval(c2);
1791     if (!i1 || !i2) return null;
1792     if (i1.varName !== i2.varName) return null;
1793
1794     const aInB = intervalSubset(i1, i2);
1795     const bInA = intervalSubset(i2, i1);
1796
1797     if (aInB && bInA) return 'equal';
1798     if (aInB) return 'a_in_b';
1799     if (bInA) return 'b_in_a';
1800     return null;
1801 }
1802
1803 // Разворот оператора при перестановке аргументов (левый/правый)
1804 function reverseOp(op) {
1805     switch (op) {
1806         case '<': return '>';
1807         case '>': return '<';
1808         case '<=': return '>=';
1809         case '>=': return '<=';
1810         case '=':
1811         case '!=':
1812             return op;
1813         default:
1814             return null;
1815     }
1816 }
```

```
1817
1818 // Аккуратный парсер числового литерала.
1819 // Возвращает число или null, если строка не чисто числовая.
1820 function parseNumberLiteral(s) {
1821     if (typeof s !== 'string') return null;
1822     const trimmed = s.trim().replace(',', '.');
1823
1824     // Только простые вещи: -123, 45, 3.14
1825     if (!/^-\?\d+(\.\d+)?$/ .test(trimmed)) return null;
1826
1827     const n = Number(trimmed);
1828     return Number.isFinite(n) ? n : null;
1829 }
1830
1831
1832 function negateAtomKey(key) {
1833     if (!key) return null;
1834     if (key.startsWith('eq0:')) return 'ne0:' + key.slice(4);
1835     if (key.startsWith('ne0:')) return 'eq0:' + key.slice(4);
1836     if (key.startsWith('cmp:')) {
1837         const parts = key.slice(4).split(':');
1838         if (parts.length === 3) {
1839             const negOp = negateOp(parts[1]);
1840             if (negOp) return `cmp:${parts[0]}:${negOp}:${parts[2]}`;
1841         }
1842     }
1843     return null;
1844 }
1845
1846 function isNegation(a, b) {
1847     if (!a || !b) return false;
1848     if (a.type === 'eq0' && b.type === 'ne0' && a.v === b.v) return true;
1849     if (a.type === 'ne0' && b.type === 'eq0' && a.v === b.v) return true;
1850     if (a.type === 'cmp' && b.type === 'cmp' && a.l === b.l && a.r === b.r) {
1851         return a.op === negateOp(b.op);
1852     }
1853     if (a.type === 'not' && condEq(a.x, b)) return true;
1854     if (b.type === 'not' && condEq(b.x, a)) return true;
1855     return false;
1856 }
1857
1858 function condEq(a, b) {
1859     if (a === b) return true;
1860     if (!a || !b) return false;
1861     if (a.type !== b.type) return false;
1862
1863     switch (a.type) {
1864         case 'eq0':
1865         case 'ne0':
1866             return a.v === b.v;
1867         case 'cmp':
1868             return a.l === b.l && a.op === b.op && a.r === b.r;
1869         case 'true':
1870         case 'false':
1871             return true;
1872         case 'not':
1873             return condEq(a.x, b.x);
1874         case 'and':
1875         case 'or':
1876             return (condEq(a.a, b.a) && condEq(a.b, b.b)) ||
1877                     (condEq(a.a, b.b) && condEq(a.b, b.a));
1878         default:
1879             return false;
1880     }
1881 }
```

```
1882
1883     function flattenAnd(c) {
1884         if (!c) return [];
1885         if (c.type === 'and') return [...flattenAnd(c.a), ...flattenAnd(c.b)];
1886         return [c];
1887     }
1888
1889     function flattenOr(c) {
1890         if (!c) return [];
1891         if (c.type === 'or') return [...flattenOr(c.a), ...flattenOr(c.b)];
1892         return [c];
1893     }
1894
1895     function buildAnd(terms) {
1896         if (terms.length === 0) return TrueCond;
1897         let result = terms[0];
1898         for (let i = 1; i < terms.length; i++) {
1899             result = And(result, terms[i]);
1900         }
1901         return result;
1902     }
1903
1904     function buildOr(terms) {
1905         if (terms.length === 0) return FalseCond;
1906         let result = terms[0];
1907         for (let i = 1; i < terms.length; i++) {
1908             result = Or(result, terms[i]);
1909         }
1910         return result;
1911     }
1912
1913 // Поглощение для AND: X AND (X OR Y) = X
1914 function applyAndAbsorption(terms) {
1915     if (!terms || terms.length < 2) return terms;
1916
1917     const keep = new Array(terms.length).fill(true);
1918
1919     for (let i = 0; i < terms.length; i++) {
1920         if (!keep[i]) continue;
1921         const ti = terms[i];
1922         if (!ti || ti.type !== 'or') continue;
1923
1924         const orParts = flattenOr(ti);
1925         let drop = false;
1926
1927         outer:
1928         for (const part of orParts) {
1929             for (let j = 0; j < terms.length; j++) {
1930                 if (j === i || !keep[j]) continue;
1931                 if (condEq(part, terms[j])) {
1932                     drop = true;
1933                     break outer;
1934                 }
1935             }
1936         }
1937
1938         if (drop) {
1939             keep[i] = false;
1940         }
1941     }
1942
1943     return terms.filter((_, idx) => keep[idx]);
1944 }
1945
1946 // Поглощение для OR: X OR (X AND Y) = X
```

```
1947 function applyOrAbsorption(terms) {
1948     if (!terms || terms.length < 2) return terms;
1949
1950     const keep = new Array(terms.length).fill(true);
1951
1952     for (let i = 0; i < terms.length; i++) {
1953         if (!keep[i]) continue;
1954         const ti = terms[i];
1955         if (!ti || ti.type !== 'and') continue;
1956
1957         const andParts = flattenAnd(ti);
1958         let drop = false;
1959
1960         outer:
1961             for (const part of andParts) {
1962                 for (let j = 0; j < terms.length; j++) {
1963                     if (j === i || !keep[j]) continue;
1964                     if (condEq(part, terms[j])) {
1965                         drop = true;
1966                         break outer;
1967                     }
1968                 }
1969             }
1970
1971             if (drop) {
1972                 keep[i] = false;
1973             }
1974         }
1975
1976     return terms.filter((_, idx) => keep[idx]);
1977 }
1978
1979 // === Упрощение условий ===
1980 function simplifyCond(c) {
1981     _depth++;
1982     if (_depth > MAX_DEPTH) {
1983         _depth--;
1984         return c;
1985     }
1986
1987     try {
1988         return simplifyCondCore(c);
1989     } finally {
1990         _depth--;
1991     }
1992 }
1993
1994 function simplifyCondCore(c) {
1995     if (!c || c.kind !== 'cond') return c;
1996
1997     switch (c.type) {
1998         case 'true':
1999         case 'false':
2000         case 'eq0':
2001         case 'ne0':
2002         case 'cmp':
2003             return c;
2004
2005         case 'not': {
2006             const x = simplifyCondCore(c.x);
2007             if (!x) return TrueCond;
2008             if (x.type === 'true') return FalseCond;
2009             if (x.type === 'false') return TrueCond;
2010             if (x.type === 'not') return simplifyCondCore(x.x);
2011             if (x.type === 'eq0') return Ne0(x.v);
2012         }
2013     }
2014 }
```

```
2012     if (x.type === 'ne0') return Eq0(x.v);
2013     if (x.type === 'cmp') {
2014         const negOp = negateOp(x.op);
2015         if (negOp) return Cmp(x.l, negOp, x.r);
2016     }
2017     if (x.type === 'and') return simplifyCondCore(Or(Not(x.a), Not(x.b)));
2018     if (x.type === 'or') return simplifyCondCore(And(Not(x.a), Not(x.b)));
2019     return Not(x);
2020 }
2021
2022 case 'and': {
2023     const a = simplifyCondCore(c.a);
2024     const b = simplifyCondCore(c.b);
2025
2026     if (!a) return b;
2027     if (!b) return a;
2028     if (a.type === 'false' || b.type === 'false') return FalseCond;
2029     if (a.type === 'true') return b;
2030     if (b.type === 'true') return a;
2031
2032     const allTerms = [...flattenAnd(a), ...flattenAnd(b)];
2033
2034 // === НОВОЕ: Сразу собираем все eq0/ne0 для быстрой проверки ===
2035 const eq0Vars = new Map(); // var -> term
2036 const ne0Vars = new Map(); // var -> term
2037 const cmpTerms = [];
2038 const otherTerms = [];
2039
2040 for (const t of allTerms) {
2041     if (t.type === 'true') continue;
2042     if (t.type === 'false') return FalseCond;
2043
2044     if (t.type === 'eq0') {
2045         // Проверка на противоречие сразу
2046         if (ne0Vars.has(t.v)) {
2047             console.log(`Противоречие найдено: ${t.v} = 0 AND ${t.v} != 0`);
2048             return FalseCond;
2049         }
2050         eq0Vars.set(t.v, t);
2051     } else if (t.type === 'ne0') {
2052         // Проверка на противоречие сразу
2053         if (eq0Vars.has(t.v)) {
2054             console.log(`Противоречие найдено: ${t.v} != 0 AND ${t.v} = 0`);
2055             return FalseCond;
2056         }
2057         ne0Vars.set(t.v, t);
2058     } else if (t.type === 'cmp') {
2059         cmpTerms.push(t);
2060     } else if (t.type === 'or') {
2061         // === НОВОЕ: Проверяем каждую ветку OR на противоречие с контекстом ===
2062         const orTerms = flattenOr(t);
2063         const validBranches = [];
2064
2065         for (const branch of orTerms) {
2066             let branchValid = true;
2067
2068             if (branch.type === 'ne0' && eq0Vars.has(branch.v)) {
2069                 console.log(`OR ветка ${branch.v} != 0 противоречит контексту ${branch.v} = 0`);
2070                 branchValid = false;
2071             } else if (branch.type === 'eq0' && ne0Vars.has(branch.v)) {
2072                 console.log(`OR ветка ${branch.v} = 0 противоречит контексту ${branch.v} != 0`);
2073                 branchValid = false;
2074             }
2075         }
2076     }
2077 }
```

```
2075             if (branchValid) {
2076                 validBranches.push(branch);
2077             }
2078         }
2079     }
2080
2081     if (validBranches.length === 0) {
2082         console.log(`Все ветки OR противоречат контексту → FALSE`);
2083         return FalseCond;
2084     } else if (validBranches.length === 1) {
2085         // Если осталась только одна ветка OR, добавляем её напрямую
2086         const singleBranch = validBranches[0];
2087         if (singleBranch.type === 'eq0') {
2088             if (eq0Vars.has(singleBranch.v)) return FalseCond;
2089             eq0Vars.set(singleBranch.v, singleBranch);
2090         } else if (singleBranch.type === 'ne0') {
2091             if (ne0Vars.has(singleBranch.v)) return FalseCond;
2092             ne0Vars.set(singleBranch.v, singleBranch);
2093         } else {
2094             otherTerms.push(singleBranch);
2095         }
2096     } else {
2097         // Перестраиваем OR только с валидными ветками
2098         otherTerms.push(buildOr(validBranches));
2099     }
2100 } else {
2101     otherTerms.push(t);
2102 }
2103 }
2104
2105 // Собираем уникальные атомы
2106 const atomMap = new Map();
2107
2108 for (const [v, term] of eq0Vars) {
2109     const key = atomKey(term);
2110     if (key) atomMap.set(key, term);
2111 }
2112
2113 for (const [v, term] of ne0Vars) {
2114     const key = atomKey(term);
2115     if (key) atomMap.set(key, term);
2116 }
2117
2118 for (const term of cmpTerms) {
2119     const key = atomKey(term);
2120     if (key) {
2121         const negKey = negateAtomKey(key);
2122         if (negKey && atomMap.has(negKey)) {
2123             return FalseCond;
2124         }
2125         if (!atomMap.has(key)) {
2126             atomMap.set(key, term);
2127         }
2128     }
2129 }
2130
2131 let uniqueAtoms = Array.from(atomMap.values());
2132 uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'and');
2133
2134 let result = [...uniqueAtoms, ...otherTerms];
2135
2136 // Поглощение: X AND (X OR Y) = X
2137 result = applyAndAbsorption(result);
2138
2139 if (result.length === 0) return TrueCond;
```

```
2140     if (result.length === 1) return result[0];
2141
2142     return buildAnd(result);
2143 }
2144
2145     case 'or': {
2146         const a = simplifyCondCore(c.a);
2147         const b = simplifyCondCore(c.b);
2148
2149         if (!a) return b;
2150         if (!b) return a;
2151         if (a.type === 'true' || b.type === 'true') return TrueCond;
2152         if (a.type === 'false') return b;
2153         if (b.type === 'false') return a;
2154
2155         const allTerms = [...flattenOr(a), ...flattenOr(b)];
2156         const atomMap = new Map();
2157         const otherTerms = [];
2158
2159         for (const t of allTerms) {
2160             if (t.type === 'true') return TrueCond;
2161             if (t.type === 'false') continue;
2162
2163             const key = atomKey(t);
2164             if (key) {
2165                 const negKey = negateAtomKey(key);
2166                 if (negKey && atomMap.has(negKey)) {
2167                     return TrueCond;
2168                 }
2169                 if (!atomMap.has(key)) {
2170                     atomMap.set(key, t);
2171                 }
2172             } else {
2173                 otherTerms.push(t);
2174             }
2175         }
2176
2177         let uniqueAtoms = Array.from(atomMap.values());
2178         uniqueAtoms = removeRedundantCmpAtoms(uniqueAtoms, 'or');
2179
2180         let result = [...uniqueAtoms, ...otherTerms];
2181
2182         // Поглощение: X OR (X AND Y) = X
2183         result = applyOrAbsorption(result);
2184
2185         if (result.length === 0) return FalseCond;
2186         if (result.length === 1) return result[0];
2187
2188         return buildOr(result);
2189     }
2190
2191     default:
2192         return c;
2193 }
2194 }
2195
2196 // === Сравнение выражений ===
2197 function exprEq(a, b) {
2198     if (a === b) return true;
2199     if (!a && !b) return true;
2200     if (!a || !b) return false;
2201     if (a.type !== b.type) return false;
2202
2203     switch (a.type) {
2204         case 'const': return a.n === b.n;
```

```
2205     case 'var': return a.name === b.name;
2206     case 'op': return a.op === b.op && exprEq(a.l, b.l) && exprEq(a.r, b.r);
2207     case 'when': return condEq(a.c, b.c) && exprEq(a.t, b.t) && exprEq(a.e, b.e);
2208     default: return false;
2209   }
2210 }
2211
2212 // === Упрощение выражений ===
2213 function simplifyExpr(expr) {
2214   _depth++;
2215   if (_depth > MAX_DEPTH) {
2216     _depth--;
2217     return expr;
2218   }
2219
2220   try {
2221     return simplifyExprCore(expr);
2222   } finally {
2223     _depth--;
2224   }
2225 }
2226
2227 function simplifyExprCore(expr) {
2228   if (!expr || expr.kind !== 'expr') return expr;
2229
2230   switch (expr.type) {
2231     case 'const':
2232     case 'var':
2233       return expr;
2234
2235     case 'op': {
2236       const l = simplifyExprCore(expr.l);
2237       const r = simplifyExprCore(expr.r);
2238
2239       if (expr.op === '+') {
2240         if (r?.type === 'const' && r.n === 0) return l;
2241         if (l?.type === 'const' && l.n === 0) return r;
2242       }
2243       if (expr.op === '*') {
2244         if (l?.type === 'const' && l.n === 0) return Const(0);
2245         if (r?.type === 'const' && r.n === 0) return Const(0);
2246         if (l?.type === 'const' && l.n === 1) return r;
2247         if (r?.type === 'const' && r.n === 1) return l;
2248       }
2249       return Op(expr.op, l, r);
2250     }
2251
2252     case 'when': {
2253       const c = simplifyCond(expr.c);
2254       const t = simplifyExprCore(expr.t);
2255       const e = simplifyExprCore(expr.e);
2256
2257       if (c?.type === 'true') return t;
2258       if (c?.type === 'false') return e;
2259       if (exprEq(t, e)) return t;
2260
2261       return When(c, t, e);
2262     }
2263
2264     default:
2265       return expr;
2266   }
2267 }
2268
2269 // === Печать ===
```

```
2270 function printCond(c) {
2271     if (!c) return 'TRUE';
2272
2273     switch (c.type) {
2274         case 'eq0': return `${c.v} = 0`;
2275         case 'ne0': return `${c.v} != 0`;
2276         case 'cmp': return `${c.l} ${c.op} ${c.r}`;
2277         case 'and': return `(${printCond(c.a)} AND ${printCond(c.b)})`;
2278         case 'or': return `(${printCond(c.a)} OR ${printCond(c.b)})`;
2279         case 'not': return `NOT(${printCond(c.x)})`;
2280         case 'true': return 'TRUE';
2281         case 'false': return 'FALSE';
2282         default: return '?';
2283     }
2284 }
2285
2286 function printExpr(e) {
2287     if (!e) return '0';
2288
2289     switch (e.type) {
2290         case 'const': return String(e.n);
2291         case 'var': return e.name;
2292         case 'op': return `(${printExpr(e.l)}${e.op}${printExpr(e.r)})`;
2293         case 'when': return `WHEN(${printCond(e.c)}, ${printExpr(e.t)}, ${printExpr(e.e)})`;
2294         default: return '?';
2295     }
2296 }
2297
2298 window.Optimizer = {
2299     Eq0, Ne0, Cmp, And, Or, Not, TrueCond, FalseCond,
2300     Const, Var, Op, When,
2301     simplifyCond, simplifyExpr,
2302     printCond, printExpr,
2303     condEq, exprEq
2304 };
2305
2306 codegen.js:
2307 // js/codegen.js
2308
2309 const CodeGen = {
2310     _cache: {},
2311     _branchCache: {},
2312     _resolveCache: {},
2313     _visiting: new Set(),
2314
2315     reset() {
2316         this._cache = {};
2317         this._branchCache = {};
2318         this._resolveCache = {};
2319         this._visiting = new Set();
2320     },
2321
2322     toExpr(valueStr) {
2323         const s = String(valueStr).trim();
2324         if (s === '0') return Optimizer.Const(0);
2325         const num = parseFloat(s);
2326         if (!isNaN(num) && String(num) === s) return Optimizer.Const(num);
2327         return Optimizer.Var(s);
2328     },
2329
2330     exprToName(exprAst) {
2331         if (!exprAst) return '0';
2332         if (exprAst.type === 'var') return exprAst.name;
2333         if (exprAst.type === 'const') return String(exprAst.n);
```

