Kapitel 2 - Codierer und Multiplexer

Kombinatorische Schaltungen

Definition nach DIN 44300/93:
Ein Schaltnetz oder kombinatorischer Funktionsblock ist eine Funktionseinheit zum Verarbeiten von Schaltvariablen, deren Wert am Ausgang zu beliebigem Zeitpunkt nur vom Eingangswert zum gleichen Zeitpunkt abhängt.

Codierer und Decodierer

Ein Code ist (im Kontext der digitalen Informationsverarbeitung) ein Bitstring zur eindeutigen Bezeichnung eines Objekts. Eine Codierung ist dann die Erzeugung und Zuordnung von Objekten zu Codes. Formal ist eine Codierung C eine Abbildung:

C: M $\rightarrow$ $\{0,1\}^n$, wobei n die Länge eines Codes ist und M die Menge der Objekte.

Es gibt nun vielfältige Möglichkeiten, die Funktionsvorschrift festzulegen. Im Allgemeinen möchte man die Länge der Codes so kurz wie möglich halten. Außerdem sollten alle Codes gleich lang sein.

Sei | M | die Anzahl der zu codierenden Objekte (z.B. | M | = 1000). Alle Codes sollen dieselbe Länge haben.

Wie lang (n) müssen die Codes mindestens sein?

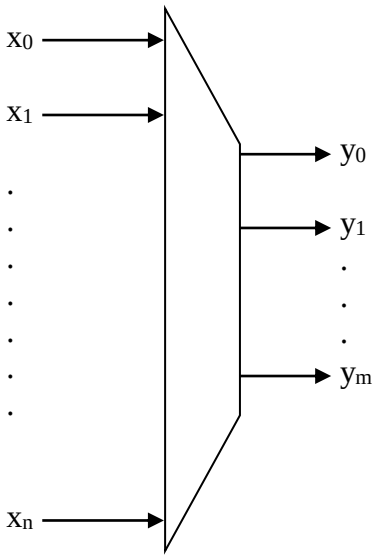
Wir betrachten im Folgenden nur noch Codierungen, die Codes gleicher Länge erzeugen. Nach der Erzeugung von Codes betrachten wir nun die Umkehrung, d.h. die Entschlüsselung von gegebenen Codes. Dabei soll für eine vorgegebene Menge von Codes jedem Code ein Objekt zugeordnet werden. Keine zwei Codes dürfen dasselbe Objekt bezeichnen. Formal ist eine Decodierung D eine Abbildung:

D: $\{0,1\}^n$ $\rightarrow$ M, wobei n die Länge der Codes ist und M die Menge der Objekte.

Wie groß ist | M |, wenn alle Elemente aus $\{0,1\}^n$ decodiert werden sollen?

Codierer

In der Digitaltechnik werden Codierer durch Trapeze dargestellt:

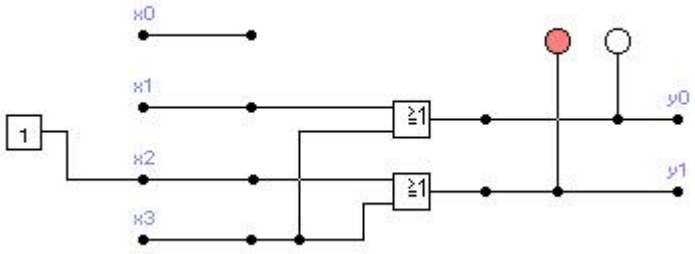


Die x_i sind die Eingänge, die y_j die Ausgänge. Jede Eingangsleitung steht für ein Objekt, zu dem ein Code = Bitstring $y_m \ldots y_0$ erzeugt werden soll. Es wird vorausgesetzt, dass immer nur genau eine Eingangsleitung aktiv (= 1) ist und alle anderen nicht (= 0). Ein Code soll, als Betragszahl $y_m \ldots y_0$ interpretiert, den Index der aktiven Eingangsleitung darstellen.