```
2334         return Optimizer.printExpr(exprAst);
2335     },
2336
2337     mergeCond(a, b) {
2338         if (!a && !b) return null;
2339         if (!a) return b;
2340         if (!b) return a;
2341         if (Optimizer.condEq && Optimizer.condEq(a, b)) return a;
2342         return Optimizer.And(a, b);
2343     },
2344
2345     getConn(toId, toPort) {
2346         return AppState.connections.find(c => c.toElement === toId && c.toPort ===
2347         toPort);
2348     },
2349
2350     getConns(toId, prefix) {
2351         return AppState.connections.filter(c => c.toElement === toId &&
2352         c.toPort.startsWith(prefix));
2353     },
2354
2355     buildFormulaExpr(elem) {
2356         const expression = elem.props.expression || '0';
2357         let result = expression;
2358         const formulaRefs = result.match(/formula-\d+/g) || [];
2359
2360         for (const ref of formulaRefs) {
2361             const refElem = AppState.elements[ref];
2362             if (refElem && refElem.type === 'formula') {
2363                 const refExpr = this.buildFormulaExpr(refElem);
2364                 result = result.replace(new RegExp(ref, 'g'), `(${refExpr})`);
2365             }
2366         }
2367
2368         return result;
2369     },
2370
2371     // === Получить ЧИСТУЮ логику элемента ===
2372     getPureLogic(id) {
2373         const cacheKey = `logic:${id}`;
2374         if (cacheKey in this._cache) {
2375             return this._cache[cacheKey];
2376         }
2377
2378         const elem = AppState.elements[id];
2379         if (!elem) return null;
2380
2381         let logic = null;
2382
2383         switch (elem.type) {
2384             case 'if': {
2385                 const leftConn = this.getConn(id, 'in-0');
2386                 const rightConn = this.getConn(id, 'in-1');
2387
2388                 const leftVal = leftConn ? this.getValue(leftConn.fromElement) :
2389                 Optimizer.Const(0);
2390                 const rightVal = rightConn ? this.getValue(rightConn.fromElement) :
2391                 Optimizer.Const(0);
2392
2393                 const op = (elem.props.operator || '=').trim();
2394                 const leftName = this.exprToName(leftVal);
2395                 const rightName = this.exprToName(rightVal);
2396
2397                 const leftZero = leftVal.type === 'const' && leftVal.n === 0;
2398                 const rightZero = rightVal.type === 'const' && rightVal.n === 0;
```

```
2395         switch (op) {
2396             case '=':
2397                 if (rightZero) {
2398                     logic = Optimizer.Eq0(leftName);
2399                 } else if (leftZero) {
2400                     logic = Optimizer.Eq0(rightName);
2401                 } else {
2402                     logic = Optimizer.Cmp(leftName, '=', rightName);
2403                 }
2404                 break;
2405             case '!=':
2406                 if (rightZero) {
2407                     logic = Optimizer.Ne0(leftName);
2408                 } else if (leftZero) {
2409                     logic = Optimizer.Ne0(rightName);
2410                 } else {
2411                     logic = Optimizer.Cmp(leftName, '!=', rightName);
2412                 }
2413                 break;
2414             case '>':
2415             case '<':
2416             case '>=':
2417             case '<=':
2418                 logic = Optimizer.Cmp(leftName, op, rightName);
2419                 break;
2420             default:
2421                 logic = Optimizer.TrueCond;
2422             }
2423             break;
2424         }
2425     }
2426
2427     case 'and':
2428     case 'or': {
2429         const isAnd = elem.type === 'and';
2430         const count = elem.props.inputCount || 2;
2431         let result = null;
2432
2433         for (let i = 0; i < count; i++) {
2434             const conn = this.getConn(id, `in-${i}`);
2435             if (!conn) continue;
2436
2437             const val = this.getPureLogic(conn.fromElement);
2438             if (!val) continue;
2439
2440             if (result === null) {
2441                 result = val;
2442             } else {
2443                 result = isAnd ? Optimizer.And(result, val) :
2444 Optimizer.Or(result, val);
2445             }
2446             logic = result || Optimizer.FalseCond;
2447             break;
2448         }
2449
2450         case 'not': {
2451             const conn = this.getConn(id, 'in-0');
2452             const inputLogic = conn ? this.getPureLogic(conn.fromElement) : null;
2453             logic = Optimizer.Not(inputLogic || Optimizer.FalseCond);
2454             break;
2455         }
2456
2457         case 'separator': {
2458             const conn = this.getConn(id, 'in-0');
```

```
2459             logic = conn ? this.getPureLogic(conn.fromElement) :
2460             Optimizer.FalseCond;
2461             break;
2462         }
2463     default:
2464         logic = null;
2465     }
2466
2467     // ↓ новая часть: добавляем контекст с cond-порта для логических элементов
2468     if (elem.type === 'if' || elem.type === 'and' || elem.type === 'or' ||
2469     elem.type === 'not') {
2470         const ctx = this.getConditionFromPort(id);
2471         if (ctx) {
2472             if (logic) {
2473                 logic = Optimizer.And(ctx, logic);
2474             } else {
2475                 logic = ctx;
2476             }
2477         }
2478
2479         this._cache[cacheKey] = logic;
2480         return logic;
2481     },
2482
2483     // === Получить значение ===
2484     getValue(id) {
2485         const elem = AppState.elements[id];
2486         if (!elem) return Optimizer.Const(0);
2487
2488         switch (elem.type) {
2489             case 'input-signal':
2490                 // Имя сигнала или id как Var...
2491                 return this.toExpr(elem.props.name || id);
2492
2493             case 'const':
2494                 return Optimizer.Const(Number(elem.props.value) || 0);
2495
2496             case 'formula': {
2497                 // Используем текст формулы как выражение
2498                 const exprStr = this.buildFormulaExpr(elem) || '0';
2499                 return this.toExpr(exprStr);
2500             }
2501
2502             default:
2503                 // На всякий случай – даём символическое имя, а не 0
2504                 if (elem.props && typeof elem.props.name === 'string') {
2505                     return this.toExpr(elem.props.name);
2506                 }
2507                 return this.toExpr(id);
2508             }
2509         },
2510
2511     // === Получить ПОЛНОЕ условие для ветки сепаратора ===
2512     getBranchCondition(sepId, fromPort) {
2513         const cacheKey = `${sepId}:${fromPort}`;
2514         if (cacheKey in this._branchCache) {
2515             return this._branchCache[cacheKey];
2516         }
2517
2518         const sep = AppState.elements[sepId];
2519         if (!sep || sep.type !== 'separator') return null;
2520
2521         const inputLogic = this.getPureLogic(sepId);
```

```
2522     const sepContext = this.getConditionFromPort(sepId);
2523
2524     let branchLogic;
2525     if (fromPort === 'out-1') {
2526         branchLogic = inputLogic ? Optimizer.Not(inputLogic) : Optimizer.TrueCond;
2527     } else {
2528         branchLogic = inputLogic || Optimizer.TrueCond;
2529     }
2530
2531     let result;
2532     if (sepContext) {
2533         result = Optimizer.And(sepContext, branchLogic);
2534     } else {
2535         result = branchLogic;
2536     }
2537
2538     this._branchCache[cacheKey] = result;
2539     return result;
2540 },
2541
2542 // === Получить условие от cond-порта ===
2543 getConditionFromPort(id) {
2544     const conn = this.getConn(id, 'cond-0');
2545     if (!conn) return null;
2546
2547     const sourceElem = AppState.elements[conn.fromElement];
2548     if (!sourceElem) return null;
2549
2550     if (sourceElem.type === 'separator') {
2551         return this.getBranchCondition(conn.fromElement, conn.fromPort);
2552     }
2553
2554     return this.getPureLogic(conn.fromElement);
2555 },
2556
2557 // === Основная функция разрешения ===
2558 resolve(id) {
2559     if (id in this._resolveCache) {
2560         return this._resolveCache[id];
2561     }
2562
2563     if (this._visiting.has(id)) {
2564         return null;
2565     }
2566     this._visiting.add(id);
2567
2568     const elem = AppState.elements[id];
2569     if (!elem) {
2570         this._visiting.delete(id);
2571         return null;
2572     }
2573
2574     let result = null;
2575
2576     try {
2577         switch (elem.type) {
2578             case 'input-signal':
2579                 result = {
2580                     isValue: true,
2581                     cond: null,
2582                     expr: this.toExpr(elem.props.name || id)
2583                 };
2584                 break;
2585
2586             case 'const': {
```

```
2587         const cond = this.getConditionFromPort(id);
2588         result = {
2589             isValue: true,
2590             cond: cond,
2591             expr: Optimizer.Const(Number(elem.props.value) || 0)
2592         };
2593         break;
2594     }
2595
2596     case 'formula': {
2597         let cond = this.getConditionFromPort(id);
2598
2599         const inConns = this.getConns(id, 'in-');
2600         for (const conn of inConns) {
2601             const inputNode = this.resolve(conn.fromElement);
2602             if (inputNode && inputNode.cond) {
2603                 cond = this.mergeCond(cond, inputNode.cond);
2604             }
2605         }
2606
2607         const fullExpr = this.buildFormulaExpr(elem);
2608         result = {
2609             isValue: true,
2610             cond: cond,
2611             expr: Optimizer.Var(fullExpr)
2612         };
2613         break;
2614     }
2615
2616     default:
2617         result = null;
2618     }
2619 } finally {
2620     this._visiting.delete(id);
2621 }
2622
2623     this._resolveCache[id] = result;
2624     return result;
2625 },
2626
2627 generate() {
2628     console.log('==== Генерация кода (граф) ====');
2629     this.reset();
2630
2631     try {
2632         const outputs = Object.values(AppState.elements).filter(e => e.type ===
2633 'output');
2634
2635         if (outputs.length === 0) {
2636             return /* Нет выходов */;
2637         }
2638
2639         const allVariants = [];
2640
2641         for (const out of outputs) {
2642             const conns = this.getConns(out.id, 'in-');
2643
2644             for (const conn of conns) {
2645                 console.log(`\n==== Обработка выхода ${out.id}, вход от $` +
2646 ` ${conn.fromElement} ===`);
2647                 const graph = CodeGenGraph.buildDependencyGraph(conn.fromElement);
2648                 const result = CodeGenGraph.evalGraphValue(graph);
2649                 console.log(`Результат: cond=${Optimizer.printCond(result.cond)},` +
2650 `expr=${Optimizer.printExpr(result.expr)}`);
```

```
2649             if (!result || !result.expr) continue;
2650
2651         null;
2652         const cond = result.cond ? Optimizer.simplifyCond(result.cond) :
2653             null;
2654         const isZero = result.expr.type === 'const' && result.expr.n ===
2655             0;
2656
2657         if (isZero && !cond) continue;
2658
2659         allVariants.push({
2660             cond,
2661             expr: result.expr,
2662             isZero
2663         });
2664     }
2665 }
2666
2667 console.log('Варианты:', allVariants.map(v => ({
2668     cond: Optimizer.printCond(v.cond),
2669     expr: Optimizer.printExpr(v.expr)
2670   })));
2671
2672 if (allVariants.length === 0) return '0';
2673
2674 const valueVariants = allVariants.filter(v => !v.isZero);
2675 if (valueVariants.length === 0) return '0';
2676
2677 let result = Optimizer.Const(0);
2678
2679 for (let i = valueVariants.length - 1; i >= 0; i--) {
2680     const v = valueVariants[i];
2681     if (v.cond) {
2682         result = Optimizer.When(v.cond, v.expr, result);
2683     } else {
2684         result = v.expr;
2685     }
2686 }
2687
2688 const simplified = Optimizer.simplifyExpr(result);
2689 return Optimizer.printExpr(simplified);
2690
2691     } catch (err) {
2692         console.error('Ошибка:', err);
2693         return `/* Ошибка: ${err.message} */`;
2694     }
2695 }
2696
2697 window.CodeGen = CodeGen;
2698
2699 config.js:
2700 /**
2701 * Конфигурация приложения
2702 */
2703
2704 // Типы сигналов
2705 const SIGNAL_TYPE = {
2706     NUMERIC: 'numeric',      // Числовой сигнал
2707     LOGIC: 'logic',          // Логический (может быть TRUE или FALSE)
2708     TRUE: 'true',            // Явно ИСТИНА
2709     FALSE: 'false',          // Явно ЛОЖЬ
2710     ANY: 'any'               // Любой тип
2711 };
2712
2713 // Типы проекта
```

```
2712 const PROJECT_TYPE = {
2713     PARAMETER: 'parameter',
2714     RULE: 'rule'
2715 };
2716
2717 // Конфигурация элементов
2718 const ELEMENT_TYPES = {
2719     'input-signal': {
2720         name: 'Вход',
2721         inputs: 0,
2722         outputs: 1,
2723         outputLabels: ['out'],
2724         outputTypes: [SIGNAL_TYPE.NUMERIC],
2725         color: '#4a90d9',
2726         hasProperties: true,
2727         defaultProps: { name: 'Сигнал', signalType: SIGNAL_TYPE.NUMERIC },
2728         resizable: true,
2729         minWidth: 150,
2730         minHeight: 50
2731     },
2732     'and': {
2733         name: 'И',
2734         inputs: 2, // По умолчанию 2, но может быть изменено
2735         outputs: 1,
2736         inputLabels: ['A', 'B'],
2737         inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
2738         outputLabels: ['результат'],
2739         outputTypes: [SIGNAL_TYPE.LOGIC],
2740         color: '#a855f7',
2741         hasProperties: true, // ← Теперь есть свойства (для изменения количества
2742         // входов)
2743         resizable: true,
2744         minWidth: 120,
2745         minHeight: 80,
2746         hasConditionPort: true,
2747         conditionPortType: SIGNAL_TYPE.LOGIC,
2748         defaultProps: {
2749             inputCount: 2 // ← Новое свойство
2750         }
2751     },
2752     'or': {
2753         name: 'ИЛИ',
2754         inputs: 2, // По умолчанию 2
2755         outputs: 1,
2756         inputLabels: ['A', 'B'],
2757         inputTypes: [SIGNAL_TYPE.LOGIC, SIGNAL_TYPE.LOGIC],
2758         outputLabels: ['результат'],
2759         outputTypes: [SIGNAL_TYPE.LOGIC],
2760         color: '#a855f7',
2761         hasProperties: true, // ← Теперь есть свойства
2762         resizable: true,
2763         minWidth: 120,
2764         minHeight: 80,
2765         hasConditionPort: true,
2766         conditionPortType: SIGNAL_TYPE.LOGIC,
2767         defaultProps: {
2768             inputCount: 2 // ← Новое свойство
2769         }
2770     },
2771     'not': {
2772         name: 'НЕ',
2773         inputs: 1,
2774         outputs: 1,
2775         inputLabels: ['A'],
2776         inputTypes: [SIGNAL_TYPE.LOGIC],
```

```
2776     outputLabels: [ '¬A' ],
2777     outputTypes: [ SIGNAL_TYPE.LOGIC ],
2778     color: '#a855f7',
2779     hasProperties: false,
2780     resizable: true,
2781     minWidth: 100,
2782     minHeight: 60,
2783     hasConditionPort: true,
2784     conditionPortType: SIGNAL_TYPE.LOGIC
2785   },
2786   'if': {
2787     name: 'ЕСЛИ',
2788     inputs: 2,
2789     outputs: 1, // ← Только один выход!
2790     inputLabels: [ 'A', 'B' ],
2791     inputTypes: [ SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY ],
2792     outputLabels: [ 'результат' ], // ← Просто результат
2793     outputTypes: [ SIGNAL_TYPE.LOGIC ], // ← Выход типа LOGIC
2794     color: '#e94560',
2795     hasProperties: true,
2796     defaultProps: { operator: '=' },
2797     resizable: true,
2798     minWidth: 120,
2799     minHeight: 80,
2800     hasConditionPort: true,
2801     conditionPortType: SIGNAL_TYPE.LOGIC
2802   },
2803   'separator': { // ← НОВЫЙ ЭЛЕМЕНТ
2804     name: 'Сепаратор',
2805     inputs: 1,
2806     outputs: 2,
2807     inputLabels: [ 'сигнал' ],
2808     inputTypes: [ SIGNAL_TYPE.LOGIC ],
2809     outputLabels: [ 'ИСТИНА', 'ЛОЖЬ' ],
2810     outputTypes: [ SIGNAL_TYPE.TRUE, SIGNAL_TYPE.FALSE ], // ← TRUE и FALSE
2811     color: '#f59e0b',
2812     hasProperties: false,
2813     resizable: true,
2814     minWidth: 120,
2815     minHeight: 80,
2816     hasConditionPort: true,
2817     conditionPortType: SIGNAL_TYPE.LOGIC
2818   },
2819   'const': {
2820     name: 'Константа',
2821     inputs: 0,
2822     outputs: 1,
2823     outputLabels: [ 'out' ],
2824     outputTypes: [ SIGNAL_TYPE.NUMERIC ],
2825     color: '#3b82f6',
2826     hasProperties: true,
2827     defaultProps: { value: 0 },
2828     resizable: true,
2829     minWidth: 120,
2830     minHeight: 60,
2831     hasConditionPort: true,
2832     conditionPortType: SIGNAL_TYPE.LOGIC
2833   },
2834   'formula': {
2835     name: 'Формула',
2836     inputs: 2,
2837     outputs: 1,
2838     inputLabels: [ 'in1', 'in2' ],
2839     inputTypes: [ SIGNAL_TYPE.ANY, SIGNAL_TYPE.ANY ],
2840     outputLabels: [ 'результат' ],
```

```
2841     outputTypes: [SIGNAL_TYPE.NUMERIC],
2842     color: '#f59e0b',
2843     hasProperties: true,
2844     resizable: true,
2845     minWidth: 140,
2846     minHeight: 80,
2847     defaultProps: {
2848       expression: '',
2849       inputCount: 2
2850     },
2851     hasConditionPort: true,
2852     conditionPortType: SIGNAL_TYPE.LOGIC
2853   },
2854   'output': {
2855     name: 'Выход',
2856     inputs: 1,
2857     outputs: 0,
2858     inputLabels: ['сигнал'],
2859     inputTypes: [SIGNAL_TYPE.ANY],
2860     color: '#10b981',
2861     hasProperties: true,
2862     defaultProps: { label: 'Выход', outputGroup: '' },
2863     resizable: true,
2864     minWidth: 150,
2865     minHeight: 60,
2866   }, // ← важно, если предыдущий элемент не заканчивается запятой
2867   'group': {
2868     name: 'Группа',
2869     inputs: 0,
2870     outputs: 0,
2871     color: '#6b7280',
2872     resizable: true,
2873     minWidth: 200,
2874     minHeight: 120,
2875     hasProperties: true,
2876     defaultProps: { title: 'Группа' }
2877   }
2878 };
2879
2880 const VIEWPORT_CONFIG = {
2881   minZoom: 0.1,
2882   maxZoom: 3,
2883   zoomStep: 0.1,
2884   panSpeed: 1,
2885   canvasWidth: 5000,
2886   canvasHeight: 5000
2887 };
2888
2889 const MINIMAP_CONFIG = {
2890   width: 200,
2891   height: 150,
2892   padding: 10
2893 };
2894
2895 connections.js:
2896 /**
2897  * Модуль работы с соединениями
2898 */
2899
2900 const Connections = {
2901   /**
2902    * Настройка обработчиков порта
2903    */
2904   setupPortHandlers(port) {
2905     port.addEventListener('mousedown', (e) => {
```

```
2906     e.stopPropagation();
2907
2908     if (port.classList.contains('output')) {
2909         const elemId = port.dataset.element;
2910         const portName = port.dataset.port;
2911         const signalType = getOutputPortType(elemId, portName);
2912
2913         AppState.connectingFrom = {
2914             element: elemId,
2915             port: portName
2916         };
2917         AppState.connectingFromType = signalType;
2918
2919         this.highlightCompatiblePorts(signalType);
2920
2921         const svg = document.getElementById('connections-svg');
2922         const startPos = this._getPortCanvasCenter(port);
2923
2924         AppState.tempLine = document.createElementNS('http://www.w3.org/2000/
2925         svg', 'path');
2926         AppState.tempLine.setAttribute('class', 'temp-connection');
2927         AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} L $
2928         ${startPos.x} ${startPos.y}`);
2929         svg.appendChild(AppState.tempLine);
2930     }
2931
2932     port.addEventListener('mouseup', (e) => {
2933         e.stopPropagation();
2934         e.preventDefault();
2935
2936         if (AppState.connectingFrom && port.classList.contains('input')) {
2937             const toElement = port.dataset.element;
2938             const toPortName = port.dataset.port;
2939             const inputType = getInputPortType(toElement, toPortName);
2940
2941             if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
2942                 this.clearConnectionState();
2943                 return;
2944             }
2945
2946             if (AppState.connectingFrom.element !== toElement) {
2947                 const targetElem = AppState.elements[toElement];
2948                 const allowMultipleInputs = targetElem?.type === 'output';
2949
2950                 const exists = AppState.connections.some(c =>
2951                     c.toElement === toElement && c.toPort === toPortName
2952                 );
2953
2954                 if (!exists || allowMultipleInputs) {
2955                     AppState.connections.push({
2956                         fromElement: AppState.connectingFrom.element,
2957                         fromPort: AppState.connectingFrom.port,
2958                         toElement,
2959                         toPort: toPortName,
2960                         signalType: AppState.connectingFromType
2961                     });
2962
2963                     port.classList.add('connected');
2964                     this.drawConnections();
2965                     this.clearConnectionState();
2966                     return;
2967                 }
2968             }
2969         }
2970     });
2971 }
```

```
2969         this.clearConnectionState();
2970     });
2971 
2972     port.addEventListener('mouseenter', () => {
2973         if (AppState.connectingFrom && port.classList.contains('input')) {
2974             const toPortName = port.dataset.port;
2975             const inputType = getInputPortType(port.dataset.element, toPortName);
2976 
2977             if (!areTypesCompatible(AppState.connectingFromType, inputType)) {
2978                 if (AppState.tempLine) {
2979                     AppState.tempLine.classList.add('invalid');
2980                 }
2981             }
2982         }
2983     });
2984 
2985     port.addEventListener('mouseleave', () => {
2986         if (AppState.tempLine) {
2987             AppState.tempLine.classList.remove('invalid');
2988         }
2989     });
2990 },
2991 
2992 /**
2993 * Подсветка совместимых портов
2994 */
2995 highlightCompatiblePorts(signalType) {
2996     document.querySelectorAll('.port.input').forEach(port => {
2997         const inputType = getInputPortType(port.dataset.element,
2998         port.dataset.port);
2999 
3000         if (areTypesCompatible(signalType, inputType)) {
3001             port.classList.add('compatible-highlight');
3002         } else {
3003             port.classList.add('incompatible');
3004         }
3005     });
3006 },
3007 
3008 /**
3009 * Очистка состояния соединения
3010 */
3011 clearConnectionState() {
3012     if (AppState.tempLine) {
3013         AppState.tempLine.remove();
3014         AppState.tempLine = null;
3015     }
3016     AppState.connectingFrom = null;
3017     AppState.connectingFromType = null;
3018 
3019     document.querySelectorAll('.port').forEach(port => {
3020         port.classList.remove('compatible-highlight', 'incompatible');
3021     });
3022 },
3023 
3024 /**
3025 * Отрисовка временной линии соединения
3026 */
3027 drawTempConnection(e) {
3028     if (!AppState.tempLine || !AppState.connectingFrom) return;
3029 
3030     const fromElem = document.getElementById(AppState.connectingFrom.element);
3031     if (!fromElem) return;
3032 }
```

```
3033     const fromPort = fromElem.querySelector(`[data-port="${AppState.connectingFrom.port}"]`);  
3034     if (!fromPort) return;  
3035  
3036     const startPos = this._getPortCanvasCenter(fromPort);  
3037     const endPos = screenToCanvas(e.clientX, e.clientY);  
3038  
3039     const horizontalDist = Math.abs(endPos.x - startPos.x);  
3040     const controlDist = Math.max(horizontalDist * 0.4, 50);  
3041  
3042     // Тянем всегда от выхода (вектор 1, 0)  
3043     const cx1 = startPos.x + controlDist;  
3044     const cy1 = startPos.y;  
3045  
3046     // Вторая точка контроля для плавности за курсором  
3047     const cx2 = endPos.x - controlDist;  
3048     const cy2 = endPos.y;  
3049  
3050     AppState.tempLine.setAttribute('d', `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x} ${endPos.y}`);  
3051     AppState.tempLine.setAttribute('fill', 'none');  
3052 },  
3053  
3054 /**  
3055 * Отрисовка всех соединений  
3056 */  
3057 drawConnections() {  
3058     const svg = document.getElementById('connections-svg');  
3059  
3060     // 1. Очистка старых линий  
3061     svg.querySelectorAll('path:not(.temp-connection)').forEach(p => p.remove());  
3062  
3063     // 2. Сброс визуального состояния портов  
3064     document.querySelectorAll('.port.connected').forEach(port => {  
3065         port.classList.remove('connected');  
3066     });  
3067  
3068     // 3. Перебор всех соединений из AppState  
3069     AppState.connections.forEach(conn => {  
3070         const fromElem = document.getElementById(conn.fromElement);  
3071         const toElem = document.getElementById(conn.toElement);  
3072  
3073         if (!fromElem || !toElem) return;  
3074  
3075         const fromPort = fromElem.querySelector(`[data-port="${conn.fromPort}"]`);  
3076         const toPort = toElem.querySelector(`[data-port="${conn.toPort}"]`);  
3077  
3078         if (!fromPort || !toPort) return;  
3079  
3080         fromPort.classList.add('connected');  
3081         toPort.classList.add('connected');  
3082  
3083         const startPos = this._getPortCanvasCenter(fromPort);  
3084         const endPos = this._getPortCanvasCenter(toPort);  
3085  
3086         if (!startPos || !endPos) return;  
3087  
3088         // Расстояние для изгиба кривой  
3089         const horizontalDist = Math.abs(endPos.x - startPos.x);  
3090         const verticalDist = Math.abs(endPos.y - startPos.y);  
3091         const controlDist = Math.max(horizontalDist * 0.4, 50);  
3092  
3093         // --- ЛОГИКА ГЕОМЕТРИИ (Вектора касательных) ---  
3094         let d;  
3095         let cx1 = startPos.x;
```

```
3096     let cy1 = startPos.y;
3097     let cx2 = endPos.x;
3098     let cy2 = endPos.y;
3099
3100     // ВЫХОД (Source): Касательная (1, 0) -> Всегда вправо
3101     cx1 = startPos.x + controlDist;
3102     cy1 = startPos.y;
3103
3104     // ВХОД (Target):
3105     if (conn.toPort === 'cond-0') {
3106         // Технический порт: Касательная (0, 1) в декартовой (вверх)
3107         // В экранных координатах Y инвертирован, поэтому отнимаем от Y
3108         cx2 = endPos.x;
3109         cy2 = endPos.y - controlDist; // Линия заходит сверху вертикально
3110     } else {
3111         // Обычный вход: Касательная (-1, 0) -> Слева направо
3112         cx2 = endPos.x - controlDist;
3113         cy2 = endPos.y;
3114     }
3115
3116     d = `M ${startPos.x} ${startPos.y} C ${cx1} ${cy1}, ${cx2} ${cy2}, ${endPos.x}
3117 ${endPos.y}`;
3118
3119     const path = document.createElementNS('http://www.w3.org/2000/svg', 'path');
3120     path.setAttribute('d', d);
3121     path.setAttribute('fill', 'none'); // Чтобы не было черных полигонов
3122
3123     // --- ЛОГИКА ЦВЕТА (Классы) ---
3124     let cssClass = 'connection';
3125     const type = conn.signalType;
3126
3127     // Приоритет новым типам сигналов
3128     if (type === SIGNAL_TYPE.TRUE) cssClass += ' true-conn';
3129     else if (type === SIGNAL_TYPE.FALSE) cssClass += ' false-conn';
3130     else if (type === SIGNAL_TYPE.LOGIC) cssClass += ' logic-conn';
3131     else if (type === SIGNAL_TYPE.NUMERIC) cssClass += ' numeric-conn';
3132     else if (type === SIGNAL_TYPE.ANY) cssClass += ' any-conn';
3133
3134     path.setAttribute('class', cssClass);
3135
3136     // Обработчики событий
3137     path.style.pointerEvents = 'stroke';
3138     path.style.cursor = 'pointer';
3139     path.addEventListener('click', () => this.handleConnectionClick(conn));
3140
3141     svg.appendChild(path);
3142 };
3143
3144 if (typeof Outputs !== 'undefined' && Outputs.updateOutputStatus) {
3145     Outputs.updateOutputStatus();
3146 }
3147 Viewport.updateMinimap();
3148 /**
3149 * Обработка клика по соединению (удаление)
3150 */
3151 handleConnectionClick(conn) {
3152     if (confirm('Удалить соединение?')) {
3153         AppState.connections = AppState.connections.filter(c =>
3154             !(c.fromElement === conn.fromElement &&
3155                 c.fromPort === conn.fromPort &&
3156                 c.toElement === conn.toElement &&
3157                 c.toPort === conn.toPort)
3158         );
3159 }
```

```
3160             this.drawConnections();
3161         }
3162     },
3163
3164     /**
3165      * Получение центра порта в координатах Canvas
3166      */
3167     _getPortCanvasCenter(portEl) {
3168         if (!portEl) return null;
3169
3170         const rect = portEl.getBoundingClientRect();
3171         return screenToCanvas(
3172             rect.left + rect.width / 2,
3173             rect.top + rect.height / 2
3174         );
3175     }
3176 };
3177
3178 elements.js:
3179 /**
3180  * Модуль работы с элементами схемы
3181 */
3182
3183 const Elements = {
3184     /**
3185      * Генерация HTML для элемента
3186      */
3187     createElementHTML(elemType, elemId, x, y, props = {}, width, height) {
3188         const config = ELEMENT_TYPES[elemType];
3189         if (!config) throw new Error(`Неизвестный тип элемента: ${elemType}`);
3190
3191         const safe = (value, fallback = '') => (value === null || value ===
3192 undefined) ? fallback : String(value);
3193         const w = width ?? config.minWidth ?? 120;
3194         const h = height ?? config.minLength ?? 60;
3195
3196         const getPortClass = (signalType, direction) => {
3197             const base = direction === 'output' ? 'port output' : 'port input';
3198             if (signalType === SIGNAL_TYPE.LOGIC) return `${base} logic-port`;
3199             if (signalType === SIGNAL_TYPE.NUMBER) return `${base} number-port`;
3200             return `${base} any-port`;
3201         };
3202
3203         // Эта функция buildConditionPort будет вызываться ИНАЧЕ, а не внутри
3204         innerHTML
3205         // Она тут остается, но ее результат не встраивается в HTML-строку
3206         // напрямую, кроме формулы
3207         const buildConditionPortHTML = () => {
3208             return `
3209                 <div class="condition-port-wrapper">
3210                     <div class="condition-port-label">условие</div>
3211                     <div class="port input condition-port"
3212                         data-port="cond-0"
3213                         data-element="${elemId}"
3214                         data-signal-type="${SIGNAL_TYPE.LOGIC}"
3215                         title="Техническое условие">
3216                     </div>
3217                 </div>`;
3218         };
3219
3220         const buildInputPorts = (count, types = [], labels = []) => {
3221             let html = '';
3222             for (let i = 0; i < count; i++) {
3223                 const type = types[i] ?? types[types.length - 1] ??
```

```
SIGNAL_TYPE.ANY;
3222           html += `<div class="${getPortClass(type, 'input')}" data-
port="in-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
|| 'Вход ${i+1}'}`></div>`;
3223           }
3224           return html;
3225       };
3226
3227       const buildOutputPorts = (count, types = [], labels = []) => {
3228           let html = '';
3229           for (let i = 0; i < count; i++) {
3230               const type = types[i] ?? types[types.length - 1] ??
SIGNAL_TYPE.ANY;
3231               html += `<div class="${getPortClass(type, 'output')}" data-
port="out-${i}" data-element="${elemId}" data-signal-type="${type}" title="${labels[i]
|| 'Выход ${i+1}'}`></div>`;
3232           }
3233           return html;
3234       };
3235
3236       const resizeHandles = config.resizable ? `<div class="resize-handle
handle-se" data-direction="se"></div><div class="resize-handle handle-e" data-
direction="e"></div><div class="resize-handle handle-s" data-direction="s"></div>` :
'';
3237           // hasCondClass будет добавляться в addElement
3238           // const hasCondClass = config.hasConditionPort ? 'has-condition-port' :
'';
3239
3240           let innerHTML = '';
3241
3242           if (elemType === 'input-signal') {
3243               const name = safe(props.name, 'Сигнал');
3244               const type = props.signalType || SIGNAL_TYPE.NUMBER;
3245               const symbol = type === SIGNAL_TYPE.LOGIC ? '☒' : '☒';
3246               innerHTML = `
3247                   <div class="element-header" style="background:$
{config.color};">Источник</div>
3248                   <div class="element-body">
3249                       <div class="element-symbol">
3250                           <span class="input-signal-icon">${symbol}</span>
3251                           <span class="input-signal-name">${name}</span>
3252                       </div>
3253                       <div class="ports-right">
3254                           ${buildOutputPorts(1, [type], ['Выход'])}
3255                       </div>
3256                   </div>`;
3257           }
3258           else if (elemType === 'const') {
3259               innerHTML =
3260                   <div class="element-header" style="background:$
{config.color};">Константа</div>
3261                   <div class="element-body">
3262                       <div class="element-symbol">${props.value ?? 0}</div>
3263                       <div class="ports-right">
3264                           ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER], ['Значение'])}
3265                       </div>
3266                   </div>`;
3267           }
3268           else if (elemType === 'separator') {
3269               innerHTML =
3270                   <div class="element-header" style="background:$
{config.color};">Сепаратор</div>
3271                   <div class="element-body">
3272                       <div class="ports-left">${buildInputPorts(1,
config.inputTypes, config.inputLabels)}</div>
```