Beispiel: n=15, m=3, x_{12}=1, Rest=0, dann soll der Codierer $y_3 \ldots y_0$ = 1100 liefern.

Überlegungen zur Konstruktion am Beispiel eines 4 zu 2 Codierers:
Eingaben: $x_3x_2x_1x_0$: 0001, 0010, 0100, 1000
Ausgaben: y_1y_0: 00, 01, 10, 11

Ist der Index des aktiven Eingangs = 2, muss y_1 auf 1 und y_0 auf 0 gesetzt werden:

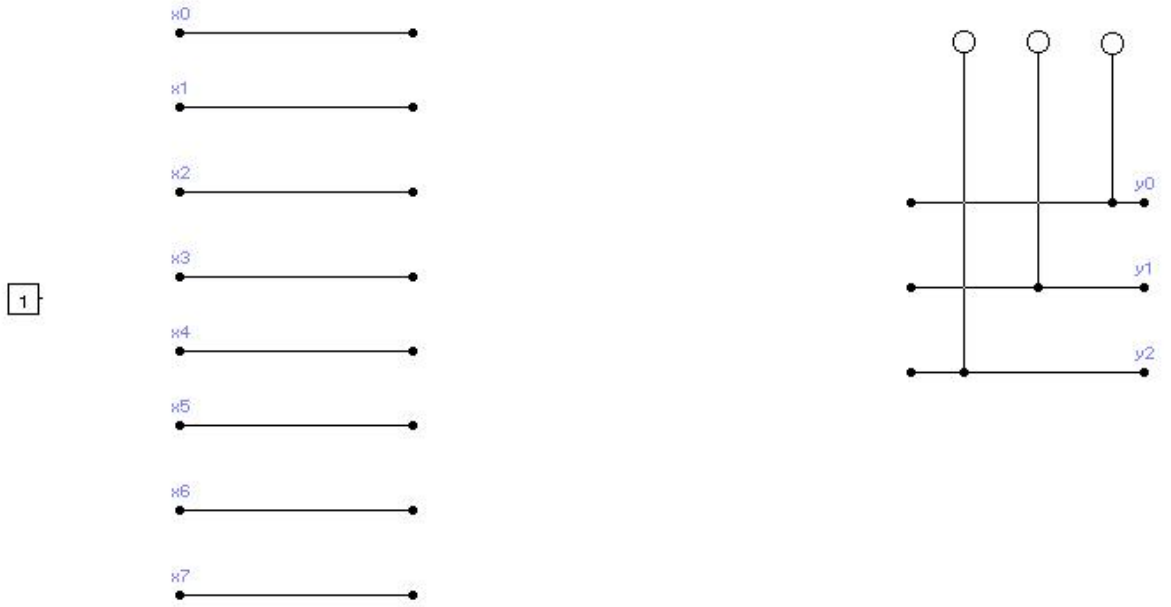


Versuch 200 8-zu-3-Codierer

Bauen Sie in der Datei v200 einen 8-zu-3-Codierer, der folgende Codierungsvorschrift umsetzt.

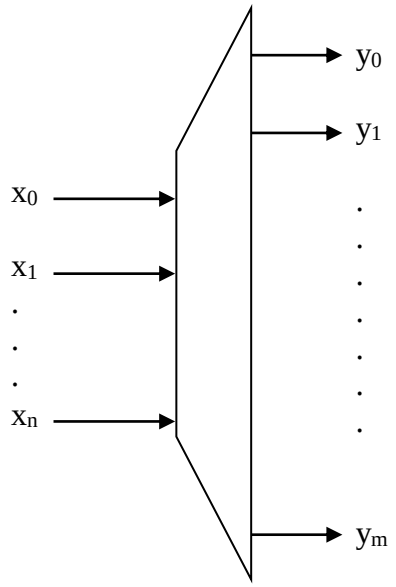
Jede Eingangsleitung steht für ein Objekt, zu dem ein Code (Bitstring $y_2y_1y_{0)}$ erzeugt werden soll. Es wird vorausgesetzt, dass immer nur genau eine Eingangsleitung aktiv (= 1) ist und alle anderen anderen nicht (= 0). Ein Code soll, als Betragszahl $y_2y_1y_0$ interpretiert, den Index der aktiven Eingangsleitung darstellen.