```
3273                     <div class="element-symbol">/\x</div>
3274                     <div class="ports-right">
3275                         <div class="port output logic-port true-port" data-
3276 port="out-0" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.TRUE}"
3277 title="ИСТИНА"></div>
3278                         <div class="port output logic-port false-port" data-
3279 port="out-1" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.FALSE}"
3280 title="Л0ЖЬ"></div>
3281                     </div>
3282                 </div>`;
3283             }
3284         else if (elemType === 'and' || elemType === 'or') {
3285             const gateSymbol = elemType === 'and' ? 'Λ' : '∨';
3286             const inputCount = props.inputCount || config.defaultProps?.inputCount
3287             || 2;
3288
3289             // Генерируем динамические входы
3290             let inputsHTML = '';
3291             for (let i = 0; i < inputCount; i++) {
3292                 inputsHTML += `<div class="port input logic-port" data-port="in-$
3293 {i}" data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Вход $
3294 {i+1}"></div>`;
3295             }
3296
3297             innerHTML = `
3298                 <div class="element-header" style="background:${config.color};">$
3299 {config.name}</div>
3300                 <div class="element-body">
3301                     <div class="ports-left">
3302                         ${inputsHTML}
3303                     </div>
3304                     <div class="element-symbol">${gateSymbol}</div>
3305                     <div class="ports-right">
3306                         <div class="port output logic-port" data-port="out-0"
3307 data-element="${elemId}" data-signal-type="${SIGNAL_TYPE.LOGIC}" title="Результат"></
3308 div>
3309                     </div>
3310                 </div>`;
3311             }
3312         else if (elemType === 'if') {
3313             const op = safe(props.operator, '=');
3314             innerHTML =
3315                 <div class="element-header" style="background:$
3316 {config.color};">Условие</div>
3317                 <div class="element-body">
3318                     <div class="ports-left">${buildInputPorts(2,
3319 config.inputTypes, config.inputLabels)}</div>
3320                     <div class="element-symbol">${op}</div>
3321                     <div class="ports-right">
3322                         ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['результат'])}
3323                     </div>
3324                 </div>`;
3325             }
3326         else if (elemType === 'not') {
3327             innerHTML =
3328                 <div class="element-header" style="background:$
3329 {config.color};">НЕ</div>
3330                 <div class="element-body">
3331                     <div class="ports-left">${buildInputPorts(1,
3332 [SIGNAL_TYPE.LOGIC], ['A'])}</div>
3333                     <div class="element-symbol">¬</div>
3334                     <div class="ports-right">
3335                         ${buildOutputPorts(1, [SIGNAL_TYPE.LOGIC], ['¬A'])}
3336                     </div>
3337                 </div>`;
3338             }
```

```
3324      }
3325      else if (elemType === 'formula') {
3326          const inputCount = props.inputCount || config.defaultProps?.inputCount
3327          || config.inputs || 2;
3328          const expression = safe(props.expression);
3329          const displayExpression = expression
3330          ? (expression.length > 12 ? `${expression.slice(0, 12)}...` :
3331            expression)
3332          : 'f(x)';
3333          innerHTML = `
3334              ${buildConditionPortHTML()}
3335              <div class="element-header" style="background:$
{config.color};">Формула</div>
3336                  <div class="element-body">
3337                      <div class="ports-left">${buildInputPorts(inputCount,
3338                        config.inputTypes, config.inputLabels)}</div>
3339                      <div class="element-symbol">${displayExpression}</div>
3340                      <div class="ports-right">
3341                          ${buildOutputPorts(1, [SIGNAL_TYPE.NUMBER],
3342                            ['Результат'])}
3343                      </div>
3344                  </div>`;
3345      }
3346      else if (elemType === 'output') {
3347          innerHTML = `
3348              <div class="element-header" style="background:$
{config.color};">Выход</div>
3349                  <div class="element-body">
3350                      <div class="ports-left">
3351                          ${buildInputPorts(1, [SIGNAL_TYPE.ANY], ['сигнал'])}
3352                      </div>
3353                      <div class="element-symbol">${safe(props.label, 'Выход')}</
3354                  div>
3355                      <div class="ports-right"></div>
3356                  </div>`;
3357      }
3358      else if (elemType === 'group') {
3359          const title = props.title || 'Группа';
3360          innerHTML = `
3361              <div class="group-content">
3362                  <div class="group-title">${title}</div>
3363              </div>`;
3364      }
3365      else { // Для любых других (fallback)
3366          innerHTML = `
3367              <div class="element-header" style="background:${config.color};">$
{config.name}</div>
3368                  <div class="element-body">
3369                      <div class="ports-left">${buildInputPorts(config.inputs || 0,
3370                        config.inputTypes, config.inputLabels)}</div>
3371                      <div class="element-symbol">${config.name}</div>
3372                      <div class="ports-right">
3373                          ${buildOutputPorts(config.outputs || 0,
3374                            config.outputTypes, config.outputLabels)}
3375                      </div>
3376                  </div>`;
3377      }
3378
3379      const html = `
3380          <div class="element ${elemType}" id="${elemId}"
3381              style="left:${x}px; top:${y}px; width:${w}px; height:${h}px;"
```

```
data-type="${elemType}">
    ${innerHTML}
    ${resizeHandles}
</div>';

        return { html, width: w, height: h };
    },
    /**
     * Добавление элемента
     */
    addElement(elemType, x, y, props = {}, elemId = null, customWidth = null,
customHeight = null) {
        const config = ELEMENT_TYPES[elemType];
        if (!config) {
            console.error(`Неизвестный тип элемента: ${elemType}`);
            return null;
        }

        if (!elemId) {
            elemId = `${elemType}-${++AppState.elementCounter}`;
        }

        let width = customWidth;
        let height = customHeight;

        if (width === null || width === undefined) {
            width = config.minLength || 140;
        }
        if (height === null || height === undefined) {
            height = config.minLength || 70;
        }

        try {
            const result = this.createElementHTML(elemType, elemId, x, y, props,
width, height);
            if (!result || !result.html) {
                console.error('createElementHTML вернул пустой результат');
                return null;
            }

            const workspace = document.getElementById('workspace');
            const wrapper = document.createElement('div');
            wrapper.innerHTML = result.html.trim();
            const element = wrapper.firstChild;
            if (!element) {
                console.error('Не удалось создать DOM элемент из HTML');
                return null;
            }

            // Добавляем класс для отступа
            if (config.hasConditionPort) {
                element.classList.add('has-condition-port');
            }

            workspace.appendChild(element);

            AppState.elements[elemId] = {
                id: elemId,
                type: elemType,
                x,
                y,
                width: result.width || width,
                height: result.height || height,
                props: { ... (config.defaultProps || {}), ... (props || {}) }
            }
        } catch (error) {
            console.error(`Ошибка при создании элемента ${elemType}: ${error}`);
        }
    }
}
```

```
3441                     };
3442
3443                     // ЕСЛИ У ЭЛЕМЕНТА ЕСТЬ COND-ПОРТ (И ОН НЕ ФОРМУЛА, КОТОРАЯ УЖЕ ИМЕЕТ
3444                     // ЕГО В HTML)
3445                     if (config.hasConditionPort && elemType !== 'formula') {
3446                         const condPortWrapper = document.createElement('div');
3447                         condPortWrapper.innerHTML =
3448                             `

3449                             <div class="condition-port-label">условие</div>
3450                             <div class="port input condition-port"
3451                                 data-port="cond-0"
3452                                 data-element="${elemId}"
3453                                 data-signal-type="${SIGNAL_TYPE.LOGIC}"
3454                                 title="Техническое условие">
3455                             </div>
3456

`;
3457                     element.prepend(condPortWrapper.firstChild); // Вставляем в
3458                     самое начало элемента
3459                     }
3460
3461                     this.setupElementHandlers(elemId); // Передаем ID элемента
3462
3463                     // Порты инициализируются внутри setupElementHandlers, нет нужды здесь
3464                     // element.querySelectorAll('.port').forEach(port => {
3465                     //     Connections.setupPortHandlers(port);
3466                     // });
3467
3468                     Connections.drawConnections(); // Перерисовываем соединения, чтобы
3469                     // учесть новые порты
3470                     Viewport.updateMinimap();
3471                     return elemId;
3472                     } catch (err) {
3473                         console.error(`Ошибка при добавлении элемента ${elemType}:`, err);
3474                         return null;
3475                     }
3476                     },
3477
3478                     /**
3479                     * Обновление входов логического элемента (AND, OR)
3480                     */
3481                     updateLogicGateInputs(elemId, inputCount) {
3482                         const elem = document.getElementById(elemId);
3483                         if (!elem) return;
3484
3485                         const portsLeft = elem.querySelector('.ports-left');
3486                         if (!portsLeft) return;
3487
3488                         // Удаляем соединения к портам, которые больше не существуют
3489                         AppState.connections = AppState.connections.filter(c => {
3490                             if (c.toElement === elemId && c.toPort.startsWith('in-')) {
3491                                 const portNum = parseInt(c.toPort.split('-')[1], 10);
3492                                 return portNum < inputCount;
3493                             }
3494                             return true;
3495                         });
3496
3497                         // Генерируем новые входы
3498                         let inputsHTML = '';
3499                         for (let i = 0; i < inputCount; i++) {
3500                             inputsHTML += `
3501                             <div class="port input logic-port"
3502                                 data-port="in-${i}"
3503                                 data-element="${elemId}"
3504                                 data-signal-type="${SIGNAL_TYPE.LOGIC}"`
```

```
3503             title="Вход ${i+1}">
3504         </div>
3505     `;
3506   }
3507   portsLeft.innerHTML = inputsHTML;
3508
3509   // Переподключаем обработчики
3510   portsLeft.querySelectorAll('.port').forEach(port =>
3511     Connections.setupPortHandlers(port)
3512   );
3513
3514   Connections.drawConnections();
3515 },
3516
3517 /**
3518 * Удаление элемента
3519 */
3520 deleteElement(elemId) {
3521   AppState.connections = AppState.connections.filter(c =>
3522     c.fromElement !== elemId && c.toElement !== elemId
3523   );
3524
3525   const elem = document.getElementById(elemId);
3526   if (elem) elem.remove();
3527
3528   delete AppState.elements[elemId];
3529
3530   if (AppState.selectedElement === elemId) {
3531     AppState.selectedElement = null;
3532   }
3533
3534   Connections.drawConnections();
3535   Viewport.updateMinimap();
3536 },
3537
3538 /**
3539 * Выделение элемента
3540 */
3541 selectElement(elemId) {
3542   if (AppState.selectedElement) {
3543     const oldElem = document.getElementById(AppState.selectedElement);
3544     if (oldElem) oldElem.classList.remove('selected');
3545   }
3546
3547   AppState.selectedElement = elemId;
3548   const elem = document.getElementById(elemId);
3549   if (elem) elem.classList.add('selected');
3550
3551   const elemData = AppState.elements[elemId];
3552   if (elemData) {
3553     document.getElementById('selection-info').textContent =
3554       `Выбрано: ${ELEMENT_TYPES[elemData.type]?.name || elemData.type}`;
3555   }
3556 },
3557
3558 /**
3559 * Снять выделение
3560 */
3561 deselectAll() {
3562   if (AppState.selectedElement) {
3563     const elem = document.getElementById(AppState.selectedElement);
3564     if (elem) elem.classList.remove('selected');
3565     AppState.selectedElement = null;
3566   }
3567   document.getElementById('selection-info').textContent = '';
```

```
3568     },
3569
3570     /**
3571      * Настройка обработчиков элемента
3572      */
3573     setupElementHandlers(elemId) {
3574         try {
3575             const elem = document.getElementById(elemId);
3576             if (!elem) return;
3577
3578             elem.addEventListener('mousedown', (e) => {
3579                 if (e.target.classList.contains('port')) return;
3580                 if (e.target.classList.contains('resize-handle')) return;
3581
3582                 e.preventDefault();
3583                 e.stopPropagation();
3584
3585                 this.selectElement(elemId);
3586
3587                 AppState.draggingElement = elemId;
3588                 const canvasPos = screenToCanvas(e.clientX, e.clientY);
3589                 const elemData = AppState.elements[elemId];
3590                 AppState.dragOffset.x = canvasPos.x - elemData.x;
3591                 AppState.dragOffset.y = canvasPos.y - elemData.y;
3592             });
3593
3594             elem.addEventListener('dblclick', (e) => {
3595                 if (e.target.classList.contains('port')) return;
3596                 const config = ELEMENT_TYPES[AppState.elements[elemId].type];
3597                 if (config?.hasProperties) {
3598                     Modal.showPropertiesModal(elemId);
3599                 }
3600             });
3601
3602             elem.addEventListener('contextmenu', (e) => {
3603                 e.preventDefault();
3604                 this.showContextMenu(e.clientX, e.clientY, elemId);
3605             });
3606
3607             const handles = elem.querySelectorAll('.resize-handle');
3608             handles.forEach(handle => this.setupResizeHandlers(handle, elemId));
3609
3610             const ports = elem.querySelectorAll('.port');
3611             ports.forEach(port => Connections.setupPortHandlers(port));
3612
3613         } catch (err) {
3614             console.error('setupElementHandlers error for', elemId, err);
3615         }
3616     },
3617
3618     /**
3619      * Контекстное меню
3620      */
3621     showContextMenu(x, y, elemId) {
3622         const menu = document.getElementById('context-menu');
3623         menu.style.left = `${x}px`;
3624         menu.style.top = `${y}px`;
3625         menu.style.display = 'block';
3626         menu.dataset.elementId = elemId;
3627     },
3628
3629     /**
3630      * Настройка resize
3631      */
3632     setupResizeHandlers(handle, elemId) {
```

```
3633     handle.addEventListener('mousedown', (e) => {
3634         e.stopPropagation();
3635         e.preventDefault();
3636 
3637         const elemData = AppState.elements[elemId];
3638 
3639         AppState.resizing = {
3640             elemId,
3641             handle: handle.dataset.direction,
3642             startX: e.clientX,
3643             startY: e.clientY,
3644             startWidth: elemData.width,
3645             startHeight: elemData.height,
3646             startLeft: elemData.x,
3647             startTop: elemData.y
3648         };
3649     });
3650 },
3651 /**
3652 * Обработка resize
3653 */
3654 handleResize(e) {
3655     if (!AppState.resizing) return;
3656 
3657     const { elemId, handle, startX, startY, startWidth, startHeight, startLeft,
3658 startTop } = AppState.resizing;
3659     const elem = document.getElementById(elemId);
3660     const elemData = AppState.elements[elemId];
3661     const config = ELEMENT_TYPES[elemData.type];
3662 
3663     const dx = (e.clientX - startX) / AppState.viewport.zoom;
3664     const dy = (e.clientY - startY) / AppState.viewport.zoom;
3665 
3666     let newWidth = startWidth;
3667     let newHeight = startHeight;
3668     let newLeft = startLeft;
3669     let newTop = startTop;
3670 
3671     if (handle.includes('e')) {
3672         newWidth = Math.max(config.minWidth, startWidth + dx);
3673     }
3674     if (handle.includes('w')) {
3675         newWidth = Math.max(config.minWidth, startWidth - dx);
3676         newLeft = startLeft + (startWidth - newWidth);
3677     }
3678     if (handle.includes('s')) {
3679         newHeight = Math.max(config.minLength, startHeight + dy);
3680     }
3681     if (handle.includes('n')) {
3682         newHeight = Math.max(config.minLength, startHeight - dy);
3683         newTop = startTop + (startHeight - newHeight);
3684     }
3685 
3686     elem.style.width = `${newWidth}px`;
3687     elem.style.height = `${newHeight}px`;
3688     elem.style.left = `${newLeft}px`;
3689     elem.style.top = `${newTop}px`;
3690 
3691     elemData.width = newWidth;
3692     elemData.height = newHeight;
3693     elemData.x = newLeft;
3694     elemData.y = newTop;
3695 
3696     Connections.drawConnections();
```

```
3697     },
3698
3699     /**
3700      * Обработка перетаскивания элемента
3701      */
3702     handleDrag(e) {
3703         if (!AppState.draggingElement) return;
3704
3705         const canvasPos = screenToCanvas(e.clientX, e.clientY);
3706         const elemId = AppState.draggingElement;
3707         const elemData = AppState.elements[elemId];
3708         if (!elemData) return;
3709
3710         const newX = canvasPos.x - AppState.dragOffset.x;
3711         const newY = canvasPos.y - AppState.dragOffset.y;
3712         const dx = newX - elemData.x;
3713         const dy = newY - elemData.y;
3714
3715         // если выделено несколько
3716         const group = AppState.selectedElements && AppState.selectedElements.length >
1
3717             ? AppState.selectedElements
3718             : [elemId];
3719
3720         for (const id of group) {
3721             const elData = AppState.elements[id];
3722             if (!elData) continue;
3723             elData.x += dx;
3724             elData.y += dy;
3725             const el = document.getElementById(id);
3726             if (el) {
3727                 el.style.left = elData.x + 'px';
3728                 el.style.top = elData.y + 'px';
3729             }
3730         }
3731
3732         Connections.drawConnections();
3733     },
3734
3735     /**
3736      * Обновление входов формулы
3737      */
3738     updateFormulaInputs(elemId, inputCount) {
3739         const elem = document.getElementById(elemId);
3740         if (!elem) return;
3741
3742         const portsLeft = elem.querySelector('.ports-left');
3743         if (!portsLeft) return;
3744
3745         AppState.connections = AppState.connections.filter(c => {
3746             if (c.toElement === elemId && c.toPort.startsWith('in-')) {
3747                 const portNum = parseInt(c.toPort.split('-')[1], 10);
3748                 return portNum < inputCount;
3749             }
3750             return true;
3751         });
3752
3753         let inputsHTML = '';
3754         for (let i = 0; i < inputCount; i++) {
3755             inputsHTML += `
3756                 <div class="port input any-port"
3757                     data-port="in-${i}"
3758                     data-element="${elemId}"
3759                     data-signal-type="${SIGNAL_TYPE.ANY}"
3760                     title="in${i} (Любой)">
```

```
3761             </div>
3762         `;
3763     }
3764     portsLeft.innerHTML = inputsHTML;
3765
3766     portsLeft.querySelectorAll('.port').forEach(port =>
3767         Connections.setupPortHandlers(port)
3768     );
3769
3770     Connections.drawConnections();
3771 },
3772
3773 /**
3774 * Рассчитать оптимальный размер элемента на основе количества портов
3775 */
3776 calculateOptimalHeight(elemId, inputCount, outputCount = 1) {
3777     const elem = AppState.elements[elemId];
3778     if (!elem) return null;
3779
3780     const config = ELEMENT_TYPES[elem.type];
3781     if (!config || !config.resizable) return null;
3782
3783     // Базовая высота
3784     let baseHeight = config.minLength || 60;
3785
3786     // Каждый порт требует примерно 25-30px высоты
3787     const portSpacing = 28;
3788     const maxPorts = Math.max(inputCount, outputCount);
3789
3790     // Добавляем высоту для портов (кроме первого, который уже в baseHeight)
3791     const additionalHeight = (maxPorts - 1) * portSpacing;
3792     const newHeight = Math.max(baseHeight, baseHeight + additionalHeight);
3793
3794     return newHeight;
3795 },
3796
3797 /**
3798 * Обновление размера элемента при изменении портов
3799 */
3800 updateElementSize(elemId) {
3801     const elem = document.getElementById(elemId);
3802     const elemData = AppState.elements[elemId];
3803
3804     if (!elem || !elemData) return;
3805
3806     const config = ELEMENT_TYPES[elemData.type];
3807     if (!config || !config.resizable) return;
3808
3809     let inputCount = 0;
3810     let outputCount = config.outputs || 1;
3811
3812     // Определяем количество входов
3813     if (elemData.type === 'and' || elemData.type === 'or' || elemData.type ===
3814     'formula') {
3815         inputCount = elemData.props.inputCount || config.inputs || 2;
3816     } else {
3817         inputCount = config.inputs || 0;
3818     }
3819
3820     // Рассчитываем новую высоту
3821     const newHeight = this.calculateOptimalHeight(elemId, inputCount,
3822     outputCount);
3823
3824     if (newHeight && newHeight !== elemData.height) {
3825         elemData.height = newHeight;
3826     }
3827 }
```

```
3824         elem.style.height = `${newHeight}px`;
3825
3826         // Перерисовываем соединения, т.к. изменился размер элемента
3827         Connections.drawConnections();
3828         Viewport.updateMinimap();
3829     }
3830 }
3831
3832 };
3833
3834 modal.js:
3835 /**
3836  * Модуль модальных окон
3837  */
3838
3839 const Modal = {
3840     /**
3841      * Инициализация модальных окон
3842      */
3843     init() {
3844         // Модальное окно свойств элемента
3845         document.getElementById('modal-save').addEventListener('click', () => {
3846             this.saveElementProperties();
3847         });
3848
3849         document.getElementById('modal-cancel').addEventListener('click', () => {
3850             this.hideModal('modal-overlay');
3851         });
3852
3853         document.getElementById('modal-overlay').addEventListener('click', (e) => {
3854             if (e.target.id === 'modal-overlay') {
3855                 this.hideModal('modal-overlay');
3856             }
3857         });
3858
3859         // Модальное окно свойств проекта
3860         document.getElementById('project-modal-save').addEventListener('click', () =>
3861     {
3862         this.saveProjectProperties();
3863     });
3864
3865         document.getElementById('project-modal-cancel').addEventListener('click', () => {
3866             this.hideModal('project-modal-overlay');
3867         });
3868
3869         document.getElementById('project-modal-overlay').addEventListener('click', (e) => {
3870             if (e.target.id === 'project-modal-overlay') {
3871                 this.hideModal('project-modal-overlay');
3872             }
3873         });
3874     },
3875
3876     /**
3877      * Показать модальное окно
3878      */
3879     showModal(modalId) {
3880         document.getElementById(modalId).style.display = 'flex';
3881     },
3882
3883     /**
3884      * Скрыть модальное окно
3885      */
3886 }
```

```
3886     hideModal(modalId) {
3887         document.getElementById(modalId).style.display = 'none';
3888     },
3889
3890     /**
3891      * Показать свойства элемента
3892      */
3893     showPropertiesModal(elemId) {
3894         const elemData = AppState.elements[elemId];
3895         const elemType = elemData.type;
3896         const props = elemData.props;
3897         const config = ELEMENT_TYPES[elemType];
3898
3899         const modalOverlay = document.getElementById('modal-overlay');
3900         const modalTitle = document.getElementById('modal-title');
3901         const modalContent = document.getElementById('modal-content');
3902
3903         modalTitle.textContent = `Свойства: ${config.name}`;
3904
3905         let contentHTML = '';
3906
3907         if (elemType === 'input-signal') {
3908             const signalType = props.signalType || SIGNAL_TYPE.NUMBER;
3909
3910             contentHTML = `
3911                 <div class="modal-row">
3912                     <label>Название сигнала:</label>
3913                     <input type="text" id="prop-name" value="${props.name || ''}"
placeholder="Например: 10LBA..." />
3914                     <small style="color:#999;">
3915                         Поиск по маске через * (например: *MAA*CP*)
3916                     </small>
3917                     <div id="signal-filter-results"
3918                         style="max-height:160px; overflow-y:auto; background:#0f3460; border-
radius:5px; margin-top:6px; display:none;">
3919                         </div>
3920                     </div>
3921
3922                     <div class="modal-row">
3923                         <label>Описание сигнала:</label>
3924                         <textarea id="prop-description" readonly>${props.description || ''}</textarea>
3925                     </div>
3926
3927                     <div class="modal-row">
3928                         <label>Тип сигнала:</label>
3929                         <select id="prop-signal-type">
3930                             <option value="${SIGNAL_TYPE.NUMBER}" ${signalType === SIGNAL_TYPE.NUMBER ?
3931 'selected' : ''}>Числовой</option>
3932                             <option value="${SIGNAL_TYPE.LOGIC}" ${signalType === SIGNAL_TYPE.LOGIC ?
3933 'selected' : ''}>Логический</option>
3934                         </select>
3935                     </div>
3936             `;
3937
3938             // ВАЖНО: обработчики можно навесить только после того, как модалка вставила HTML в
3939             // DOM.
3940             // Поэтому ниже мы добавим "хуки" после того, как modalContent.innerHTML применится.
3941             // (Смотри пункт 2 – небольшая вставка в конце showPropertiesModal)
3942         } else if (elemType === 'if') {
3943             contentHTML = `
3944                 <div class="modal-row">
3945                     <label>Оператор сравнения:</label>
3946                     <select id="prop-operator">
3947                         <option value="=" ${props.operator === '=' ? 'selected' : ''}
3948                         >= (равно)</option>
```

```
3945          <option value=>" ${props.operator === '>' ? 'selected' : ''} `>
3946      >>  (больше)</option>           <option value=<" ${props.operator === '<' ? 'selected' : ''} `>
3947      ><  (меньше)</option>           <option value=>= ${props.operator === '>=' ? 'selected' : ''} `>= (больше или равно)</option>
3948          <option value=<= ${props.operator === '<=' ? 'selected' : ''} `>= (меньше или равно)</option>
3949          <option value!= ${props.operator === '!=' ? 'selected' : ''} `>! (не равно)</option>
3950          </select>
3951      </div>
3952      `;
3953  } else if (elementType === 'and' || elementType === 'or') {
3954      contentHTML =
3955          <div class="modal-row">
3956              <label>Количество входов:</label>
3957              <input type="number" id="prop-input-count" value="$
{props.inputCount || 2}" min="2" max="10">
3958          </div>
3959          <div class="modal-row">
3960              <p style="color: #aaa; font-size: 12px;">
3961                  Измените количество входных портов для этого логического
3962                  элемента.
3963                  Лишние соединения будут автоматически удалены.
3964          </p>
3965      </div>
3966  } else if (elementType === 'const') {
3967      contentHTML =
3968          <div class="modal-row">
3969              <label>Значение:</label>
3970              <input type="number" id="prop-value" value="${props.value ?? 0}"
3971 step="any">
3972          </div>
3973      `;
3974  } else if (elementType === 'group') {
3975      contentHTML =
3976          <div class="modal-row">
3977              <label>Название группы:</label>
3978              <input type="text" id="prop-title" value="${props.title || 'Группа'}">
3979          </div>;
3980  }
3981
3982  else if (elementType === 'formula') {
3983      let signalsHTML = '';
3984      AppState.connections.forEach(conn => {
3985          if (conn.toElement === elemId) {
3986              const fromElem = AppState.elements[conn.fromElement];
3987              if (fromElem) {
3988                  const signalName = fromElem.props?.name || fromElem.id;
3989                  signalsHTML += `<div class="signal-item" data-signal="$
{signalName}">${signalName} (${conn.toPort})</div>`;
3990                  }
3991              }
3992          });
3993
3994  contentHTML =
3995      <div class="modal-row">
3996          <label>Количество входов:</label>
3997          <input type="number" id="prop-input-count" value="$
{props.inputCount || 2}" min="1" max="10">
3998          </div>
3999          <div class="modal-row">
```

```
4000             <label>Входные сигналы (двойной клик для вставки):</label>
4001             <div class="signal-list" id="signal-list">
4002                 ${signalsHTML} || '<div style="color:#888;padding:5px;">Нет
4003                 подключённых сигналов</div>'
4004             </div>
4005             <div class="modal-row">
4006                 <label>Выражение:</label>
4007                 <textarea id="prop-expression">${props.expression} || ''</
4008                 textarea>
4009             </div>
4010         `;
4011     } else if (elemType === 'output') {
4012         contentHTML =
4013             <div class="modal-row">
4014                 <label>Название выхода:</label>
4015                 <input type="text" id="prop-label" value="${props.label} ||
4016                 'Выход'}">
4017             </div>
4018             <div class="modal-row">
4019                 <label>Группировка (опционально):</label>
4020                 <input type="text" id="prop-output-group" value="$
4021                 {props.outputGroup} || ''" placeholder="для логической группировки выходов">
4022             </div>
4023         `;
4024     }
4025
4026     modalContent.innerHTML = contentHTML;
4027     // --- post init handlers (когда DOM модалки уже существует) ---
4028     if (elemType === 'input-signal') {
4029         const input = document.getElementById('prop-name');
4030         const results = document.getElementById('signal-filter-results');
4031         const descField = document.getElementById('prop-description');
4032
4033         let timer = null;
4034
4035         const renderList = (items) => {
4036             if (!items || items.length === 0) {
4037                 results.innerHTML = '<div style="color:#666;padding:6px;">Нет совпадений</
4038                 div>';
4039                 results.style.display = 'block';
4040                 return;
4041             }
4042
4043             results.innerHTML = items.map(s => `
4044                 <div class="signal-result-item"
4045                     style="padding:6px 8px; cursor:pointer; border-bottom:1px solid
4046                     rgba(255,255,255,0.08);">
4047                     <div style="font-weight:600;">${s.Tagname}</div>
4048                     <div style="color:#aaa; font-size:11px;">${s.Description} || ''</div>
4049                 </div>
4050             `).join('');
4051
4052             results.style.display = 'block';
4053
4054             results.querySelectorAll('.signal-result-item').forEach((div, i) => {
4055                 div.addEventListener('click', () => {
4056                     const chosen = items[i];
4057                     input.value = chosen.Tagname;
4058                     descField.value = chosen.Description || '';
4059                     results.style.display = 'none';
4060                 });
4061             });
4062         };
4063     };
4064 }
```

```
4059
4060     const search = async () => {
4061         const mask = (input.value || '').trim();
4062
4063         // Показываем список только если пользователь реально использует маску
4064         if (!mask.includes('*')) {
4065             results.style.display = 'none';
4066             return;
4067         }
4068
4069         results.innerHTML = '<div style="color:#666;padding:6px;">Поиск...</div>';
4070         results.style.display = 'block';
4071
4072         try {
4073             // В settings.js должен быть метод Settings.fetchSignals(mask, limit)
4074             const data = await Settings.fetchSignals(mask, 50);
4075             renderList(data.items || []);
4076         } catch (e) {
4077             results.innerHTML = '<div style="color:#666;padding:6px;">Ошибка загрузки
сигналов</div>';
4078             results.style.display = 'block';
4079             console.error(e);
4080         }
4081     };
4082
4083     input.addEventListener('input', () => {
4084         clearTimeout(timer);
4085         timer = setTimeout(search, 200); // debounce
4086     });
4087
4088     // опционально: закрывать список кликом вне
4089     document.addEventListener('mousedown', (e) => {
4090         if (!results.contains(e.target) && e.target !== input) {
4091             results.style.display = 'none';
4092         }
4093     }, { once: true });
4094
4095     modalOverlay.dataset.elementId = elemId;
4096     this.showModal('modal-overlay');
4097
4098     // Обработчик вставки сигналов для формулы
4099     if (elemType === 'formula') {
4100         document.querySelectorAll('.signal-item').forEach(item => {
4101             item.addEventListener('dblclick', () => {
4102                 const signal = item.dataset.signal;
4103                 const textarea = document.getElementById('prop-expression');
4104                 textarea.value += signal;
4105                 textarea.focus();
4106             });
4107         });
4108     },
4109
4110     /**
4111      * Сохранить свойства элемента
4112      */
4113
4114     /**
4115      * Сохранить свойства элемента
4116      */
4117     saveElementProperties() {
4118         try {
4119             const modalOverlay = document.getElementById('modal-overlay');
4120             const elemId = modalOverlay.dataset.elementId;
4121             const elemData = AppState.elements[elemId];
4122             const elemType = elemData.type;
```

```
4123 const elem = document.getElementById(elemId);
4124
4125 if (elemType === 'input-signal') {
4126     const name = document.getElementById('prop-name').value || 'Сигнал';
4127     const description = document.getElementById('prop-description').value
4128     || '';
4129     const signalType = document.getElementById('prop-signal-type').value;
4130     const oldSignalType = elemData.props.signalType;
4131     elemData.props.name = name;
4132     elemData.props.description = description;
4133     elemData.props.signalType = signalType;
4134
4135     if (oldSignalType !== signalType) {
4136         AppState.connections = AppState.connections.filter(conn => {
4137             if (conn.fromElement === elemId) {
4138                 const toPortIndex = parseInt(conn.toPort.split('-')[1]);
4139                 const inputType = getInputPortType(conn.toElement,
4140                     toPortIndex);
4141                 return areTypesCompatible(signalType, inputType);
4142             }
4143             return true;
4144         });
4145
4146         const { html } = Elements.createElementHTML(
4147             elemType, elemId, elemData.x, elemData.y, elemData.props,
4148             elemData.width, elemData.height
4149         );
4150         elem.outerHTML = html;
4151
4152         Elements.setupElementHandlers(elemId);
4153         Connections.drawConnections();
4154     } else if (elemType === 'if') {
4155         const operator = document.getElementById('prop-operator').value;
4156         elemData.props.operator = operator;
4157         const symbol = elem.querySelector('.element-symbol');
4158         if (symbol) symbol.textContent = operator;
4159
4160     } else if (elemType === 'const') {
4161         const value = parseFloat(document.getElementById('prop-value').value)
4162         || 0;
4163         elemData.props.value = value;
4164         const symbol = elem.querySelector('.element-symbol');
4165         if (symbol) symbol.textContent = String(value);
4166
4167     } else if (elemType === 'formula') {
4168         const expression = document.getElementById('prop-expression').value;
4169         const inputCount = parseInt(document.getElementById('prop-input-
4170         count').value) || 2;
4171
4172         elemData.props.expression = expression;
4173         elemData.props.inputCount = inputCount;
4174
4175         const symbol = elem.querySelector('.element-symbol');
4176         if (symbol) {
4177             symbol.textContent = expression.length > 12 ? `$
4178             {expression.slice(0, 12)}...` : (expression || 'f(x)');
4179         }
4180
4181         Elements.updateFormulaInputs(elemId, inputCount);
4182         Elements.updateElementSize(elemId); // ← Добавляем это
4183     } else if (elemType === 'and' || elemType === 'or') {
4184         const inputCount = parseInt(document.getElementById('prop-input-
4185         count').value) || 2;
```

```
4181         elemData.props.inputCount = inputCount;
4182
4183         Elements.updateLogicGateInputs(elemId, inputCount);
4184         Elements.updateElementSize(elemId); // ← Добавляем это
4185
4186         const symbol = elem.querySelector('.element-symbol');
4187         if (symbol) {
4188             symbol.textContent = elemType === 'and' ? 'Λ' : '∨';
4189         }
4190
4191     } else if (elemType === 'output') {
4192         const label = document.getElementById('prop-label').value || 'Выход';
4193         const outputGroup = document.getElementById('prop-output-group').value
4194         || '';
4195
4196         elemData.props.label = label;
4197         elemData.props.outputGroup = outputGroup;
4198
4199         const symbol = elem.querySelector('.element-symbol');
4200         if (symbol) symbol.textContent = label;
4201     }
4202     else if (elemType === 'group') {
4203         const title = document.getElementById('prop-title').value || 'Группа';
4204         elemData.props.title = title;
4205         const titleEl = elem.querySelector('.group-title');
4206         if (titleEl) titleEl.textContent = title;
4207     }
4208
4209     this.hideModal('modal-overlay');
4210
4211 } catch (error) {
4212     console.error('✖ Ошибка при сохранении свойств:', error);
4213     alert('Ошибка сохранения: ' + error.message);
4214 },
4215 /**
4216 * Показать свойства проекта
4217 */
4218 showProjectPropertiesModal() {
4219     const content = document.getElementById('project-modal-content');
4220     const project = AppState.project;
4221
4222     // Генерируем HTML для списка выходов только если модуль загружен
4223     let outputsHtml = '';
4224     if (typeof Outputs !== 'undefined' && AppState.outputs) {
4225         const logicalOutputsHtml = AppState.outputs.logical.length > 0
4226             ? AppState.outputs.logical.map(output =>
4227                 <div class="output-item"
4228                     data-element-id="${output.elementId}"
4229                     onmouseenter="Outputs.highlightOutput('${output.elementId}', true)"
4230                     onmouseleave="Outputs.highlightOutput('${output.elementId}', false)"
4231                     onclick="Outputs.navigateToOutput('${output.elementId}')";
4232                 Modal.hideModal('project-modal-overlay');"
4233                     <span class="output-icon">>${output.portLabel === 'Да' ? '✓' : '✗'}</span>
4234                     <span class="output-name">>${output.elementName}</span>
4235                     <span class="output-port">>→ ${output.portLabel}</span>
4236                 </div>
4237             ).join('')
4238             : '<div class="no-outputs">Нет логических выходов</div>';
4239
4240     const numericOutputsHtml = AppState.outputs.numeric.length > 0
```