Hinweis: Durch Doppelklick auf ein Gatter kann die Anzahl der Eingänge verändert werden.



Decodierer

Decodierer werden ebenfalls durch Trapeze dargestellt:

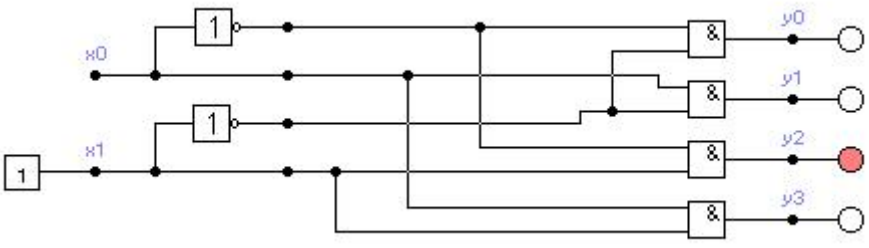


Die x_i sind die Eingänge, die y_j die Ausgänge. Die Belegung aller Eingänge durch Nullen und Einsen stellt einen Code dar. Der Decodierer aktiviert (= 1 setzen) diejenige Ausgangsleitung y_i, deren Index dem Bitstring $x_n \ldots x_0$, interpretiert als Betragszahl, entspricht. Alle anderen Ausgänge werden auf Null gesetzt.

Ein Beispiel: n=3, m=15, $x_3x_2x_1x_0 = 1000$, dann soll der Decodierer $y_8 = 1$ und alle anderen $y_i = 0$ setzen.

Überlegungen zur Konstruktion am Beispiel eines 2 zu 4 Decodierers:
Eingaben: x_1x_0: 00, 01, 10, 11
Ausgaben: $y_3\ y_2y_1y_0$: 0001, 0010, 0100, 1000

y_2 soll nur genau dann 1 sein, wenn $x_0 = 0$ und $x_1 = 1$ ist.



Versuch 210 3-zu-8-Decodierer

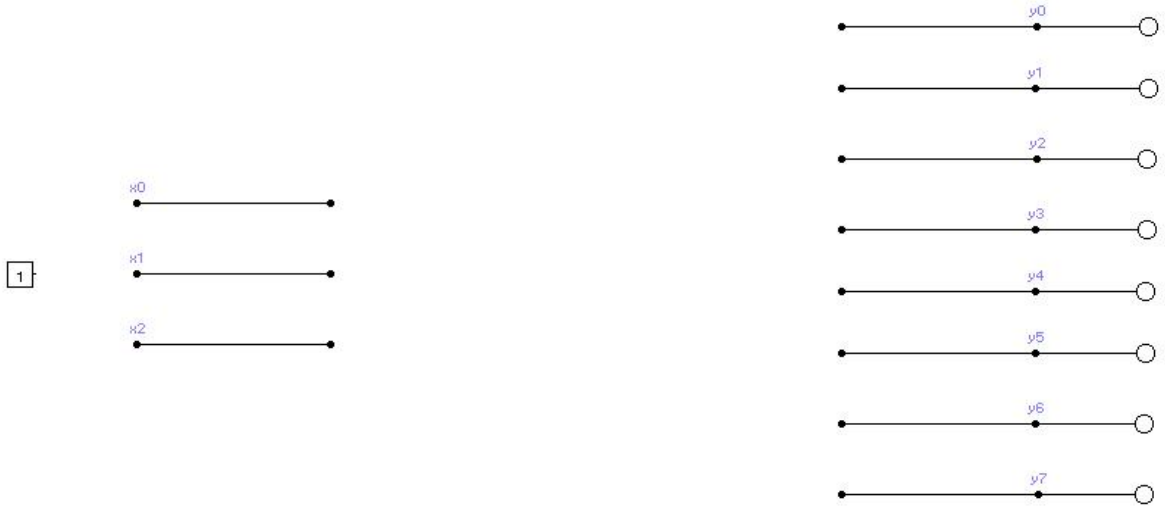
Bauen Sie in der Datei v210 einen 3-zu-8-Decodierer mit folgender Funktion:

Die x_i sind die Eingänge, die y_j die Ausgänge. Die Belegung aller Eingänge durch Nullen und Einsen stellt einen Code dar. Der Decodierer aktiviert (= 1 setzen) diejenige Ausgangsleitung y_i, deren Index dem Bitstring $x_2x_1x_0$, interpretiert als Betragszahl, entspricht. Alle anderen Ausgänge werden auf Null gesetzt.

Nebenbedingung: Sie dürfen höchstens elf Gatter verwenden.

Hinweis: Durch Doppelklick auf ein Gatter kann die Anzahl der Eingänge verändert werden.

Datei v210:



Prioritätenencoder

Encoder ist nur ein anderes Wort (aus dem Engl. abgeleitet) für Codierer.

Bei 2^n-zu-n-Codierern wird gefordert, dass genau ein Eingang auf 1 und alle anderen auf 0 gesetzt werden. Diese Bedingung ist oft nicht einzuhalten bzw. auch gar nicht erwünscht. Beim Prioritätenencoder wird diese Bedingung fallengelassen.

Im Gegenzug muss die Semantik dieses Bausteins erweitert werden. Ist wie beim bisherigen Codierer nur genau eine Eingangsleitung 1, liefert auch der Prioritätenencoder einen Code, der, als Betragszahl interpretiert, dem Index der aktiven Eingangsleitung (=1) entspricht. Sind aber zwei oder mehr Eingangsleitungen 1, wertet der Baustein eine vorher festgelegte Priorität unter den Eingangsleitungen aus.

Beispiel für eine Prioritätenfestlegung: x_i hat eine höhere Priorität als x_j, falls $i < j$.

Der Encoder liefert dann den Code für den Index des Eingangs mit der höchsten Priorität.

Für obiges Beispiel bedeutet dies: Es wird der Eingang bestimmt, der mit 1 belegt ist und für den gilt, dass es keinen mit 1 belegten Eingang mit niedrigerem Index gibt.