```
4241             ? AppState.outputs.numeric.map(output => `<div class="output-item numeric" data-element-id="${output.elementId}" onmouseenter="Outputs.highlightOutput('${output.elementId}')", onmouseleave="Outputs.highlightOutput('${output.elementId}')", onclick="Outputs.navigateToOutput('${output.elementId}')";`  
4242             true)"  
4243             false)"  
4244             Modal.hideModal('project-modal-overlay');`>  
4245             <span class="output-icon"></span>  
4246             <span class="output-name">${output.elementName}</span>  
4247             <span class="output-port">→ значение</span>  
4248             `)>.join('')  
4249             : '<div class="no-outputs">Нет числовых выходов</div>';  
4250  
4251  
4252  
4253  
4254         outputsHtml = `<div class="modal-row">  
4255             <label>Выходные сигналы схемы:</label>  
4256             <div class="outputs-container">  
4257                 <div class="outputs-section">  
4258                     <div class="outputs-section-title">  
4259                         <span class="section-icon"> </span>  
4260                         Логические выходы (${AppState.outputs.logical.length})  
4261                     </div>  
4262                     <div class="outputs-list">  
4263                         ${logicalOutputsHtml}  
4264                     </div>  
4265                 </div>  
4266                 <div class="outputs-section">  
4267                     <div class="outputs-section-title">  
4268                         <span class="section-icon"> </span>  
4269                         Числовые выходы (${AppState.outputs.numeric.length})  
4270                     </div>  
4271                     <div class="outputs-list">  
4272                         ${numericOutputsHtml}  
4273                     </div>  
4274                 </div>  
4275             </div>  
4276             <div class="outputs-hint">  
4277                  Выходами автоматически становятся элементы, чьи выходные  
4278                 порты не подключены к другим элементам.  
4279                 Кликните на выход, чтобы перейти к нему на схеме.  
4280             </div>  
4281         </div>  
4282     `;  
4283 }  
4284  
4285     content.innerHTML = `<div class="modal-row">  
4286         <label>Код проекта:</label>  
4287         <input type="text" id="project-code" value="${project.code || ''}"  
4288         placeholder="Уникальный идентификатор">  
4289     </div>  
4290  
4291     <div class="modal-row">  
4292         <label>Тип проекта:</label>  
4293         <div class="project-type-selector">  
4294             <div class="project-type-btn ${project.type === PROJECT_TYPE.PARAMETER ? 'active' : ''}" data-type="${PROJECT_TYPE.PARAMETER}">  
4295                 <div class="type-icon"> </div>  
4296                 <div class="type-name">Параметр</div>  
4297                 <div class="type-desc">Вычисляемое значение</div>  
4298             </div>  
4299             <div class="project-type-btn ${project.type ===
```

```
PROJECT_TYPE.RULE ? 'active' : '')" data-type="${PROJECT_TYPE.RULE}">
4300             <div class="type-icon"></div>
4301             <div class="type-name">Правило</div>
4302             <div class="type-desc">Логическое условие</div>
4303         </div>
4304     </div>
4305 </div>
4306
4307     <div id="parameter-fields" class="conditional-fields ${project.type ===
PROJECT_TYPE.PARAMETER ? 'visible' : ''}">
4308         <div class="modal-row">
4309             <label>Размерность:</label>
4310             <input type="text" id="project-dimension" value="$
{project.dimension || ''}" placeholder="Например: м/с, кг, °C">
4311             </div>
4312         </div>
4313
4314         <div id="rule-fields" class="conditional-fields ${project.type ===
PROJECT_TYPE.RULE ? 'visible' : ''}">
4315             <div class="modal-row">
4316                 <label>Возможная причина:</label>
4317                 <textarea id="project-possible-cause" placeholder="Описание
возможной причины срабатывания правила">${project.possibleCause || ''}</textarea>
4318             </div>
4319             <div class="modal-row">
4320                 <label>Методические указания:</label>
4321                 <textarea id="project-guidelines" placeholder="Инструкции и
рекомендации при срабатывании правила">${project.guidelines || ''}</textarea>
4322             </div>
4323         </div>
4324
4325         ${outputsHtml}
4326     `;
4327
4328     // Обработчики переключения типа
4329     content.querySelectorAll('.project-type-btn').forEach(btn => {
4330         btn.addEventListener('click', () => {
4331             content.querySelectorAll('.project-type-btn').forEach(b =>
b.classList.remove('active'));
4332             btn.classList.add('active');
4333
4334             const type = btn.dataset.type;
4335             document.getElementById('parameter-
fields').classList.toggle('visible', type === PROJECT_TYPE.PARAMETER);
4336             document.getElementById('rule-fields').classList.toggle('visible',
type === PROJECT_TYPE.RULE);
4337         });
4338     });
4339
4340     this.showModal('project-modal-overlay');
4341 },
4342
4343 /**
4344 * Сохранить свойства проекта
4345 */
4346 saveProjectProperties() {
4347     const activeTypeBtn = document.querySelector('.project-type-btn.active');
4348     const type = activeTypeBtn ? activeTypeBtn.dataset.type :
PROJECT_TYPE.PARAMETER;
4349
4350     AppState.project.code = document.getElementById('project-code').value;
4351     AppState.project.type = type;
4352
4353     if (type === PROJECT_TYPE.PARAMETER) {
4354         AppState.project.dimension = document.getElementById('project-
```

```
dimension').value;
4355     AppState.project.possibleCause = '';
4356     AppState.project.guidelines = '';
4357 } else {
4358     AppState.project.dimension = '';
4359     AppState.project.possibleCause = document.getElementById('project-
possible-cause').value;
4360     AppState.project.guidelines = document.getElementById('project-
guidelines').value;
4361 }
4362
4363     this.hideModal('project-modal-overlay');
4364 }
4365 };
4366
4367 output.js:
4368 /**
4369 * Модуль управления выходными сигналами
4370 */
4371
4372 const Outputs = {
4373 /**
4374 * Обновление статуса выходных элементов
4375 * Вызывается при каждом изменении схемы
4376 */
4377 updateOutputStatus() {
4378     this.clearAllOutputHighlights();
4379     AppState.outputs.logical = [];
4380     AppState.outputs.numeric = [];
4381     updateFrameChildren();
4382
4383     // Обработка элементов-выходов
4384     Object.values(AppState.elements).forEach(elem => {
4385         if (!elem || elem.type !== 'output') return;
4386
4387         // Проверяем, к чему подключен вход этого выхода
4388         const inputConns = AppState.connections.filter(c =>
4389             c.toElement === elem.id && c.toPort === 'in-0'
4390         );
4391
4392         // Каждое соединение к выходу – это отдельный выход
4393         inputConns.forEach((conn, index) => {
4394             const fromElem = AppState.elements[conn.fromElement];
4395             if (!fromElem) return;
4396
4397             const outputType = conn.signalType;
4398             const outputInfo = {
4399                 id: `${elem.id}_conn_${index}`,
4400                 elementId: elem.id,
4401                 sourceElementId: conn.fromElement,
4402                 sourcePort: conn.fromPort,
4403                 portIndex: 0,
4404                 portId: 'in-0',
4405                 type: outputType,
4406                 label: elem.props?.label || 'Выход',
4407                 elementType: 'output',
4408                 elementName: elem.props?.label || 'Выход',
4409                 name: elem.props?.label || 'Выход'
4410             };
4411
4412             if (outputType === SIGNAL_TYPE.LOGIC) {
4413                 AppState.outputs.logical.push(outputInfo);
4414             } else if (outputType === SIGNAL_TYPE.NUMBER) {
4415                 AppState.outputs.numeric.push(outputInfo);
4416             }
4417         });
4418     });
4419 }
```

```
4417          // Подсветим входной порт
4418          this.highlightOutputPort(elem.id, 0, outputType);
4419      });
4420  });
4421  });
4422  this.updateOutputCounter();
4423 },
4424 /**
4425 * Очистка всех выделений выходов
4426 */
4427 clearAllOutputHighlights() {
4428     document.querySelectorAll('.port.output-active').forEach(port => {
4429         port.classList.remove('output-active');
4430     });
4431
4432     document.querySelectorAll('.element.has-output').forEach(elem => {
4433         elem.classList.remove('has-output');
4434     });
4435
4436     document.querySelectorAll('.element.output-ambiguous').forEach(el =>
4437 el.classList.remove('output-ambiguous'));
4438     document.querySelectorAll('.element.output-missing').forEach(el =>
4439 el.classList.remove('output-missing'));
4440 },
4441 /**
4442 * Выделение выходного порта
4443 */
4444 highlightOutputPort(elemId, portIndex, portType) {
4445     const elem = document.getElementById(elemId);
4446     if (!elem) return;
4447
4448     const port = elem.querySelector(`.port.output[data-port="out-${portIndex}]`);
4449     if (port) {
4450         port.classList.add('output-active');
4451     }
4452
4453     // Добавляем класс элементу (даёт общий визуал)
4454     elem.classList.add('has-output');
4455 },
4456 /**
4457 * Обновление счётчика выходов в меню
4458 */
4459 updateOutputCounter() {
4460     const counter = document.getElementById('output-counter');
4461     if (counter) {
4462         const total = AppState.outputs.logical.length +
4463 AppState.outputs.numeric.length;
4464         counter.textContent = total;
4465         counter.style.display = total > 0 ? 'inline-block' : 'none';
4466     }
4467 },
4468 /**
4469 * Получить все выходы для сохранения в проект
4470 */
4471 getOutputsForSave() {
4472     // Сохраняем информацию о frame/inner для рамок
4473     return {
4474         logical: AppState.outputs.logical.map(o => ({
4475             id: o.id,
4476             elementId: o.elementId,
```

```
4479         frameId: o.frameId || null,
4480         innerElementId: o.innerElementId || null,
4481         portIndex: o.portIndex ?? o.innerPortIndex ?? null,
4482         portLabel: o.label
4483     })),
4484     numeric: AppState.outputs.numeric.map(o => ({
4485         id: o.id,
4486         elementId: o.elementId,
4487         frameId: o.frameId || null,
4488         innerElementId: o.innerElementId || null,
4489         portIndex: o.portIndex ?? o.innerPortIndex ?? null,
4490         portLabel: o.label
4491     }))
4492   );
4493 },
4494 /**
4495 * Подсветить конкретный выход (при наведении в списке)
4496 */
4497 highlightOutput(elementId, highlight = true) {
4498     const elem = document.getElementById(elementId);
4499     if (elem) {
4500         if (highlight) {
4501             elem.classList.add('output-highlighted');
4502         } else {
4503             elem.classList.remove('output-highlighted');
4504         }
4505     }
4506 },
4507 /**
4508 * Перейти к элементу выхода на схеме (elementId – фокусируемый элемент; для рамок
4509 это id рамки)
4510 */
4511 navigateToOutput(elementId) {
4512     const elemData = AppState.elements[elementId];
4513     if (!elemData) return;
4514
4515     // Центрируем viewport на элементе
4516     const container = document.getElementById('workspace-container');
4517     const rect = container.getBoundingClientRect();
4518
4519     const centerX = elemData.x + elemData.width / 2;
4520     const centerY = elemData.y + elemData.height / 2;
4521
4522     AppState.viewport.panX = rect.width / 2 - centerX * AppState.viewport.zoom;
4523     AppState.viewport.panY = rect.height / 2 - centerY * AppState.viewport.zoom;
4524
4525     Viewport.updateTransform();
4526
4527     // Выделяем элемент
4528     Elements.selectElement(elementId);
4529
4530     // Временная подсветка
4531     this.highlightOutput(elementId, true);
4532     setTimeout(() => this.highlightOutput(elementId, false), 2000);
4533   }
4534 };
4535 };
4536
4537 project.js:
4538 /**
4539 * Модуль управления проектом (сохранение, загрузка)
4540 */
4541
4542 const Project = {
```

```
4543     /**
4544      * Инициализация
4545      */
4546      /**
4547      * Инициализация
4548      */
4549 init() {
4550     document.getElementById('btn-new').addEventListener('click', () =>
4551       this.newProject());
4552     document.getElementById('btn-save').addEventListener('click', () =>
4553       this.saveProject());
4554     document.getElementById('btn-load').addEventListener('click', () =>
4555       this.openProjectListModal());
4556     document.getElementById('btn-project-settings').addEventListener('click', () => {
4557       Modal.showProjectPropertiesModal();
4558     });
4559
4560     // Работа с модалкой выбора проекта
4561     this.projectList = [];
4562     this.filteredProjectList = [];
4563     this.selectedProjectFilename = null;
4564
4565     document.getElementById('project-cancel').addEventListener('click', () =>
4566       this.closeProjectListModal());
4567     document.getElementById('project-refresh').addEventListener('click', () =>
4568       this.refreshProjectList());
4569
4570     document.getElementById('project-load').addEventListener('click', () => {
4571       if (this.selectedProjectFilename) {
4572         this.loadProjectFromList(this.selectedProjectFilename);
4573       }
4574     });
4575
4576     /**
4577      * Новый проект
4578      */
4579     newProject() {
4580       if (Object.keys(AppState.elements).length > 0) {
4581         if (!confirm('Создать новый проект? Несохранённые изменения будут
4582 потеряны.')) {
4583           return;
4584         }
4585       }
4586       document.getElementById('workspace').innerHTML = '';
4587       document.getElementById('connections-svg').innerHTML = '';
4588
4589       setState();
4590       Viewport.updateTransform();
4591     },
4592
4593     /**
4594      * Запрос имени файла и загрузка с сервера
4595      */
4596     async loadProjectPrompt() {
4597       const filename = window.prompt(
4598         "Введите имя файла проекта для загрузки (с сервера). Пример:
4599         scheme_logic.json",
4600         AppState.project.code ? `${AppState.project.code}_$` +
4601         `{AppState.project.type}.json` : "scheme_type.json"
4602       );
4603     }
4604   }
4605 }
```

```
4600 );
4601     if (!filename) return; // Отмена
4602
4603     try {
4604         // Используем обертку из Settings.js для запроса к /api/project/load
4605         const data = await Settings.loadProject(filename);
4606
4607         // Если загрузка успешна, вызываем основную функцию обработки данных
4608         this._processLoadedData(data);
4609         alert(`Проект "${filename}" успешно загружен с сервера.`);
4610
4611     } catch (error) {
4612         console.error('Ошибка загрузки проекта:', error);
4613         alert(`Ошибка загрузки проекта: ${error.message}`);
4614     }
4615 },
4616
4617 /**
4618 * Сохранение проекта
4619 */
4620
4621 async saveProject() { // !!! Сделать функцию асинхронной (async) !!!
4622     // 1. Проверяем свойства проекта
4623     if (!AppState.project.code) {
4624         Modal.showProjectPropertiesModal();
4625         alert('Пожалуйста, укажите код проекта перед сохранением.');
4626         return;
4627     }
4628
4629     // Обновляем размеры рамок перед сохранением
4630     updateFrameChildren();
4631
4632     // 2. Сборка объекта проекта
4633     const project = {
4634         version: '1.0',
4635         project: AppState.project,
4636         elements: AppState.elements,
4637         connections: AppState.connections,
4638         counter: AppState.elementCounter,
4639         viewport: {
4640             zoom: AppState.viewport.zoom,
4641             panX: AppState.viewport.panX,
4642             panY: AppState.viewport.panY
4643         }
4644     };
4645
4646     const filename = `${AppState.project.code || 'scheme'}_${AppState.project.type}.json`;
4647
4648     // 3. Сохранение на сервер
4649     try {
4650         await Settings.saveProject(filename, project);
4651         alert(`Проект успешно сохранен на сервере как: ${filename}`);
4652     } catch (error) {
4653         console.error('Ошибка сохранения проекта:', error);
4654         alert(`Ошибка сохранения проекта: ${error.message}`);
4655     }
4656 },
4657
4658     async showProjectList() {
4659         try {
4660             const result = await Settings.listProjects(); // нужно реализовать в
settings.js
4661             const list = result.projects || [];
4662         }
```

```
4663     if (list.length === 0) {
4664         alert('Проекты в папке не найдены.');
4665         return;
4666     }
4667
4668     const choice = window.prompt(
4669         'Список проектов:\n' + list.map((p, i) => `${i + 1}. ${p.code} ||
4670 p.filename} - ${p.description}`).join('\n') +
4671         '\n\nВведите номер проекта для загрузки:',
4672         '1'
4673     );
4674     const index = parseInt(choice, 10) - 1;
4675     if (isNaN(index) || !list[index]) return;
4676
4677     await this.loadProjectByFilename(list[index].filename);
4678 } catch (error) {
4679     console.error(error);
4680     alert('Не удалось получить список проектов: ' + error.message);
4681 }
4682
4683 async loadProjectByFilename(filename) {
4684     try {
4685         const data = await Settings.loadProject(filename);
4686         this._processLoadedData(data);
4687         alert(`Проект "${filename}" загружен.`);
4688     } catch (error) {
4689         console.error(error);
4690         alert('Ошибка загрузки проекта: ' + error.message);
4691     }
4692 }
4693
4694 openProjectListModal() {
4695     const modal = document.getElementById('modal-project-list');
4696     modal.classList.remove('hidden');
4697     document.body.classList.add('modal-open'); // если есть такой класс для блокировки
 скролла
4698     this.refreshProjectList();
4699 },
4700
4701 closeProjectListModal() {
4702     const modal = document.getElementById('modal-project-list');
4703     modal.classList.add('hidden');
4704     document.body.classList.remove('modal-open');
4705 },
4706
4707 async refreshProjectList() {
4708     const tbody = document.getElementById('project-list-body');
4709     tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Загрузка...</td></
 tr>`;
4710     try {
4711         const result = await Settings.listProjects();
4712         this.projectList = result.projects || [];
4713         this.filteredProjectList = [...this.projectList];
4714         this.renderProjectList();
4715     } catch (err) {
4716         console.error(err);
4717         tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ошибка: $ {err.message}</td></tr>`;
4718     }
4719 },
4720
4721 renderProjectList() {
4722     const tbody = document.getElementById('project-list-body');
4723     const loadBtn = document.getElementById('project-load');
```