Ausgabe: $\text{Min}(\{i \mid x_i = 1\})$

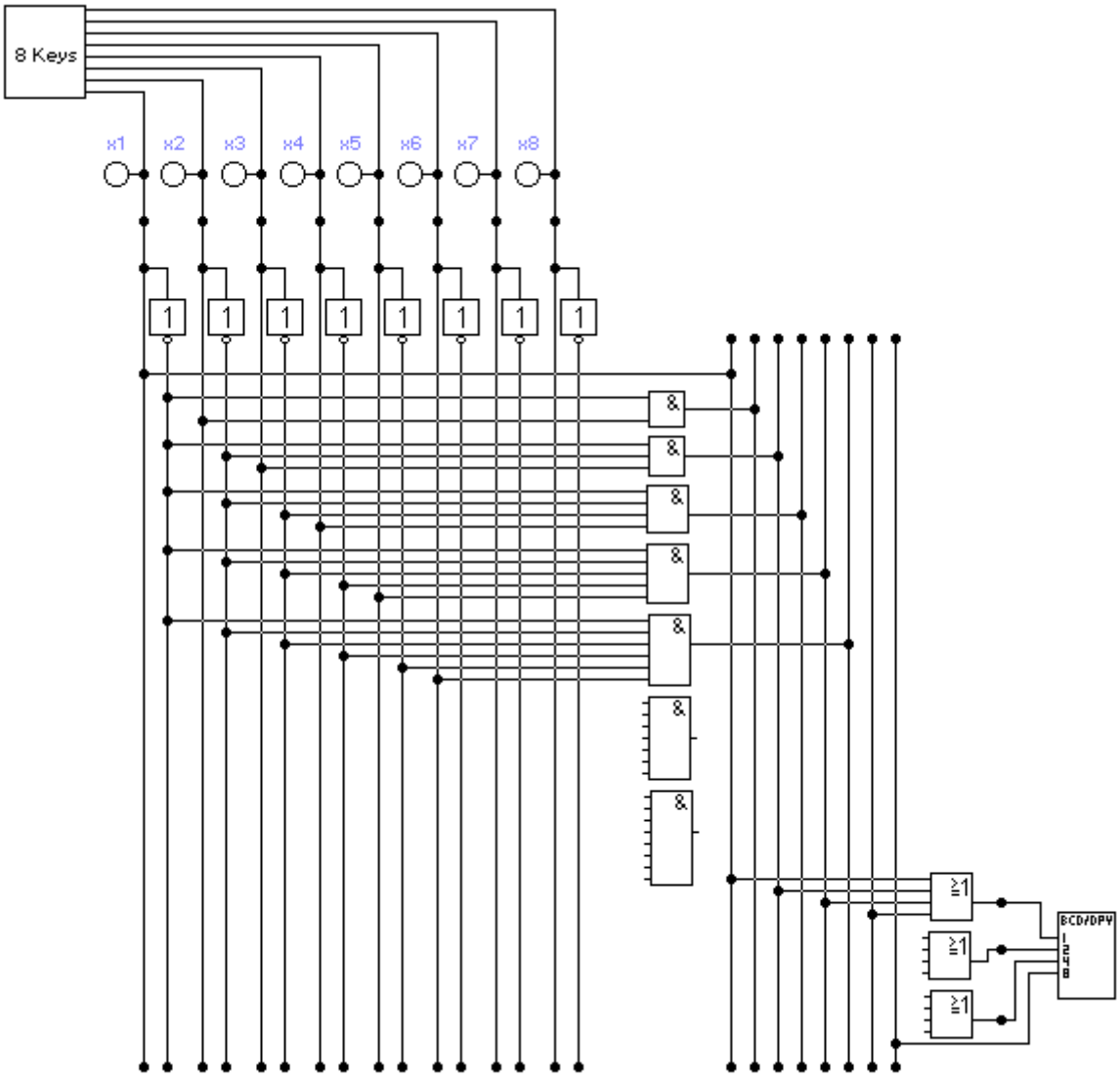
Für den so bestimmten Index wird die Betragszahl auf der 7-Segmentanzeige ausgegeben.

Versuch 220 Prioritätenencoder

Vervollständigen Sie in der Datei v220 die Schaltung eines Prioritätenencoders mit acht Eingängen und der Prioritätenfestlegung „Min ({i | x_i = 1}) hat die höchste Priorität“.

Anmerkung: Die Ausgabe wurde um ein Bit erweitert und die Eingänge von 1 bis 8 indiziert, um für den Fall, dass kein Eingang mit 1 belegt ist, eine 0 ausgeben zu können.

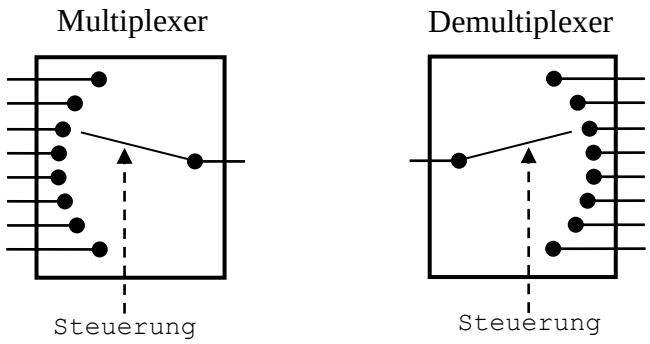
Testen Sie ihre Schaltung. Die Eingänge (8 Keys) können mit Hilfe der Tasten 1 bis 8 auf 0 oder 1 gesetzt werden.



Wie genau und wodurch wird die Prioritätensteuerung umgesetzt?

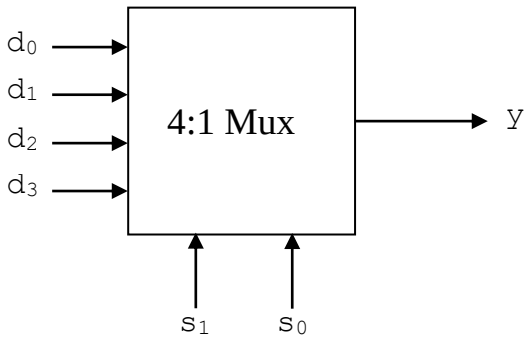
Multiplexer und Demultiplexer

In einem Rechner benötigt man zwischen Komponenten Leitungen, um Daten auszutauschen. Da ein voll vermaschtes Netz zwischen allen Komponenten sehr aufwändig und häufig auch nicht erforderlich ist, z.B. wenn nicht alle Datenwege gleichzeitig benötigt werden, braucht man je nach Transferrichtung einen „Sammler“ (Multiplexer) oder “Verteiler“ (Demultiplexer).



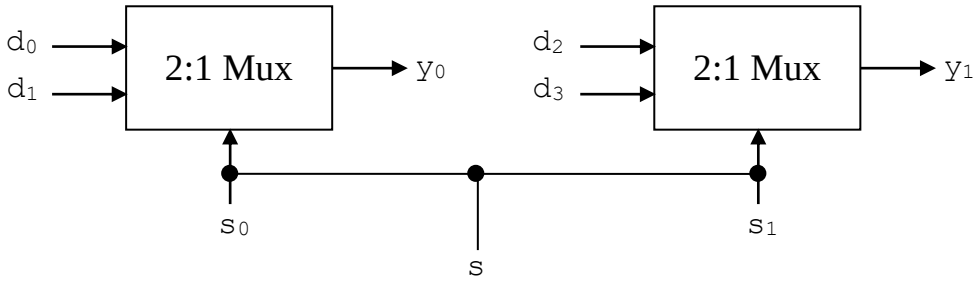
Versuch 230 Multiplexer

Multiplexer „sammeln“ Bündel von Leitungen und schalten eins der Eingangsbündel auf das Ausgangsbündel durch. Anwendungen finden sich bspw. in einem Rechenwerk, in dem eine ALU (Arithmetcal Logical Unit) ihre Operatoren aus mehreren verschiedenen Quellen beziehen kann. Hier ein Beispiel für einen 4 zu 1 Multiplexer:



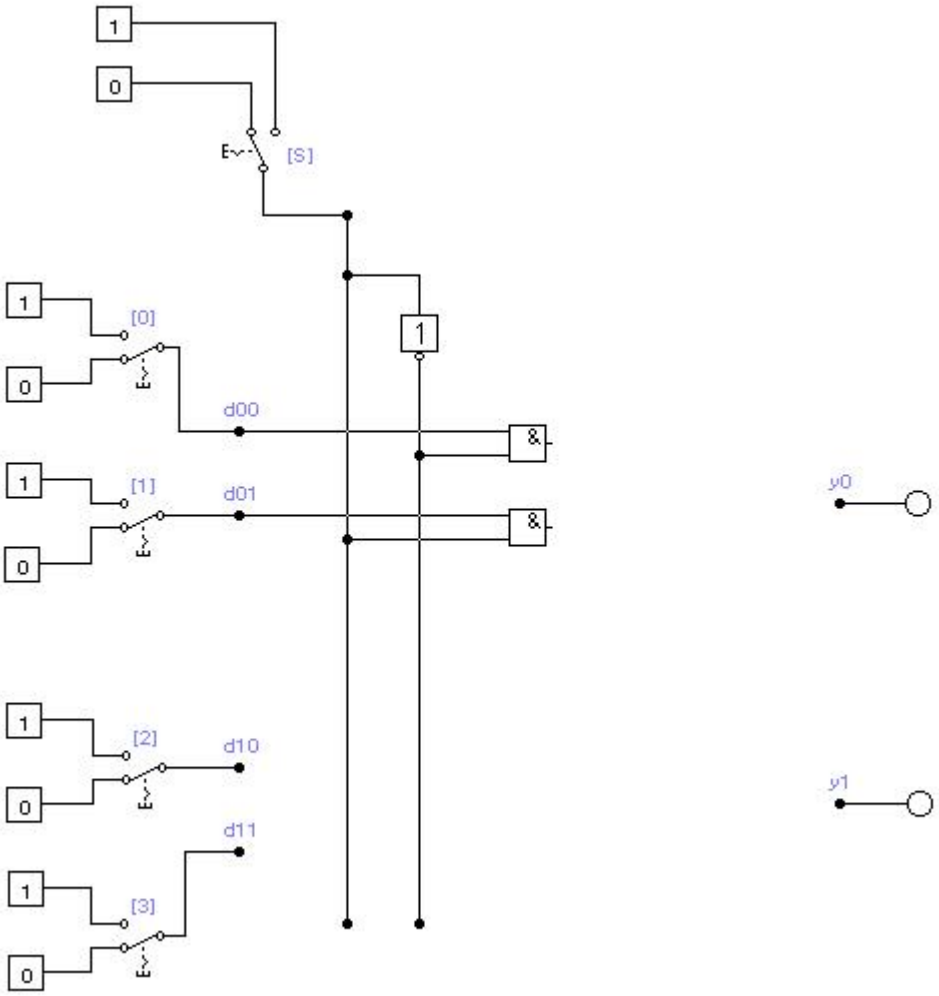
Einer der Dateneingänge d_i soll mit Hilfe der Steuereingänge s_0 und s_1 auf den Ausgang y durchgeschaltet werden.

Man kann auch mehrere Multiplexer in einem Bauteil (Chip) herstellen. Als Übungsbeispiel sei in v230 ein 2-fach 2 zu 1 Multiplexer aufgeführt, bei dem der Steuereingang der einzelnen 2 zu 1 Multiplexer zu einem einzigen Steuereingang zusammengefasst sind. Das Bauteil hat 4 Dateneingänge d_0, d_1, d_2, d_3 und 2 Ausgänge y_0, y_1 und einen Steuereingang s.



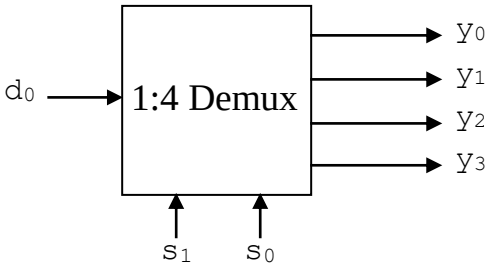
Ergänzen Sie die Schaltung in Datei v230 zu einem 2-fach 2 zu 1 Multiplexer.

Überlegungen zur Konstruktion: Bei s = 0 soll d_{00} auf y_0 und d_{10} auf y_1 durchgeschaltet werden. Bei s = 1 soll entsprechend die andere Datenleitung durchgeschaltet werden.



Versuch 240 Demultiplexer

Ein Demultiplexer ist ein Verteiler, der ein Bündel von Leitungen auf jeweils zwei oder mehr andere Bündel von Leitungen durchschaltet. Ein Beispiel:



Das Eingangsbündel besteht hier nur aus der Datenleitung d_0. Es gibt vier Ausgangsleitungen y_0 bis y_3. Die Steuereingänge s_0 und s_1 bestimmen, auf welches y die Leitung d_0 durchgeschaltet wird. Z.B. wird bei $s_1s_0 = 10$ d_0 auf y_2 durchgeschaltet.

Ergänzen Sie die Schaltung in Datei v240 zu einem 2-fach 1-zu-4-Demultiplexer.