```
4724     loadBtn.disabled = true;
4725     this.selectedProjectFilename = null;
4726
4727     if (!this.filteredProjectList.length) {
4728         tbody.innerHTML = `<tr><td colspan="4" class="project-list__empty">Ничего не
найдено</td></tr>`;
4729         return;
4730     }
4731
4732     tbody.innerHTML = '';
4733     this.filteredProjectList.forEach((item) => {
4734         const tr = document.createElement('tr');
4735         tr.innerHTML =
4736             `<td>${item.filename}</td>
4737             <td>${item.code || ''}</td>
4738             <td>${item.description || ''}</td>
4739             <td>${item.type || ''}</td>
4740         `;
4741         tr.addEventListener('click', () => {
4742             this.highlightRow(tr);
4743             this.selectedProjectFilename = item.filename;
4744             loadBtn.disabled = false;
4745         });
4746         tr.addEventListener('dblclick', () => {
4747             this.highlightRow(tr);
4748             this.selectedProjectFilename = item.filename;
4749             loadBtn.disabled = false;
4750             this.loadProjectFromList(item.filename);
4751         });
4752         tbody.appendChild(tr);
4753     });
4754 },
4755
4756 highlightRow(row) {
4757     const tbody = row.parentElement;
4758     [...tbody.children].forEach((tr) => tr.classList.remove('selected'));
4759     row.classList.add('selected');
4760 },
4761
4762
4763 // Фильтр по поисковой строке
4764 filterProjectList(query) {
4765     const q = (query || '').trim().toLowerCase();
4766     if (!q) {
4767         this.filteredProjectList = [...this.projectList];
4768     } else {
4769         this.filteredProjectList = this.projectList.filter((item) => {
4770             return [
4771                 item.filename,
4772                 item.code,
4773                 item.description,
4774                 item.type
4775             ].some((field) => (field || '').toLowerCase().includes(q));
4776         });
4777     }
4778     this.renderProjectList();
4779 },
4780
4781 async loadProjectFromList(filename) {
4782     try {
4783         const data = await Settings.loadProject(filename);
4784         this._processLoadedData(data);
4785         this.closeProjectListModal();
4786         alert(`Проект "${filename}" успешно загружен.`);
4787     } catch (error) {
```

```
4788     console.error(error);
4789     alert('Ошибка загрузки проекта: ' + error.message);
4790   }
4791 },
4792
4793
4794
4795
4796 /**
4797 * Загрузка проекта
4798 */
4799 _processLoadedData(data) {
4800 try {
4801   document.getElementById('workspace').innerHTML = '';
4802   document.getElementById('connections-svg').innerHTML = '';
4803   resetState();
4804
4805   if (data.project) {
4806     AppState.project = { ...AppState.project, ...data.project };
4807   }
4808
4809   AppState.elementCounter = data.counter || 0;
4810
4811   if (data.viewport) {
4812     AppState.viewport.zoom = data.viewport.zoom || 1;
4813     AppState.viewport.panX = data.viewport.panX || 0;
4814     AppState.viewport.panY = data.viewport.panY || 0;
4815   }
4816
4817   const elements = data.elements || {};
4818   Object.values(elements)
4819     .filter(e => e.type === 'output-frame')
4820     .forEach(elemData => {
4821       Elements.addElement(
4822         elemData.type,
4823         elemData.x,
4824         elemData.y,
4825         elemData.props,
4826         elemData.id,
4827         elemData.width,
4828         elemData.height
4829       );
4830     });
4831
4832   Object.values(elements)
4833     .filter(e => e.type !== 'output-frame')
4834     .forEach(elemData => {
4835       Elements.addElement(
4836         elemData.type,
4837         elemData.x,
4838         elemData.y,
4839         elemData.props,
4840         elemData.id,
4841         elemData.width,
4842         elemData.height
4843       );
4844     });
4845
4846   AppState.connections = data.connections || [];
4847   AppState.elementCounter = Math.max(
4848     AppState.elementCounter,
4849     ...Object.values(elements).map(e => e.id || 0)
4850   );
4851
4852   Viewport.updateTransform();
```

```
4853     Connections.drawConnections();
4854     updateFrameChildren();
4855   } catch (e) {
4856     alert('Ошибка обработки данных проекта: ' + e.message);
4857     console.error(e);
4858   }
4859 }
4860 };
4861
4862 settings.js:
4863 const Settings = {
4864   config: null,
4865
4866   async init() {
4867     // тянем настройки (не обязательно, но полезно)
4868     try {
4869       const r = await fetch('/api/settings');
4870       if (r.ok) this.config = await r.json();
4871     } catch (e) {
4872       console.warn('Settings load failed:', e);
4873     }
4874   },
4875
4876   async fetchSignals(mask, limit = 50) {
4877     const url = `/api/signals?q=${encodeURIComponent(mask || '')}&limit=${encodeURIComponent(limit)}`;
4878     const r = await fetch(url);
4879     if (!r.ok) throw new Error('Failed to fetch signals');
4880     return await r.json(); // {items, total}
4881   },
4882   // ... в объекте Settings
4883
4884   async saveProject(filename, projectData) {
4885     if (!filename.endsWith('.json')) {
4886       filename += '.json';
4887     }
4888     const r = await fetch('/api/project/save', {
4889       method: 'POST',
4890       headers: { 'Content-Type': 'application/json' },
4891       body: JSON.stringify({
4892         filename: filename,
4893         content: projectData
4894       })
4895     });
4896     if (!r.ok) throw new Error('Failed to save project');
4897     return r.json();
4898   },
4899
4900   async listProjects() {
4901     const r = await fetch('/api/project/list');
4902     if (!r.ok) throw new Error('Failed to list projects');
4903     return r.json();
4904   },
4905
4906   async loadProject(filename) {
4907     if (!filename.endsWith('.json')) {
4908       filename += '.json';
4909     }
4910     const r = await fetch(`/api/project/load/${encodeURIComponent(filename)}`);
4911     if (!r.ok) {
4912       if (r.status === 404) {
4913         throw new Error(`Project "${filename}" not found (404)`);
4914       }
4915     }
4916     throw new Error('Failed to load project');
4917   }
4918 }
```

```
4917     return r.json();
4918 }
4919
4920 // ...
4921 };
4922
4923 state.js:
4924 /**
4925 * Глобальное состояние приложения
4926 */
4927
4928 const AppState = {
4929     // Элементы схемы
4930     elements: {},
4931     connections: [],
4932     elementCounter: 0,
4933
4934     // Выделение
4935     selectedElement: null,
4936
4937     // Перетаскивание
4938     draggingElement: null,
4939     dragOffset: { x: 0, y: 0 },
4940     isDraggingFromPalette: false,
4941     dragPreview: null,
4942     dragType: null,
4943
4944     // Соединения
4945     connectingFrom: null,
4946     connectingFromType: null,
4947     tempLine: null,
4948
4949     // Resize
4950     resizing: null,
4951
4952     // Viewport (масштабирование и перемещение)
4953     viewport: {
4954         zoom: 1,
4955         panX: 0,
4956         panY: 0,
4957         isPanning: false,
4958         lastMouseX: 0,
4959         lastMouseY: 0
4960     },
4961
4962     // Свойства проекта
4963     project: {
4964         code: '',
4965         type: PROJECT_TYPE.PARAMETER,
4966         // Для параметра
4967         dimension: '',
4968         // Для правила
4969         possibleCause: '',
4970         guidelines: ''
4971     },
4972
4973     // Выходные сигналы (автоматически определяются)
4974     outputs: {
4975         logical: [],    // Логические выходы [{elementId, portIndex, portLabel, ...}]
4976         numeric: []    // Числовые выходы (формулы)
4977     }
4978 };
4979
4980 /**
4981 * Сброс состояния
```

```
4982  */
4983 function resetState() {
4984     AppState.elements = {};
4985     AppState.connections = [];
4986     AppState.elementCounter = 0;
4987     AppState.selectedElement = null;
4988     AppState.draggingElement = null;
4989     AppState.connectingFrom = null;
4990     AppState.tempLine = null;
4991     AppState.resizing = null;
4992
4993     AppState.viewport = {
4994         zoom: 1,
4995         panX: 0,
4996         panY: 0,
4997         isPanning: false,
4998         lastMouseX: 0,
4999         lastMouseY: 0
5000     };
5001
5002     AppState.project = {
5003         code: '',
5004         type: PROJECT_TYPE.PARAMETER,
5005         dimension: '',
5006         possibleCause: '',
5007         guidelines: ''
5008     };
5009
5010     AppState.outputs = {
5011         logical: [],
5012         numeric: []
5013     };
5014 }
5015
5016 utils.js:
5017 /**
5018 * Вспомогательные функции
5019 */
5020
5021 /**
5022 * Генерация уникального ID
5023 */
5024 function generateId() {
5025     AppState.elementCounter++;
5026     return `elem_${AppState.elementCounter}`;
5027 }
5028
5029 function getInputPortType(elementId, portIdentifier) {
5030     const element = AppState.elements[elementId];
5031     if (!element) return SIGNAL_TYPE.ANY;
5032
5033     const config = ELEMENT_TYPES[element.type];
5034     if (!config) return SIGNAL_TYPE.ANY;
5035
5036     let portIndex = portIdentifier;
5037
5038     // Обработка технического порта условия
5039     if (typeof portIdentifier === 'string') {
5040         if (portIdentifier === 'cond-0' && config.hasConditionPort) {
5041             return config.conditionPortType || SIGNAL_TYPE.LOGIC;
5042         }
5043
5044         if (portIdentifier.startsWith('in-')) {
5045             portIndex = parseInt(portIdentifier.split('-')[1], 10);
5046         }
5047     }
5048 }
```

```
5047     }
5048
5049     if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5050         portIndex = 0;
5051     }
5052
5053     // Динамические входы для AND/OR берут тип из конфига
5054     if ((element.type === 'and' || element.type === 'or')) {
5055         return SIGNAL_TYPE.LOGIC; // Логические элементы всегда ожидают LOGIC на
5056         // входе
5057     }
5058
5059     if (element.type === 'formula') {
5060         return SIGNAL_TYPE.ANY;
5061     }
5062
5063     const types = config.inputTypes || [];
5064     if (types.length === 0) return SIGNAL_TYPE.ANY;
5065
5066     if (portIndex < types.length) {
5067         return types[portIndex] || SIGNAL_TYPE.ANY;
5068     }
5069
5070     return types[types.length - 1] || SIGNAL_TYPE.ANY;
5071 }
5072 function getOutputPortType(elementId, portIdentifier) {
5073     const element = AppState.elements[elementId];
5074     if (!element) return SIGNAL_TYPE.ANY;
5075
5076     const config = ELEMENT_TYPES[element.type];
5077     if (!config) return SIGNAL_TYPE.ANY;
5078
5079     let portIndex = portIdentifier;
5080
5081     if (typeof portIdentifier === 'string') {
5082         if (portIdentifier.startsWith('out-')) {
5083             portIndex = parseInt(portIdentifier.split('-')[1], 10);
5084         }
5085     }
5086
5087     if (Number.isNaN(portIndex) || portIndex === null || portIndex === undefined) {
5088         portIndex = 0;
5089     }
5090
5091     const types = config.outputTypes || [];
5092     if (types.length === 0) return SIGNAL_TYPE.ANY;
5093
5094     if (portIndex < types.length) {
5095         return types[portIndex] || SIGNAL_TYPE.ANY;
5096     }
5097
5098     return types[types.length - 1] || SIGNAL_TYPE.ANY;
5099 }
5100 /**
5101 * Проверка совместимости типов сигналов
5102 *
5103 * Новая логика:
5104 * - ANY совместим со всем
5105 * - TRUE совместим с LOGIC, TRUE, ANY
5106 * - FALSE совместим с LOGIC, FALSE, ANY
5107 * - LOGIC совместим с LOGIC, TRUE, FALSE, ANY
5108 * - NUMERIC совместим с NUMERIC, ANY
5109 */
5110 function areTypesCompatible(outputType, inputType) {
```

```
5111 // Если один из типов ANY - совместимы
5112 if (outputType === SIGNAL_TYPE.ANY || inputType === SIGNAL_TYPE.ANY) {
5113     return true;
5114 }
5115
5116 // Если типы одинаковые - совместимы
5117 if (outputType === inputType) {
5118     return true;
5119 }
5120
5121 // TRUE/FALSE совместимы с LOGIC
5122 if ((outputType === SIGNAL_TYPE.TRUE || outputType === SIGNAL_TYPE.FALSE) &&
5123     inputType === SIGNAL_TYPE.LOGIC) {
5124     return true;
5125 }
5126
5127 // LOGIC совместим с TRUE/FALSE (в случае если ожидается конкретный тип)
5128 if (outputType === SIGNAL_TYPE.LOGIC &&
5129     (inputType === SIGNAL_TYPE.TRUE || inputType === SIGNAL_TYPE.FALSE)) {
5130     return true;
5131 }
5132
5133     return false;
5134 }
5135
5136 /**
5137 * Проверка, находится ли элемент внутри рамки
5138 */
5139 function isInsideFrame(elemId, frameId) {
5140     const elem = AppState.elements[elemId];
5141     const frame = AppState.elements[frameId];
5142
5143     if (!elem || !frame || frame.type !== 'output-frame') return false;
5144
5145     const elemCenterX = elem.x + elem.width / 2;
5146     const elemCenterY = elem.y + elem.height / 2;
5147
5148     return elemCenterX > frame.x &&
5149         elemCenterX < frame.x + frame.width &&
5150         elemCenterY > frame.y &&
5151         elemCenterY < frame.y + frame.height;
5152 }
5153
5154 /**
5155 * Обновить принадлежность элементов к рамкам
5156 */
5157 function updateFrameChildren() {
5158     // Сначала очистим children у рамок и parentFrame у всех элементов
5159     Object.values(AppState.elements).forEach(elem => {
5160         if (elem.type === 'output-frame') {
5161             elem.children = [];
5162         } else {
5163             // удаляем parentFrame по умолчанию (пересчитаем ниже)
5164             if (elem.parentFrame) delete elem.parentFrame;
5165         }
5166     });
5167
5168     // Назначаем принадлежность: для каждого элемента ищем рамку, в которую он
5169     // попадает
5170     Object.values(AppState.elements).forEach(elem => {
5171         if (!elem || elem.type === 'output-frame') return;
5172
5173         Object.values(AppState.elements).forEach(frame => {
5174             if (!frame || frame.type !== 'output-frame') return;
```

```
5175     if (isInsideFrame(elem.id, frame.id)) {
5176         // добавляем в массив детей рамки
5177         frame.children.push(elem.id);
5178         // отмечаем у элемента родительскую рамку
5179         if (AppState.elements[elem.id]) {
5180             AppState.elements[elem.id].parentFrame = frame.id;
5181         }
5182     }
5183   });
5184 });
5185 }
5186
5187 /**
5188 * Преобразование координат экрана в координаты холста
5189 */
5190 function screenToCanvas(screenX, screenY) {
5191     const container = document.getElementById('workspace-container');
5192     const rect = container.getBoundingClientRect();
5193
5194     const x = (screenX - rect.left - AppState.viewport.panX) / AppState.viewport.zoom;
5195     const y = (screenY - rect.top - AppState.viewport.panY) / AppState.viewport.zoom;
5196
5197     return { x, y };
5198 }
5199
5200 /**
5201 * Преобразование координат холста в координаты экрана
5202 */
5203 function canvasToScreen(canvasX, canvasY) {
5204     const container = document.getElementById('workspace-container');
5205     const rect = container.getBoundingClientRect();
5206
5207     const x = canvasX * AppState.viewport.zoom + AppState.viewport.panX + rect.left;
5208     const y = canvasY * AppState.viewport.zoom + AppState.viewport.panY + rect.top;
5209
5210     return { x, y };
5211 }
5212
5213 /**
5214 * Проверка, является ли порт выходным (не подключен к другим элементам)
5215 */
5216 function isOutputPort(elemId, portIndex) {
5217     const portKey = `out-${portIndex}`;
5218
5219     // Проверяем, есть ли соединения от этого порта
5220     const hasConnection = AppState.connections.some(conn =>
5221         conn.fromElement === elemId && conn.fromPort === portKey
5222     );
5223
5224     return !hasConnection;
5225 }
5226
5227 /**
5228 * Получить информацию о выходном порте
5229 */
5230 function getOutputPortInfo(elemId, portIndex) {
5231     const elem = AppState.elements[elemId];
5232     if (!elem) return null;
5233
5234     const config = ELEMENT_TYPES[elem.type];
5235     if (!config) return null;
5236
5237     return {
5238         elementId: elemId,
5239         elementType: elem.type,
```

```
5240     elementName: config.name,
5241     portIndex: portIndex,
5242     portLabel: config.outputLabels?.[portIndex] || `out${portIndex}`,
5243     portType: config.outputTypes?.[portIndex] || SIGNAL_TYPE.ANY,
5244     // Дополнительная информация для идентификации
5245     displayName: `${config.name} → ${config.outputLabels?.[portIndex]} || `out$`[portIndex]``
5246   );
5247 }
5248
5249 viewport.js:
5250 /**
5251 * Модуль управления viewport (масштабирование и перемещение)
5252 */
5253
5254 const Viewport = {
5255   /**
5256    * Инициализация viewport
5257    */
5258   init() {
5259     this.setupZoomControls();
5260     this.setupPanning();
5261     this.setupMouseWheel();
5262     this.setupMinimap();
5263     this.setupCursorPosition();
5264     this.updateTransform();
5265     const container = document.getElementById('workspace-container');
5266     const rect = container.getBoundingClientRect();
5267     AppState.viewport.panX = 100; // немного отступить от левого края
5268     AppState.viewport.panY = (rect.height / 2) - 2500 * 0.5 *
AppState.viewport.zoom;
5269     this.updateTransform();
5270   },
5271
5272   /**
5273    * Настройка кнопок масштабирования
5274    */
5275   setupZoomControls() {
5276     document.getElementById('btn-zoom-in').addEventListener('click', () => {
5277       this.setZoom(AppState.viewport.zoom + VIEWPORT_CONFIG.zoomStep);
5278     });
5279
5280     document.getElementById('btn-zoom-out').addEventListener('click', () => {
5281       this.setZoom(AppState.viewport.zoom - VIEWPORT_CONFIG.zoomStep);
5282     });
5283
5284     document.getElementById('btn-zoom-reset').addEventListener('click', () => {
5285       this.setZoom(1);
5286       this.setPan(0, 0);
5287     });
5288
5289     document.getElementById('btn-zoom-fit').addEventListener('click', () => {
5290       this.fitToContent();
5291     });
5292   },
5293
5294   /**
5295    * Настройка перемещения (пан)
5296    */
5297   setupPanning() {
5298     const container = document.getElementById('workspace-container');
5299
5300     container.addEventListener('mousedown', (e) => {
5301       // Средняя кнопка мыши или пробел + левая кнопка
5302       if (e.button === 1 || (e.button === 0 && e.target === container)) {
```

```
5303         e.preventDefault();
5304         AppState.viewport.isPanning = true;
5305         AppState.viewport.lastMouseX = e.clientX;
5306         AppState.viewport.lastMouseY = e.clientY;
5307         container.style.cursor = 'grabbing';
5308     }
5309 });
5310
5311 document.addEventListener('mousemove', (e) => {
5312     if (AppState.viewport.isPanning) {
5313         const dx = e.clientX - AppState.viewport.lastMouseX;
5314         const dy = e.clientY - AppState.viewport.lastMouseY;
5315
5316         this.setPan(
5317             AppState.viewport.panX + dx,
5318             AppState.viewport.panY + dy
5319         );
5320
5321         AppState.viewport.lastMouseX = e.clientX;
5322         AppState.viewport.lastMouseY = e.clientY;
5323     }
5324 });
5325
5326 document.addEventListener('mouseup', (e) => {
5327     if (AppState.viewport.isPanning) {
5328         AppState.viewport.isPanning = false;
5329         document.getElementById('workspace-container').style.cursor = '';
5330     }
5331 });
5332
5333 // Клавиша пробел для режима перемещения
5334 document.addEventListener('keydown', (e) => {
5335     if (e.code === 'Space' && !e.repeat) {
5336         document.getElementById('workspace-container').style.cursor = 'grab';
5337     }
5338 });
5339
5340 document.addEventListener('keyup', (e) => {
5341     if (e.code === 'Space') {
5342         document.getElementById('workspace-container').style.cursor = '';
5343     }
5344 });
5345 },
5346
5347 /**
5348 * Настройка масштабирования колесом мыши
5349 */
5350 setupMouseWheel() {
5351     const container = document.getElementById('workspace-container');
5352
5353     container.addEventListener('wheel', (e) => {
5354         e.preventDefault();
5355
5356         const rect = container.getBoundingClientRect();
5357         const mouseX = e.clientX - rect.left;
5358         const mouseY = e.clientY - rect.top;
5359
5360         // Позиция мыши на холсте до масштабирования
5361         const canvasPosBeforeX = (mouseX - AppState.viewport.panX) /
5362             AppState.viewport.zoom;
5363         const canvasPosBeforeY = (mouseY - AppState.viewport.panY) /
5364             AppState.viewport.zoom;
5365
5366         // Новый масштаб
5367         const delta = e.deltaY > 0 ? -VIEWPORT_CONFIG.zoomStep :
```

```
VIEWPORT_CONFIG.zoomStep;
5366     const newZoom = Math.max(
5367         VIEWPORT_CONFIG.minZoom,
5368         Math.min(VIEWPORT_CONFIG.maxZoom, AppState.viewport.zoom + delta)
5369     );
5370
5371     // Корректируем pan, чтобы точка под курсором осталась на месте
5372     const newPanX = mouseX - canvasPosBeforeX * newZoom;
5373     const newPanY = mouseY - canvasPosBeforeY * newZoom;
5374
5375     AppState.viewport.zoom = newZoom;
5376     AppState.viewport.panX = newPanX;
5377     AppState.viewport.panY = newPanY;
5378
5379     this.updateTransform();
5380 }, { passive: false });
5381 },
5382 /**
5383 * Установить масштаб
5384 */
5385 setZoom(zoom) {
5386     const container = document.getElementById('workspace-container');
5387     const rect = container.getBoundingClientRect();
5388
5389     // Центр экрана
5390     const centerX = rect.width / 2;
5391     const centerY = rect.height / 2;
5392
5393     // Позиция центра на холсте
5394     const canvasCenterX = (centerX - AppState.viewport.panX) /
5395     AppState.viewport.zoom;
5396     const canvasCenterY = (centerY - AppState.viewport.panY) /
5397     AppState.viewport.zoom;
5398
5399     // Новый масштаб
5400     const newZoom = Math.max(
5401         VIEWPORT_CONFIG.minZoom,
5402         Math.min(VIEWPORT_CONFIG.maxZoom, zoom)
5403     );
5404
5405     // Корректируем pan
5406     AppState.viewport.panX = centerX - canvasCenterX * newZoom;
5407     AppState.viewport.panY = centerY - canvasCenterY * newZoom;
5408     AppState.viewport.zoom = newZoom;
5409
5410     this.updateTransform();
5411 },
5412 /**
5413 * Установить смещение
5414 */
5415 setPan(x, y) {
5416     AppState.viewport.panX = x;
5417     AppState.viewport.panY = y;
5418     this.updateTransform();
5419 },
5420 /**
5421 * Вписать содержимое в экран
5422 */
5423 fitToContent() {
5424     const elements = Object.values(AppState.elements);
5425     if (elements.length === 0) {
5426         this.setZoom(1);
```

```
5428         this.setPan(0, 0);
5429         return;
5430     }
5431
5432     // Находим границы содержимого
5433     let minX = Infinity, minY = Infinity;
5434     let maxX = -Infinity, maxY = -Infinity;
5435
5436     elements.forEach(elem => {
5437         minX = Math.min(minX, elem.x);
5438         minY = Math.min(minY, elem.y);
5439         maxX = Math.max(maxX, elem.x + elem.width);
5440         maxY = Math.max(maxY, elem.y + elem.height);
5441     });
5442
5443     const contentWidth = maxX - minX;
5444     const contentHeight = maxY - minY;
5445
5446     const container = document.getElementById('workspace-container');
5447     const rect = container.getBoundingClientRect();
5448
5449     const padding = 50;
5450     const availableWidth = rect.width - padding * 2;
5451     const availableHeight = rect.height - padding * 2;
5452
5453     const zoomX = availableWidth / contentWidth;
5454     const zoomY = availableHeight / contentHeight;
5455     const newZoom = Math.min(zoomX, zoomY, 1);
5456
5457     AppState.viewport.zoom = Math.max(VIEWPORT_CONFIG.minZoom, newZoom);
5458     AppState.viewport.panX = padding - minX * AppState.viewport.zoom +
5459     (availableWidth - contentWidth * AppState.viewport.zoom) / 2;
5460     AppState.viewport.panY = padding - minY * AppState.viewport.zoom +
5461     (availableHeight - contentHeight * AppState.viewport.zoom) / 2;
5462
5463     this.updateTransform();
5464 },
5465 /**
5466 * Обновить трансформацию
5467 */
5468 updateTransform() {
5469     const workspace = document.getElementById('workspace');
5470     const svg = document.getElementById('connections-svg');
5471
5472     const transform = `translate(${AppState.viewport.panX}px, ${AppState.viewport.panY}px) scale(${AppState.viewport.zoom})`;
5473     workspace.style.transform = transform;
5474     svg.style.transform = transform;
5475
5476     // Обновляем отображение масштаба
5477     document.getElementById('zoom-level').textContent = `${Math.round(AppState.viewport.zoom * 100)}%`;
5478
5479     // Обновляем мини-карту
5480     this.updateMinimap();
5481 },
5482 /**
5483 * Настройка мини-карты
5484 */
5485 setupMinimap() {
5486     const minimap = document.getElementById('minimap');
5487     const canvas = document.getElementById('minimap-canvas');
```

```
5489     canvas.width = MINIMAP_CONFIG.width;
5490     canvas.height = MINIMAP_CONFIG.height;
5491
5492     // Клик по мини-карте для перемещения
5493     minimap.addEventListener('click', (e) => {
5494         const rect = minimap.getBoundingClientRect();
5495         const x = e.clientX - rect.left;
5496         const y = e.clientY - rect.top;
5497
5498         this.navigateToMinimapPosition(x, y);
5499     });
5500 },
5501
5502 /**
5503 * Обновить мини-карту
5504 */
5505 updateMinimap() {
5506     const canvas = document.getElementById('minimap-canvas');
5507     const ctx = canvas.getContext('2d');
5508     const viewportEl = document.getElementById('minimap-viewport');
5509
5510     // Очищаем
5511     ctx.fillStyle = '#0a0ala';
5512     ctx.fillRect(0, 0, canvas.width, canvas.height);
5513
5514     // Масштаб мини-карты
5515     const scale = Math.min(
5516         canvas.width / VIEWPORT_CONFIG.canvasWidth,
5517         canvas.height / VIEWPORT_CONFIG.canvasHeight
5518     );
5519
5520     // Рисуем элементы
5521     Object.values(AppState.elements).forEach(elem => {
5522         const x = elem.x * scale;
5523         const y = elem.y * scale;
5524         const w = Math.max(elem.width * scale, 2);
5525         const h = Math.max(elem.height * scale, 2);
5526
5527         ctx.fillStyle = ELEMENT_TYPES[elem.type]?.color || '#4a90d9';
5528         ctx.fillRect(x, y, w, h);
5529     });
5530
5531     // Рисуем viewport
5532     const container = document.getElementById('workspace-container');
5533     const rect = container.getBoundingClientRect();
5534
5535     const vpX = (-AppState.viewport.panX / AppState.viewport.zoom) * scale;
5536     const vpY = (-AppState.viewport.panY / AppState.viewport.zoom) * scale;
5537     const vpW = (rect.width / AppState.viewport.zoom) * scale;
5538     const vpH = (rect.height / AppState.viewport.zoom) * scale;
5539
5540     viewportEl.style.left = `${vpX}px`;
5541     viewportEl.style.top = `${vpY}px`;
5542     viewportEl.style.width = `${vpW}px`;
5543     viewportEl.style.height = `${vpH}px`;
5544
5545 },
5546
5547 /**
5548 * Перейти к позиции на мини-карте
5549 */
5550 navigateToMinimapPosition(minimapX, minimapY) {
5551     const canvas = document.getElementById('minimap-canvas');
5552     const container = document.getElementById('workspace-container');
5553     const rect = container.getBoundingClientRect();
```

```
5554
5555     const scale = Math.min(
5556         canvas.width / VIEWPORT_CONFIG.canvasWidth,
5557         canvas.height / VIEWPORT_CONFIG.canvasHeight
5558     );
5559
5560     const canvasX = minimapX / scale;
5561     const canvasY = minimapY / scale;
5562
5563     // Центрируем viewport на этой точке
5564     AppState.viewport.panX = rect.width / 2 - canvasX * AppState.viewport.zoom;
5565     AppState.viewport.panY = rect.height / 2 - canvasY * AppState.viewport.zoom;
5566
5567     this.updateTransform();
5568 },
5569 /**
5570  * Отслеживание позиции курсора
5571 */
5572 setupCursorPosition() {
5573     const container = document.getElementById('workspace-container');
5574
5575     container.addEventListener('mousemove', (e) => {
5576         const pos = screenToCanvas(e.clientX, e.clientY);
5577         document.getElementById('cursor-pos').textContent =
5578             `X: ${Math.round(pos.x)}, Y: ${Math.round(pos.y)} `;
5579     });
5580 }
5581 }
5582 };
5583
5584 main.py:
5585 import os
5586 import json
5587 from typing import Dict, List
5588 import pandas as pd
5589 from fastapi import FastAPI, HTTPException, Request, Response
5590 from fastapi.responses import JSONResponse
5591 from fastapi.staticfiles import StaticFiles
5592
5593 BASE_DIR = os.path.dirname(os.path.abspath(__file__))
5594 SETTINGS_PATH = os.path.join(BASE_DIR, "settings.json")
5595
5596 def load_settings() -> Dict:
5597     with open(SETTINGS_PATH, "r", encoding="utf-8") as f:
5598         return json.load(f)
5599
5600 def load_signals_from_folder(folder: str) -> List[Dict]:
5601     # folder может быть относительным
5602     folder_abs = folder if os.path.isabs(folder) else
5603     os.path.normpath(os.path.join(BASE_DIR, folder))
5604     if not os.path.isdir(folder_abs):
5605         raise FileNotFoundError(f"signalDataFolder not found: {folder_abs}")
5606
5607     signals_map = {} # Tagname -> Description (последний wins)
5608     for name in os.listdir(folder_abs):
5609         if not name.lower().endswith(".csv"):
5610             continue
5611         path = os.path.join(folder_abs, name)
5612         try:
5613             df = pd.read_csv(path, sep=';')[['Tagname', 'Description']]
5614             df = df.dropna(subset=['Tagname'])
5615             for _, row in df.iterrows():
5616                 tag = str(row['Tagname']).strip()
5617                 desc = "" if pd.isna(row['Description']) else
5618                 str(row['Description']).strip()
```

```
5617         if tag:
5618             signals_map[tag] = desc
5619     except Exception as e:
5620         # пропускаем "плохие" csv, но можно логировать
5621         print(f"[WARN] failed to read {path}: {e}")
5622
5623     # в список
5624     out = [{'Tagname': k, 'Description': v} for k, v in signals_map.items()]
5625     out.sort(key=lambda x: x['Tagname'])
5626     return out
5627
5628
5629
5630 app = FastAPI()
5631
5632 # Кэш сигналов в памяти
5633 STATE = {
5634     "settings": None,
5635     "signals": None
5636 }
5637
5638 @app.on_event("startup")
5639 def startup():
5640     settings = load_settings()
5641     STATE["settings"] = settings
5642     folder = settings.get("signalDataFolder")
5643     if not folder:
5644         raise RuntimeError("settings.json: signalDataFolder is required")
5645     STATE["signals"] = load_signals_from_folder(folder)
5646     print(f"[OK] loaded signals: {len(STATE['signals'])}")
5647
5648 @app.get("/api/settings")
5649 def api_settings():
5650     return STATE["settings"]
5651
5652 @app.get("/api/signals")
5653 def api_signals(q: str = "", limit: int = 50):
5654     """
5655     q – маска со * (например *МАА*CP*)
5656     """
5657     signals = STATE["signals"] or []
5658     if not q:
5659         return {"items": signals[:limit], "total": len(signals)}
5660
5661     # маска * -> regex
5662     import re
5663     escaped = re.escape(q).replace(r"\*", ".*")
5664     rx = re.compile("^" + escaped + "$", re.IGNORECASE)
5665
5666     items = [s for s in signals if rx.match(s["Tagname"])]
5667     return {"items": items[:max(1, min(limit, 500))], "total": len(items)}
5668
5669 # Helper: возвращает абсолютный путь к файлу проекта
5670 def get_project_path(filename: str):
5671     folder = STATE["settings"].get("projectDataFolder")
5672     if not folder:
5673         raise RuntimeError("projectDataFolder not configured")
5674
5675     # Нормализуем путь к папке проектов
5676     project_dir = folder if os.path.isabs(folder) else
5677     os.path.normpath(os.path.join(BASE_DIR, folder))
5678
5679     # Проверяем, что filename безопасен (не пытается выйти за пределы папки)
5680     if '..' in filename or '/' in filename or '\\\\' in filename:
5681         raise HTTPException(status_code=400, detail="Invalid filename")
```

```
5681     path = os.path.join(project_dir, filename)
5682     # Проверяем, что итоговый путь лежит внутри разрешенной директории
5683     if not path.startswith(project_dir):
5684         raise HTTPException(status_code=400, detail="Path traversal attempt")
5685
5686     return path
5687
5688 @app.post("/api/project/save")
5689 async def save_project(request: Request):
5690     try:
5691         data = await request.json()
5692         filename = data.get("filename")
5693         content = data.get("content")
5694
5695         if not filename or not content:
5696             raise HTTPException(status_code=400, detail="Filename and content are
5697 required")
5698
5699         path = get_project_path(filename)
5700
5701         # Сохраняем как JSON
5702         with open(path, "w", encoding="utf-8") as f:
5703             json.dump(content, f, indent=2)
5704
5705         return {"status": "ok", "message": f"Project saved to {filename}"}
5706
5707     except HTTPException as e:
5708         raise e
5709     except Exception as e:
5710         print(f"Error saving project: {e}")
5711         raise HTTPException(status_code=500, detail="Internal server error during
5712 save")
5713
5714 @app.get("/api/project/load/{filename}")
5715 def load_project(filename: str):
5716     try:
5717         path = get_project_path(filename)
5718
5719         if not os.path.exists(path):
5720             raise HTTPException(status_code=404, detail="Project not found")
5721
5722         with open(path, "r", encoding="utf-8") as f:
5723             content = json.load(f)
5724
5725         return content
5726
5727     except HTTPException as e:
5728         raise e
5729     except Exception as e:
5730         print(f"Error loading project: {e}")
5731         raise HTTPException(status_code=500, detail="Internal server error during
load")
5732
5733 @app.get("/api/project/list")
5734 def list_projects():
5735     folder = STATE["settings"].get("projectDataFolder")
5736     if not folder:
5737         raise HTTPException(status_code=500, detail="Project folder not configured")
5738
5739     project_dir = folder if os.path.isabs(folder) else
5740     os.path.normpath(os.path.join(BASE_DIR, folder))
5741     os.makedirs(project_dir, exist_ok=True)
5742
5743     projects = []
```

```
5742     for fname in sorted(os.listdir(project_dir)):
5743         if not fname.endswith(".json"):
5744             continue
5745         path = os.path.join(project_dir, fname)
5746         try:
5747             with open(path, "r", encoding="utf-8") as f:
5748                 payload = json.load(f)
5749         except Exception:
5750             continue
5751         project_meta = payload.get("project", {})
5752         projects.append({
5753             "filename": fname,
5754             "code": project_meta.get("code") or project_meta.get("tagname") or "",
5755             "description": project_meta.get("description") or "",
5756             "type": project_meta.get("type") or ""
5757         })
5758     return {"projects": projects}
5759
5760
5761 # Раздаём фронтенд
5762 WEB_DIR = os.path.normpath(os.path.join(BASE_DIR, "..", "web"))
5763 app.mount("/", StaticFiles(directory=WEB_DIR, html=True), name="web")
```